Instrucciones Detalladas para Implementar Cola de Progreso Estilo Google Drive

Resumen Ejecutivo

Este documento proporciona las instrucciones específicas para modificar tu sistema actual de subida de archivos, transformando el modal básico con spinner en una cola de progreso avanzada que aparece en la esquina inferior derecha de la pantalla, similar a la implementación de Google Drive. Las modificaciones incluyen cambios específicos en archivos existentes y la creación de nuevos componentes CSS y JavaScript.

Modificaciones en modal.php

1. Eliminar el Spinner Básico

Ubicación: Líneas 25-30 aproximadamente

Código a ELIMINAR:

HTML

```
<!-- Spinner para indicar carga -->
<div id="upload-spinner" class="d-none text-center mt-3">
    <div class="spinner-border text-primary" role="status">
        <span class="visually-hidden">Cargando...</span>
    </div>
    <p class="mt-2">Subiendo...</p>
</div>
```

2. Modificar el Contenedor de Lista de Archivos

Ubicación: Línea 24 aproximadamente

Código ACTUAL:

HTML

```
<div id="file-list" class="mt-3"></div>
```

Código NUEVO:

HTML

```
<div id="file-list" class="mt-3"></div>
<div id="upload-status" class="mt-3 d-none">
    <div class="alert alert-info">
        <i class="material-icons me-2">cloud_upload</i>
        Los archivos se están subiendo. Puedes ver el progreso en la esquina
inferior derecha.
    </div>
</div>
```

Crear Nuevo Archivo: upload-progress.css

Ubicación: Crear en la carpeta de CSS de tu proyecto

Contenido completo del archivo:

CSS

```
/* Cola de Progreso Estilo Google Drive */
.upload-queue-container {
    position: fixed;
    bottom: 20px;
    right: 20px;
    width: 350px;
    max-height: 400px;
    background: #fff;
    border-radius: 8px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
    z-index: 9999;
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-
serif;
    display: none;
    overflow: hidden;
}

.upload-queue-header {
    padding: 16px 20px 12px;
    border-bottom: 1px solid #e8eaed;
    display: flex;
    align-items: center;
    justify-content: space-between;
    background: #f8f9fa;
}

.upload-queue-title {
    font-size: 14px;
    font-weight: 500;
```

```
    color: #202124;
    margin: 0;
}

.upload-queue-close {
    background: none;
    border: none;
    padding: 4px;
    cursor: pointer;
    border-radius: 4px;
    color: #5f6368;
    transition: background-color 0.2s;
}

.upload-queue-close:hover {
    background-color: #f1f3f4;
}

.upload-queue-body {
    max-height: 320px;
    overflow-y: auto;
    padding: 8px 0;
}

.upload-item {
    padding: 12px 20px;
    border-bottom: 1px solid #f1f3f4;
    transition: background-color 0.2s;
}

.upload-item:last-child {
    border-bottom: none;
}

.upload-item:hover {
    background-color: #f8f9fa;
}

.upload-item-header {
    display: flex;
    align-items: center;
    margin-bottom: 8px;
}

.upload-item-icon {
    width: 20px;
    height: 20px;
    margin-right: 12px;
```

```
    color: #5f6368;
    flex-shrink: 0;
}

.upload-item-name {
    flex: 1;
    font-size: 13px;
    color: #202124;
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
    margin-right: 8px;
}

.upload-item-status {
    font-size: 12px;
    color: #5f6368;
    flex-shrink: 0;
}

.upload-item-progress {
    display: flex;
    align-items: center;
    gap: 8px;
}

.upload-progress-bar {
    flex: 1;
    height: 4px;
    background-color: #e8eaed;
    border-radius: 2px;
    overflow: hidden;
}

.upload-progress-fill {
    height: 100%;
    background-color: #1a73e8;
    border-radius: 2px;
    transition: width 0.3s ease;
    width: 0%;
}

.upload-progress-percentage {
    font-size: 11px;
    color: #5f6368;
    min-width: 35px;
    text-align: right;
}
```

```css
.upload-item-actions {
    display: flex;
    align-items: center;
    gap: 8px;
    margin-top: 4px;
}

.upload-action-btn {
    background: none;
    border: none;
    padding: 4px;
    cursor: pointer;
    border-radius: 4px;
    color: #5f6368;
    font-size: 12px;
    transition: background-color 0.2s;
}

.upload-action-btn:hover {
    background-color: #f1f3f4;
}

/* Estados de los archivos */
.upload-item.pending .upload-item-icon {
    color: #fbbc04;
}

.upload-item.uploading .upload-progress-fill {
    background-color: #1a73e8;
}

.upload-item.completed .upload-item-icon {
    color: #34a853;
}

.upload-item.completed .upload-progress-fill {
    background-color: #34a853;
}

.upload-item.error .upload-item-icon {
    color: #ea4335;
}

.upload-item.error .upload-progress-fill {
    background-color: #ea4335;
}
```

```css
.upload-item.error .upload-item-status {
    color: #ea4335;
}

/* Animaciones */
@keyframes slideInUp {
    from {
        transform: translateY(100%);
        opacity: 0;
    }
    to {
        transform: translateY(0);
        opacity: 1;
    }
}

.upload-queue-container.show {
    display: block;
    animation: slideInUp 0.3s ease-out;
}

/* Responsive */
@media (max-width: 768px) {
    .upload-queue-container {
        width: calc(100vw - 40px);
        right: 20px;
        left: 20px;
    }
}

/* Scrollbar personalizado */
.upload-queue-body::-webkit-scrollbar {
    width: 6px;
}

.upload-queue-body::-webkit-scrollbar-track {
    background: #f1f3f4;
}

.upload-queue-body::-webkit-scrollbar-thumb {
    background: #dadce0;
    border-radius: 3px;
}

.upload-queue-body::-webkit-scrollbar-thumb:hover {
    background: #bdc1c6;
}
```

Crear Nuevo Archivo: upload-queue.js

Ubicación: Crear en la carpeta de JavaScript de tu proyecto

Contenido completo del archivo:

JavaScript

```
/**
 * Sistema de Cola de Progreso para Subida de Archivos
 * Estilo Google Drive
 */

class UploadQueue {
    constructor() {
        this.queue = [];
        this.activeUploads = 0;
        this.maxConcurrentUploads = 3;
        this.container = null;
        this.init();
    }

    init() {
        this.createContainer();
        this.bindEvents();
    }

    createContainer() {
        // Crear el contenedor principal
        this.container = document.createElement('div');
        this.container.className = 'upload-queue-container';
        this.container.innerHTML = `
            <div class="upload-queue-header">
                <h3 class="upload-queue-title">Subiendo archivos</h3>
                <button class="upload-queue-close"
onclick="uploadQueue.hide()">
                    <i class="material-icons" style="font-size:
18px;">close</i>
                </button>
            </div>
            <div class="upload-queue-body" id="upload-queue-body">
            </div>
        `;
        document.body.appendChild(this.container);
    }

    bindEvents() {
        // Cerrar con Escape
```

```
        document.addEventListener('keydown', (e) => {
            if (e.key === 'Escape' &&
this.container.classList.contains('show')) {
                this.hide();
            }
        });
    }

    addFile(file, uploadFunction) {
        const fileId = this.generateId();
        const uploadItem = {
            id: fileId,
            file: file,
            name: file.name,
            size: file.size,
            status: 'pending',
            progress: 0,
            uploadFunction: uploadFunction,
            xhr: null
        };

        this.queue.push(uploadItem);
        this.renderItem(uploadItem);
        this.show();
        this.processQueue();

        return fileId;
    }

    renderItem(item) {
        const queueBody = document.getElementById('upload-queue-body');
        const itemElement = document.createElement('div');
        itemElement.className = `upload-item ${item.status}`;
        itemElement.id = `upload-item-${item.id}`;

        itemElement.innerHTML = `
            <div class="upload-item-header">
                <i class="material-icons upload-item-icon">description</i>
                <span class="upload-item-name"
title="${item.name}">${item.name}</span>
                <span class="upload-item-
status">${this.getStatusText(item.status)}</span>
            </div>
            <div class="upload-item-progress">
                <div class="upload-progress-bar">
                    <div class="upload-progress-fill" style="width:
${item.progress}%"></div>
                </div>
```

```
                <span class="upload-progress-percentage">${item.progress}%
</span>
            </div>
            <div class="upload-item-actions">
                ${item.status === 'uploading' ?
                    `<button class="upload-action-btn"
onclick="uploadQueue.cancelUpload('${item.id}')">Cancelar</button>` :
                    item.status === 'error' ?
                    `<button class="upload-action-btn"
onclick="uploadQueue.retryUpload('${item.id}')">Reintentar</button>` :
                    ''
                }
            </div>
        `;

        queueBody.appendChild(itemElement);
    }

    updateItem(itemId, updates) {
        const item = this.queue.find(q => q.id === itemId);
        if (!item) return;

        Object.assign(item, updates);

        const element = document.getElementById(`upload-item-${itemId}`);
        if (!element) return;

        // Actualizar clase de estado
        element.className = `upload-item ${item.status}`;

        // Actualizar estado
        const statusElement = element.querySelector('.upload-item-status');
        statusElement.textContent = this.getStatusText(item.status);

        // Actualizar progreso
        const progressFill = element.querySelector('.upload-progress-fill');
        const progressText = element.querySelector('.upload-progress-
percentage');
        progressFill.style.width = `${item.progress}%`;
        progressText.textContent = `${item.progress}%`;

        // Actualizar acciones
        const actionsContainer = element.querySelector('.upload-item-
actions');
        if (item.status === 'uploading') {
            actionsContainer.innerHTML = `<button class="upload-action-btn"
onclick="uploadQueue.cancelUpload('${item.id}')">Cancelar</button>`;
        } else if (item.status === 'error') {
```

```
            actionsContainer.innerHTML = `<button class="upload-action-btn"
onclick="uploadQueue.retryUpload('${item.id}')">Reintentar</button>`;
        } else {
            actionsContainer.innerHTML = '';
        }
    }

    processQueue() {
        const pendingItems = this.queue.filter(item => item.status ===
'pending');

        while (this.activeUploads < this.maxConcurrentUploads &&
pendingItems.length > 0) {
            const item = pendingItems.shift();
            this.startUpload(item);
        }
    }

    startUpload(item) {
        this.activeUploads++;
        this.updateItem(item.id, { status: 'uploading' });

        // Llamar a la función de subida personalizada
        item.uploadFunction(item, (progress) => {
            this.updateItem(item.id, { progress: Math.round(progress) });
        }, (success, error) => {
            this.activeUploads--;
            if (success) {
                this.updateItem(item.id, { status: 'completed', progress:
100 });
                setTimeout(() => this.removeItem(item.id), 3000);
            } else {
                this.updateItem(item.id, { status: 'error', progress: 0 });
            }
            this.processQueue();
        });
    }

    cancelUpload(itemId) {
        const item = this.queue.find(q => q.id === itemId);
        if (!item) return;

        if (item.xhr) {
            item.xhr.abort();
        }

        this.activeUploads--;
        this.removeItem(itemId);
```

```
        this.processQueue();
    }

    retryUpload(itemId) {
        const item = this.queue.find(q => q.id === itemId);
        if (!item) return;

        this.updateItem(itemId, { status: 'pending', progress: 0 });
        this.processQueue();
    }

    removeItem(itemId) {
        const element = document.getElementById(`upload-item-${itemId}`);
        if (element) {
            element.remove();
        }

        this.queue = this.queue.filter(q => q.id !== itemId);

        if (this.queue.length === 0) {
            setTimeout(() => this.hide(), 1000);
        }
    }

    show() {
        this.container.classList.add('show');
    }

    hide() {
        this.container.classList.remove('show');
    }

    getStatusText(status) {
        const statusTexts = {
            'pending': 'En cola',
            'uploading': 'Subiendo...',
            'completed': 'Completado',
            'error': 'Error'
        };
        return statusTexts[status] || status;
    }

    generateId() {
        return 'upload_' + Date.now() + '_' +
Math.random().toString(36).substr(2, 9);
    }
}
```

```javascript
// Inicializar la cola global
const uploadQueue = new UploadQueue();
```

Modificaciones Específicas en files.js

1. Incluir los Nuevos Archivos CSS y JS

Ubicación: En el archivo HTML principal donde se incluye files.js

Código a AGREGAR antes de la etiqueta de cierre `</head>`:

HTML

```html
<link rel="stylesheet" href="ruta/a/tu/css/upload-progress.css">
<script src="ruta/a/tu/js/upload-queue.js"></script>
```

2. Modificar el Event Listener de Subida de Archivos

Ubicación: Líneas 245-350 aproximadamente (función que maneja `file.addEventListener("change"`)

Código ACTUAL a REEMPLAZAR:

JavaScript

```javascript
// Maneja la subida de archivos individuales al seleccionar un archivo.
if (file) {
    file.addEventListener("change", async function (e) {
        const files = e.target.files;
        const fileList = document.querySelector("#file-list");
        const uploadSpinner = document.querySelector("#upload-spinner");

        if (files.length === 0) {
            alertaPerzonalizada("warning", "No se seleccionaron archivos");
            if (fileList) fileList.innerHTML = "";
            if (uploadSpinner) uploadSpinner.classList.add("d-none");
            return;
        }

        // ... resto del código actual
    });
}
```

Código NUEVO:

JavaScript

```
// Maneja la subida de archivos individuales al seleccionar un archivo.
if (file) {
    file.addEventListener("change", async function (e) {
        const files = e.target.files;
        const fileList = document.querySelector("#file-list");
        const uploadStatus = document.querySelector("#upload-status");

        if (files.length === 0) {
            alertaPerzonalizada("warning", "No se seleccionaron archivos");
            if (fileList) fileList.innerHTML = "";
            if (uploadStatus) uploadStatus.classList.add("d-none");
            return;
        }

        // Validaciones de tamaño y cantidad de archivos
        const totalSizeMB = Array.from(files).reduce((total, file) => total
+ file.size, 0) / (1024 * 1024);
        if (totalSizeMB > 50) {
            alertaPerzonalizada("warning", "Los archivos exceden el límite
de 50MB. Selecciona archivos más pequeños.");
            e.target.value = "";
            return;
        }

        // Mostrar archivos seleccionados
        if (fileList) {
            let html = "<ul class='list-group'>";
            for (let i = 0; i < files.length; i++) {
                html += `<li class='list-group-item'><i class='material-
icons me-2'>insert_drive_file</i>${files[i].name}</li>`;
            }
            html += "</ul>";
            fileList.innerHTML = html;
        }

        // Mostrar mensaje de estado
        if (uploadStatus) {
            uploadStatus.classList.remove("d-none");
        }

        // Obtener y renovar token si es necesario
        const expiresAt = localStorage.getItem("expires_at");
        const currentTime = Math.floor(Date.now() / 1000);
        let token = localStorage.getItem("token");

        if (token && expiresAt && currentTime > expiresAt - 300) {
            token = await renovarToken();
```

```
        }

        if (!token) {
            alertaPerzonalizada("error", "Sesión expirada. Por favor, inicia
sesión nuevamente.");
            if (uploadStatus) uploadStatus.classList.add("d-none");
            setTimeout(() => {
                window.location = base_url;
            }, 1500);
            return;
        }

        // Agregar archivos a la cola de progreso
        for (let i = 0; i < files.length; i++) {
            const file = files[i];
            uploadQueue.addFile(file, (item, onProgress, onComplete) => {
                uploadSingleFile(file, token, onProgress, onComplete);
            });
        }

        // Cerrar modal después de agregar archivos a la cola
        if (myModal) {
            myModal.hide();
        }

        // Limpiar formulario
        e.target.value = "";
        if (fileList) fileList.innerHTML = "";
        if (uploadStatus) uploadStatus.classList.add("d-none");
    });
}
```

3. Crear Nueva Función para Subida Individual

Ubicación: Agregar después de la función `renovarToken()` (aproximadamente línea 70)

Código a AGREGAR:

JavaScript

```
// Función para subir un archivo individual con seguimiento de progreso
function uploadSingleFile(file, token, onProgress, onComplete) {
    const data = new FormData();
    let carpetaValue = null;

    const id_carpeta = document.querySelector("#id_carpeta");
    if (id_carpeta && id_carpeta.value && id_carpeta.value.trim() !== "") {
        carpetaValue = id_carpeta.value.trim();
```

```
    } else {
        carpetaValue = "1";
    }

    data.append("id_carpeta", carpetaValue);
    data.append("files[]", file);

    const xhr = new XMLHttpRequest();
    const url = base_url + "admin/subirArchivo";

    // Configurar seguimiento de progreso
    xhr.upload.addEventListener("progress", function(e) {
        if (e.lengthComputable) {
            const percentComplete = (e.loaded / e.total) * 100;
            onProgress(percentComplete);
        }
    });

    xhr.addEventListener("load", function() {
        if (this.status === 200) {
            try {
                const res = JSON.parse(this.responseText);
                if (res.tipo === "success") {
                    onComplete(true);
                    actualizarEstadisticas();

                    // Recargar página después de que todos los archivos
terminen
                    setTimeout(() => {
                        if (id_carpeta && id_carpeta.value &&
id_carpeta.value.trim() !== "" && id_carpeta.value !== "1") {
                            window.location = base_url + "admin/ver/" +
id_carpeta.value;
                        } else {
                            window.location.reload();
                        }
                    }, 2000);
                } else {
                    onComplete(false, res.mensaje);
                    alertaPerzonalizada(res.tipo, res.mensaje);
                }
            } catch (e) {
                onComplete(false, "Respuesta inválida del servidor");
                alertaPerzonalizada("error", "Respuesta inválida del
servidor");
            }
        } else if (this.status === 401) {
            onComplete(false, "Sesión expirada");
```

```
            alertaPerzonalizada("error", "Sesión expirada. Inicia sesión
nuevamente.");
            setTimeout(() => {
                window.location = base_url;
            }, 1500);
        } else {
            onComplete(false, "Error en el servidor: " + this.status);
            alertaPerzonalizada("error", "Error en el servidor: " +
this.status);
        }
    });

    xhr.addEventListener("error", function() {
        onComplete(false, "Error de conexión");
        alertaPerzonalizada("error", "Error de conexión");
    });

    xhr.addEventListener("abort", function() {
        onComplete(false, "Subida cancelada");
    });

    xhr.open("POST", url, true);
    xhr.setRequestHeader("Authorization", "Bearer " + token);
    xhr.send(data);

    // Retornar el objeto xhr para poder cancelar la subida
    return xhr;
}
```

4. Modificar el Event Listener de Subida de Carpetas

Ubicación: Buscar la función que maneja `folder.addEventListener("change")` (aproximadamente líneas 400-500)

Código a MODIFICAR:
En la función de subida de carpetas, reemplazar la parte donde se muestra el spinner por:

JavaScript

```
// Reemplazar estas líneas:
if (uploadSpinner) {
    uploadSpinner.classList.remove("d-none");
    btnSubirCarpeta.disabled = true;
}

// Por estas líneas:
if (uploadStatus) {
```

```
    uploadStatus.classList.remove("d-none");
}

// Y al final de la función, reemplazar:
if (uploadSpinner) uploadSpinner.classList.add("d-none");
btnSubirCarpeta.disabled = false;

// Por:
if (uploadStatus) uploadStatus.classList.add("d-none");
```

5. Eliminar Referencias al Spinner Obsoleto

Buscar y ELIMINAR todas las referencias a:

- `uploadSpinner`
- `#upload-spinner`
- `.classList.add("d-none")` relacionadas con el spinner
- `.classList.remove("d-none")` relacionadas con el spinner

Ubicaciones aproximadas:

- Línea 250: `const uploadSpinner = document.querySelector("#upload-spinner");`
- Línea 255: `if (uploadSpinner) uploadSpinner.classList.add("d-none");`
- Línea 270: `if (uploadSpinner) uploadSpinner.classList.add("d-none");`
- Línea 285: `uploadSpinner.classList.remove("d-none");`
- Línea 340: `if (uploadSpinner) uploadSpinner.classList.add("d-none");`
- Y otras referencias similares en la función de subida de carpetas

Modificaciones en el Archivo HTML Principal

1. Incluir los Nuevos Archivos

Ubicación: En el `<head>` de tu archivo HTML principal (probablemente header.php)

Código a AGREGAR:

HTML

```
<!-- Estilos para cola de progreso -->
<link rel="stylesheet" href="<?php echo BASE_URL; ?>Assets/css/upload-
progress.css">
```

```
<!-- Script para cola de progreso (agregar antes de files.js) -->
<script src="<?php echo BASE_URL; ?>Assets/js/upload-queue.js"></script>
```

Archivos Adicionales Necesarios

1. Crear Directorio de Estilos

Si no existe, crear la carpeta: `Assets/css/`

2. Crear Directorio de Scripts

Si no existe, crear la carpeta: `Assets/js/`

3. Ubicación de Archivos

- `upload-progress.css` → `Assets/css/upload-progress.css`
- `upload-queue.js` → `Assets/js/upload-queue.js`

Funcionalidades Implementadas

Características de la Cola de Progreso

1. **Posicionamiento Fijo:** La cola aparece en la esquina inferior derecha de la pantalla
2. **Múltiples Archivos:** Soporte para subir varios archivos simultáneamente
3. **Progreso Individual:** Cada archivo muestra su propio progreso y estado
4. **Estados Visuales:** Pendiente (amarillo), Subiendo (azul), Completado (verde), Error (rojo)
5. **Acciones por Archivo:** Cancelar subidas en progreso, reintentar archivos con error
6. **Responsive:** Se adapta a pantallas móviles
7. **Animaciones:** Transiciones suaves y animación de entrada
8. **Auto-ocultación:** Se oculta automáticamente cuando todas las subidas terminan

Controles de Usuario

1. **Cerrar Cola:** Botón X en la esquina superior derecha
2. **Cancelar Subida:** Botón "Cancelar" para archivos en progreso
3. **Reintentar:** Botón "Reintentar" para archivos con error
4. **Tecla Escape:** Cierra la cola de progreso

Pruebas y Validación

1. Pruebas Básicas

- Subir un archivo individual
- Subir múltiples archivos
- Verificar que aparezca la cola de progreso
- Verificar que se actualice el progreso correctamente

2. Pruebas de Funcionalidad

- Cancelar una subida en progreso
- Reintentar un archivo con error
- Cerrar la cola manualmente
- Verificar auto-ocultación

3. Pruebas de Responsive

- Probar en dispositivos móviles
- Verificar que la cola se adapte al ancho de pantalla

Consideraciones Técnicas

Compatibilidad

- Compatible con navegadores modernos (Chrome, Firefox, Safari, Edge)
- Requiere soporte para XMLHttpRequest Level 2 (progress events)
- Funciona con Bootstrap 5 (utiliza clases existentes)

Rendimiento

- Máximo 3 subidas simultáneas (configurable)
- Gestión eficiente de memoria
- Limpieza automática de elementos DOM

Seguridad

- Mantiene el sistema de autenticación JWT existente

- Validaciones de tamaño y tipo de archivo
- Manejo seguro de errores

Esta implementación transforma completamente la experiencia de subida de archivos, proporcionando una interfaz moderna y profesional similar a Google Drive, mientras mantiene toda la funcionalidad y seguridad del sistema existente.