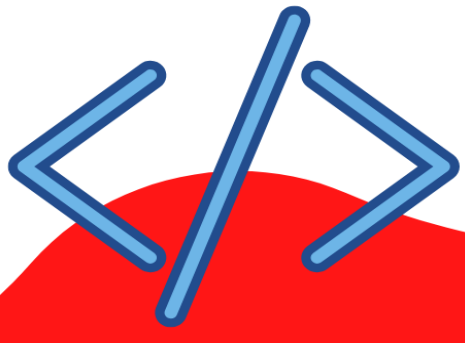
UES-FMP

CURSO DE ESPECIALIZACIÓN MÓDULO FRONT-END

MATERIAL N. 15 DOWNLOAD, READ EXCEL Y TAREAS

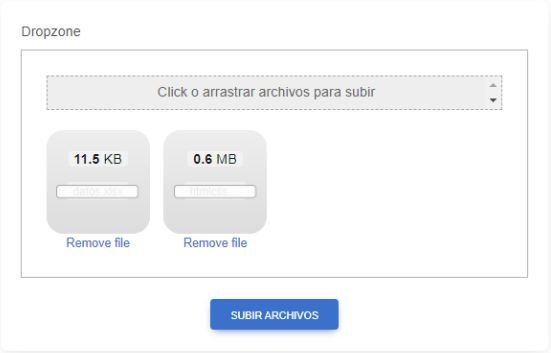


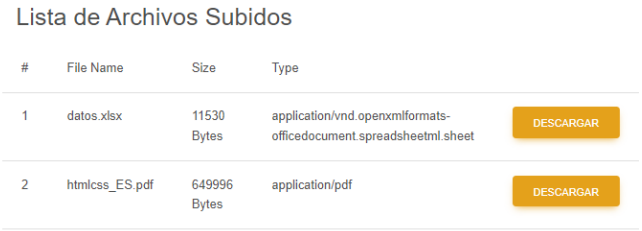
ING. YANCY ELIZABETH MARTÍNEZ DE MOLINA

Objetivos:

- Implementar la descarga de archivo.
- Leer datos de Excel
- Drag end drop

Carga y descarga de archivos





- Instalar **ngx-dropzone-wrapper**


```
npm install ngx-dropzone-wrapper --save
```

Opciones de configuraciòn

Opciones disponibles para Dropzone.

Name	Default	Description
`url`	`null`	Debe especificarse en elementos distintos del formulario (o cuando el formulario no tiene un atributo de acción). También puede proporcionar una función que se llamará con archivos y debe devolver la URL
`method`	`"post"`	Se puede cambiar a "put" si es necesario. También puede proporcionar una función que se llamará con archivos y debe devolver el método
`withCredentials`	`false`	Se establecerá en XHRequest
`timeout`	`null`	El tiempo de espera para las solicitudes XHR en milisegundos Si se establece en nulo o 0, no se establecerá ningún tiempo de espera.
`parallelUploads`	`2`	Cuántas cargas de archivos procesar en paralelo (Consulte la sección de documentación Cargas de archivos en cola para obtener más información)
`uploadMultiple`	`false`	Ya sea para enviar varios archivos en una solicitud. Si esto se establece en verdadero, entonces el elemento de entrada del archivo alternativo también tendrá el atributo múltiple. Esta opción también desencadenará eventos

		adicionales (como procesamiento múltiple). Consulte la sección de documentación de eventos para obtener más información.
chunking	false	Si desea que los archivos se carguen en fragmentos en su servidor. Esto no se puede usar en combinación con uploadMultiple. Ver [chunksUploaded](#config-chunksUploaded) para la devolución de llamada para finalizar una carga.
forceChunking	false	Si la fragmentación está habilitada, esto define si **todos** los archivos deben fragmentarse, incluso si el tamaño del archivo es inferior a chunkSize. Esto significa que se enviarán los datos adicionales del formulario de fragmentos y se invocará la devolución de llamada de chunksUploaded.
chunkSize	2000000	Si la fragmentación es verdadera, esto define el tamaño de la fracción en bytes.
parallelChunkUploads	false	Si es verdadero, los fragmentos individuales de un archivo se cargan simultáneamente.
retryChunks	false	Si se debe volver a intentar un fragmento si falla.
retryChunksLimit	3	Si retryChunks es verdadero, ¿cuántas veces se debe volver a intentar?
maxFilesize	256	El tamaño de archivo máximo (en bytes) que se permite cargar.
paramName	"file"	El nombre del parámetro de archivo que se transfiere. **NOTA**: si tiene la opción uploadMultiple establecida en verdadero, Dropzone agregará [] al nombre.
createImageThumbnails	true	Si se deben generar miniaturas para las imágenes
maxThumbnailFilesize	10	In MB. Cuando el nombre de archivo supera este límite, no se generará la miniatura.
thumbnailWidth	120	Si null, la proporción de la imagen se utilizará para calcularla.
thumbnailHeight	120	Lo mismo que thumbnailWidth. Si ambos son nulos, no se cambiará el tamaño de las imágenes.
thumbnailMethod	"crop"	Cómo se deben reducir las imágenes en caso de que se proporcionen thumbnailWidth y thumbnailHeight. Puede ser contener o recortar.
resizeWidth	null	Si se configura, las imágenes cambiarán de tamaño a estas dimensiones antes de **cargarse**. Si solo se proporciona uno, resizeWidth **o** resizeHeight, se conservará la relación de aspecto original del archivo. La función options.transformFile usa estas opciones, por lo que si se anula la función transformFile, estas opciones no hacen nada.
resizeHeight	null	See resizeWidth.
resizeMimeType	null	El tipo mimo de la imagen redimensionada (antes de que se cargue en el servidor). Si es nulo, se usará el tipo mime original. Para forzar jpeg, por ejemplo, use image/jpeg. Consulte resizeWidth para obtener más información.
resizeQuality	0.8	La calidad de las imágenes redimensionadas. Ver resizeWidth.
resizeMethod	"contain"	Cómo se deben reducir las imágenes en caso de que se proporcionen resizeWidth y resizeHeight. Puede ser contener o recortar.
filesizeBase	1000	La base que se utiliza para calcular el tamaño de archivo **mostrado**. Puede cambiar esto a 1024 si prefiere mostrar kibibytes, mebibytes, etc... 1024 es técnicamente incorrecto, porque 1024 bytes son 1 kibibyte y no 1 kilobyte. Puede cambiar esto a 1024 si no le importa la validez.

maxFiles	null	Si no es nulo, define cuántos archivos maneja este Dropzone. Si excede, se llamará al evento maxfileexceeded. El elemento dropzone obtiene la clase dz-max-files-reached en consecuencia para que pueda proporcionar comentarios visuales.
headers	null	Un objeto opcional para enviar encabezados adicionales al servidor. Por ejemplo: { "Mi-Awesome-Header": "valor de encabezado" }
clickable	true	Si es verdadero, se podrá hacer clic en el elemento de la zona de caída, si es falso, no se podrá hacer clic en nada. También puede pasar un elemento HTML, un selector CSS (para varios elementos) o una matriz de estos. En ese caso, todos esos elementos activarán una carga cuando se haga clic en ellos.
ignoreHiddenFiles	true	Si se deben ignorar los archivos ocultos en los directorios.
acceptedFiles	null	La implementación predeterminada de accept verifica el tipo mime o la extensión del archivo con esta lista. Esta es una lista separada por comas de tipos mime o extensiones de archivo. Ej.: imagen/*,aplicación/pdf,.psd Si se puede hacer clic en Dropzone, esta opción también se usará como parámetro [aceptar] (https://developer.mozilla.org/en-US/docs/HTML/Element/input#attr-accept) en la entrada del archivo oculto.
acceptedMimeTypes	null	**¡Obsoleto!** Utilice acceptFiles en su lugar.
autoProcessQueue	true	Si es falso, los archivos se agregarán a la cola, pero la cola no se procesará automáticamente. Esto puede ser útil si necesita alguna entrada adicional del usuario antes de enviar archivos (o si desea que todos los archivos se envíen a la vez). Si está listo para enviar el archivo, simplemente llame a myDropzone.processQueue(). Consulte la sección de documentación [cargas de archivos en cola](#cargas de archivos en cola) para obtener más información.
autoQueue	true	Si es falso, los archivos agregados a la zona de colocación no se pondrán en cola de forma predeterminada. Tendrá que llamar a enqueueFile(archivo) manualmente.
addRemoveLinks	false	Si es verdadero, esto agregará un enlace a cada vista previa de archivo para eliminar o cancelar (si ya se está cargando) el archivo. Las opciones dictCancelUpload, dictCancelUploadConfirmation y dictRemoveFile se utilizan para la redacción.
previewsContainer	null	Define dónde mostrar las vistas previas del archivo; si es nulo, se usa el elemento Dropzone. Puede ser un HTMLElement simple o un selector de CSS. El elemento debe tener la clase dropzone-previews para que las vistas previas se muestren correctamente.
disablePreviews	false	Establézcalo en verdadero si no desea que se muestren vistas previas.
hiddenInputContainer	"body"	Este es el elemento al que se adjuntará el campo de entrada oculto (que se utiliza al hacer clic en la zona de colocación para activar la selección de archivos). Esto puede ser importante en caso de que utilice marcos para cambiar el contenido de su página. Puede ser una cadena selectora o un elemento directamente.
capture	null	Si es nulo, no se especificará ningún tipo de captura Si es una cámara, los dispositivos móviles omitirán la selección de archivos y elegirán la cámara Si es un micrófono, los dispositivos móviles omitirán la selección de archivos y elegirán el micrófono Si es una videocámara, los dispositivos móviles omitirán la selección de archivos y elegirán la cámara en modo video En los dispositivos

		Apple, multiple debe establecerse en falso. Es posible que los archivos aceptados deban establecerse en un tipo mimo apropiado (por ejemplo, "imagen/*", "audio/*" o "video/*").
`renameFilename`	`null`	**Obsoleto**. Utilice renameFile en su lugar.
`renameFile`	`null`	Una función que se invoca antes de que el archivo se cargue en el servidor y cambia el nombre del archivo. Esta función obtiene el archivo como argumento y puede usar file.name. Se puede acceder al nombre real del archivo que se usa durante la carga a través de file.upload.filename
`forceFallback`	`false`	Si es verdadero, se forzará el retroceso. Esto es muy útil para probar primero las implementaciones de su servidor y asegurarse de que todo funcione como se esperaba sin dropzone si experimenta problemas, y para probar cómo se verán sus respaldos.

- En el módulo

```
import { DROPZONE_CONFIG, DropzoneConfigInterface, DropzoneModule } from 'ngx-dropzone-wrapper';
```

```
const config: DropzoneConfigInterface = {
  url:'http://'
};
```

```
@NgModule({
  declarations: [...
  ],
  imports: [
    CommonModule,
    ExcelRoutingModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule,
    DropzoneModule,
    FullCalendarModule,
    SharedModule,
    ModalModule.forRoot(),
    DragDropModule,
    DisabledDirective,
  ],
  providers: [
    {
      provide: DROPZONE_CONFIG,
      useValue: config,
    },
  ]
```

- En el componente

```
<div class="row">
  <div class="col-6">
    <div class="card">
      <div class="card-body">
        <div class="card-title">Dropzone</div>
        <div>
          <!--  -->
          <dropzone
            id="dropzone"
            class="dropzone"
            [config]="config"
            (removedFile)="eliminar($event)"
            [message]="'Click o arrastrar archivos para subir'">
          </dropzone>
        </div>
        <div class="text-center mt-4">
          <button type="button" class="btn btn-primary" (click)="multiple()">Subir
            Archivos</button>
        </div>
      </div>

    </div>
  </div>
  <div class="file-table col-6">…
  </div>
</div>

</div>
```

```
</div>
<div class="file-table col-6">
  <h3 class="m-3">Lista de Archivos Subidos</h3>
  <table class="table">
    <thead>
      <tr>
        <th scope="col">#</th>
        <th scope="col">File Name</th>
        <th scope="col">Size</th>
        <th scope="col">Type</th>
        <td> </td>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let file of allFiles; let i = index">
        <th scope="row">{{i+1}}</th>
        <td>{{file.archivo.name}}</td>
        <td>{{file.archivo.size}} Bytes</td>
        <td>{{file.archivo.type}}</td>
        <td>
          <button type="button" class="btn btn-warning"
            (click)="descargar(file.id ,file.archivo.name)">Descargar</button>
        </td>
      </tr>
      <tr class="text-center" *ngIf="allFiles.length === 0">
        <td colspan="4"><strong>No hay archivos para Descargar</strong></td>
      </tr>
    </tbody>
  </table>
</div>
```

```
export class UsoDopzoneComponent  {


  myFiles: File[] = [];
  allFiles: IArchivoSubido[] = [];

  // configuración de dropzone
  config: DropzoneConfigInterface = {
    maxFilesize: 500,
    addRemoveLinks: true,
    uploadMultiple: true,
    accept: (file: File) => {
      this.myFiles.push(file);
    },
  };

  constructor(private uploadService: UploadService) { }
```

```
  eliminar(file: File) {
    const result = this.myFiles.filter(f => f != file);
    this.myFiles = result;
    const result1 = this.allFiles.filter(f => f.archivo != file);
    this.allFiles = result1;
  }
```

```
descargar(id: string, name: string) {
  this.uploadService.download1(id).subscribe(resp => {
    this.administradorDescarga(name, resp);
  });
}
```

```
administradorDescarga(name: string, resp: File) {
  const dataType = resp.type;
  const dataBinary = [];
  dataBinary.push(resp);

  const filePath = window.URL.createObjectURL(new Blob(dataBinary, { type: dataType }));
  const downloadLink = document.createElement('a');
  downloadLink.href = filePath;
  downloadLink.setAttribute('download', name);
  document.body.appendChild(downloadLink);
  downloadLink.click();
}
```

- Interfaz a utilizar

```
export interface IArchivoSubido {
  id: string,
  archivo: File;
}
```

- En el servicio

```
export class UploadService {

  url = 'http://localhost:8080/file'; ///upload

  constructor(private http: HttpClient) { }
```

```
multiple(myFiles: File[]) {
  const formData = new FormData();
  myFiles.forEach(archivo => {
    formData.append('file', archivo);
  });
  return this.http.post<IArchivo[]>(`${this.url}/multiple`, formData).pipe(
    catchError((err: HttpErrorResponse) => {
      return this.errorHandler(err)
    })
  );
}
```

Capturando errores de la peticion http

```
errorHandler(error: HttpErrorResponse): Observable<never> {
  if (error.status == 500)
    return throwError(() => new Error(error.statusText))
  if (error.status == 400)
    return throwError(() => new Error('Algo pasó'));
  return throwError(() => new Error('Error al subir rchivos'));
}
```

```
download1(id:string):Observable<File> {
  const headers=new HttpHeaders().set('Content-Type','Aplication/json');
  return   this.http.get<File>(`${id}`,
    {
      headers,
      responseType: 'blob' as 'json'
    })
}
```

LEER DATOS DE EXCEL

Leer archivos Excell

Seleccionar archivo | datos.xlsx

N	Apellido	Nombre	Email	Tiempo
1	Constanza Rivas	Josué David	cr17029@ues.edu.sv	15 minutos 1 segundos
2	Cortez Carmona	Josué Daniel	cc17081@ues.edu.sv	15 minutos 50 segundos
3	Martinez Melendez	Luis Eduardo	mm17113@ues.edu.sv	13 minutos 50 segundos
4	Del Cid Mejia	William Antonio	dm18019@ues.edu.sv	5 minutos 26 segundos
5	Rodríguez Perdomo	Kely Mishel	rp18027@ues.edu.sv	5 minutos 54 segundos
6	Portillo Pérez	Jonathan Eulises	pp14016@ues.edu.sv	12 minutos 8 segundos

- Instalar la librería de Excel

  npm install xlsx --save

```
  "ngx-infinite-scroll": "^14.0.0",
  "ngx-toastr": "15.2",
  "rxjs": "~7.5.0",
  "sweetalert2": "^11.7.3",
  "tslib": "^2.3.0",
  "xlsx": "^0.18.5",
  "zone.js": "~0.11.4"
},
```

- En el componente

```
export class ReadExcelComponent {
  datos!: IDatos[];


  constructor() { }
  leerFile(event: Event) {
    const target = event.target as HTMLInputElement;
    const file = (target.files as FileList)[0];

    let filereader = new FileReader();
    filereader.readAsArrayBuffer(file);
    filereader.onload = () => {
      let data = filereader.result;
      let Workbook = xls.read(data, { type: 'array' });

      const nameSheet = Workbook.SheetNames[0];
      const hoja = Workbook.Sheets[nameSheet];
      this.datos = xls.utils.sheet_to_json(hoja, { raw: true });
      console.log(this.datos);
    }
  }

}
```

```
import * as xls from 'xlsx';
```

- La interfaz

```
export interface IDatos {
  id: string;
  Apellido:string,
  Nombre:string,
  Correo:string,
  Tiempo:string
}
```

```
Go to component
<div class="container mt-5">
  <div class="row">
    <div class="col-6">
      <div class="card">
        <div class="card-body">
          <div class="card-title">Leer archivos Excell</div>
          <div>
            <!--  -->
            <input type="file" class="form-control"
              (change)="leerFile($event)">
          </div>
        </div>
      </div>
    </div>
  </div>
```

```
▼ Array(30) i
  ▶ 0: {id: 1, Apellido: 'Constanza Rivas', Nombre: 'Josué David', Correo: 'cr17029@ues.edu.sv', Tiempo: '15 minutos 1 segundos', …}
  ▶ 1: {id: 2, Apellido: 'Cortez Carmona', Nombre: 'Josué Daniel', Correo: 'cc17081@ues.edu.sv', Tiempo: '15 minutos 50 segundos', …}
  ▶ 2: {id: 3, Apellido: 'Martinez Melendez', Nombre: 'Luis Eduardo', Correo: 'mm17113@ues.edu.sv', Tiempo: '13 minutos 50 segundos', …}
  ▶ 3: {id: 4, Apellido: 'Del Cid Mejia', Nombre: 'William Antonio', Correo: 'dm18019@ues.edu.sv', Tiempo: '5 minutos 26 segundos', …}
  ▶ 4: {id: 5, Apellido: 'Rodríguez Perdomo', Nombre: 'Kely Mishel', Correo: 'rp18027@ues.edu.sv', Tiempo: '5 minutos 54 segundos', …}
  ▶ 5: {id: 6, Apellido: 'Portillo Pérez', Nombre: 'Jonathan Eulises', Correo: 'pp14016@ues.edu.sv', Tiempo: '12 minutos 8 segundos', …}
  ▶ 6: {id: 7, Apellido: 'Palacios Pineda', Nombre: 'Brayan Josã0', Correo: 'pp16014@ues.edu.sv', Tiempo: '13 minutos 32 segundos', …}
  ▶ 7: {id: 8, Apellido: 'Portillo López', Nombre: 'Karen Beatriz', Correo: 'pl15012@ues.edu.sv', Tiempo: '11 minutos 21 segundos', …}
  ▶ 8: {id: 9, Apellido: 'Najarro Sánchez', Nombre: 'Luis Jacinto', Correo: 'ns15005@ues.edu.sv', Tiempo: '5 minutos 46 segundos', …}
  ▶ 9: {id: 10, Apellido: 'Cruz Paz', Nombre: 'Marvin Neftali', Correo: 'cp17019@ues.edu.sv', Tiempo: '9 minutos 53 segundos', …}
```

```
  <div class="row">
    <table class="table" *ngIf="datos.length>0">
      <thead>
        <tr>
          <th>N</th>
          <th>Apellido</th>
          <th>Nombre</th>
          <th>Email</th>
          <th>Tiempo</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let x of datos">
          <td>{{x.id}} </td>
          <td>{{x.Apellido}} </td>
          <td>{{x.Nombre}} </td>
          <td>{{x.Correo}} </td>
          <td> <span>{{x.Tiempo}}</span>
          </td>
        </tr>
      </tbody>

    </table>
  </div>
</div>
```

Actividad evaluada en clases

- Armacenar datos en un archivo de Excel con la estructura de una de las tablas del backen de la Clinica.
- Leer los datos del archivo de Excel y registrar la información en la tabla del backend correspondiente.
- Realizar la petición get al endpoint de la tabla del backend y mostrar la informacón en el frontend.
- Le agregan la paginación.

Drag And Drop

Drag and drop es una funcionalidad en Angular que permite mover elementos de una zona a otra «arrastrando» y «soltando».

Para ello, necesitamos instalar la dependencia **@angular/cdk**
En cada bloque que queramos usar el drag and drop, tenemos que poner la directiva **cdkDropList**, además del id con **cdkDropList** de nuevo.
Con **cdkDropListData** indicamos que datos vamos a agarrar, del primer array.
Con **cdkDropListConnectedTo** le indicamos con que otro bloque esta relacionado, es decir, indicando su id.
Se usar la directiva **cdkDrag** en el elemento que vamos a arrastrar.
También, se indica el evento que se realizará al soltar el elemento.

```
(cdkDropListDropped)="drop($event)">
```

Mas documentación

https://material.angular.io/cdk/drag-drop/api

- Instalar angular cdk, para poder utilizar drag and drop

  npm i @angular/cdk@14.0.0 –force

- En el módulo

```
@NgModule({
  declarations: [ ···
  ],
  imports: [
    CommonModule,
    ExcelRoutingModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule,
    DropzoneModule,
    FullCalendarModule,
    SharedModule,
    ModalModule.forRoot(),
    DragDropModule,
    DisabledDirective,
  ],
```

```
import {DragDropModule} from '@angular/cdk/drag-drop';
```

- La Interfaz

```
export interface ITarea {
  tarea: string;
  descripcion:string,
  fechaInicio:string,
  fechaFin:string,
  participantes:Array<string>,
  terminado:boolean
}
```

- En el componente

```
<div class="container mt-5">
  <div class="row">
    <h1 class="text-center"> Tareas </h1>
    <div class="col-lg-4 p-3">
      <button type="button" class="btn btn-primary"
        (click)="openModal(template)"> + Tarea</button>
    </div>
  </div>
  <div class="row">
    <div class="col-lg-4 p-3">···
    </div>
    <div class="col-lg-4 p-3">···
    </div>
    <div class="col-lg-4 p-3">···
    </div>
  </div>
</div>

<ng-template #template>···
</ng-template>
```

```
<div class="row">
  <div class="col-lg-4 p-3">
    <h4>Pendientes</h4>
    <div cdkDropList class="tarea-list" [cdkDropListData]="porHacer"
      [cdkDropListConnectedTo]="['lista-progreso']"
      (cdkDropListDropped)="drop($event)" id="lista-pendiente">
      <div class="row tarea-box" *ngFor="let t of porHacer; index as phi"
        cdkDrag>
        <!-- inicio acordión -->
        <div class="accordion" id="accordionExample">···
        </div>
        <!-- fin acordión -->

      </div>
    </div>
  </div>
  <div class="col-lg-4 p-3">···
  </div>
  <div class="col-lg-4 p-3">···
  </div>
</div>
</div>

<ng-template #template>···
</ng-template>
```

```
<!-- inicio acordión -->
<div class="accordion" id="accordionExample">
  <div class="accordion-item">
    <h2 class="accordion-header" id="headingOne">
      <button class="accordion-button collapsed" type="button"
        data-bs-toggle="collapse" data-bs-target="#phi"
        aria-expanded="false" aria-controls="collapseOne"
        fdprocessedid="vje0vc">
        {{t.tarea}}
      </button>

    </h2>
    <div id="phi" class="accordion-collapse collapse"…
    </div>
    <div class=" form-group row">
      <div class="form-group col-md-1">
        <i class="bi bi-trash-fill " style="color: brown;"
          (click)="quitarTarea(phi)"></i>
      </div>

    </div>

  </div>
</div>
<!-- fin acordión -->
```

```
    </h2>
    <div id="phi" class="accordion-collapse collapse"
      aria-labelledby="headingOne" data-bs-parent="#accordionExample"
      style>
      <div class="accordion-body">
        <div class="form-group form-floating ">
          <small>{{t.descripcion}}</small>
        </div>
        <div class="form-group row">
          <div class="form-group col-md-6">
            <small>Inicio</small>
            <p> {{t.fechaInicio}}</p>
          </div>
          <div class="form-group col-md-6">
            <small class="p">fin</small>
            <p> {{t.fechaFin}}</p>
          </div>
        </div>
        <div class="row"
          *ngFor="let participante of t.participantes;index as ix">
          <small>{{participante}}</small>
        </div>

      </div>
    </div>
```

```
<div class="col-lg-4 p-3">
  <h4>En Curso</h4>
  <div cdkDropList class="tarea-list" [cdkDropListData]="progreso"
    [cdkDropListConnectedTo]="['lista-pendiente','lista-terminadas']"
    (cdkDropListDropped)="drop($event)" id="lista-progreso">
    <div class="row tarea-box" *ngFor="let t of progreso;index as pri"
      cdkDrag>
      <!-- inicio acordión -->
      <div class="accordion" id="accordionExample">…
      </div>
      <!-- fin acordión -->
    </div>
  </div>
</div>
<div class="col-lg-4 p-3">…
```

```
<!-- inicio acordión -->
<div class="accordion" id="accordionExample">
  <div class="accordion-item">
    <h2 class="accordion-header" id="headingOne">
      <button class="accordion-button collapsed" type="button"
        data-bs-toggle="collapse" data-bs-target="#pri"
        aria-expanded="false" aria-controls="collapseOne"
        fdprocessedid="vje0vc">
        {{t.tarea}}
      </button>
    </h2>
    <div id="pri" class="accordion-collapse collapse"
      aria-labelledby="headingOne" data-bs-parent="#accordionExample"
      style>
      <div class="accordion-body">
        <div class="form-group form-floating ">
          <small>{{t.descripcion}}</small>
        </div>
        <div class="form-group row">
          <div class="form-group col-md-6">
            <small>Inicio</small>
            <p> {{t.fechaInicio}}</p>
          </div>
          <div class="form-group col-md-6">
            <small class="p">Fin</small>
            <p> {{t.fechaFin}}</p>
          </div>
        </div>
        <div class="row"
          *ngFor="let participante of t.participantes;index as ix">
          <small>{{participante}}</small>
        </div>
      </div>
    </div>
  </div>
</div>
<!-- fin acordión -->
```

```
<div class="col-lg-4 p-3">
  <h4>Finalizadas</h4>
  <div cdkDropList class="tarea-list" [cdkDropListData]="terminada"
    (cdkDropListDropped)="drop($event)" id="lista-terminadas">
    <div class=" row tarea-box" *ngFor="let t of terminada;index as tei"
      cdkDrag>
      <!-- inicio acordión -->
      <div class="accordion" id="accordionExample">…
      </div>
      <!-- fin acordión -->
    </div>
  </div>
</div>
```

3 3.1

```
<!-- inicio acordión -->
<div class="accordion" id="accordionExample">
  <div class="accordion-item">
    <h2 class="accordion-header" id="headingOne">
      <button class="accordion-button collapsed" type="button"
        data-bs-toggle="collapse" data-bs-target="#tei"
        aria-expanded="false" aria-controls="collapseOne"
        fdprocessedid="vje0vc">
        {{t.tarea}}
      </button>
    </h2>
    <div id="tei" class="accordion-collapse collapse"…
    </div>
    <div class=" form-group row">
      <div class="form-group col-md-1">
        <i class="bi bi-check-circle" style="color: blue;"></i>
      </div>
    </div>
  </div>
</div>
<!-- fin acordión -->
```

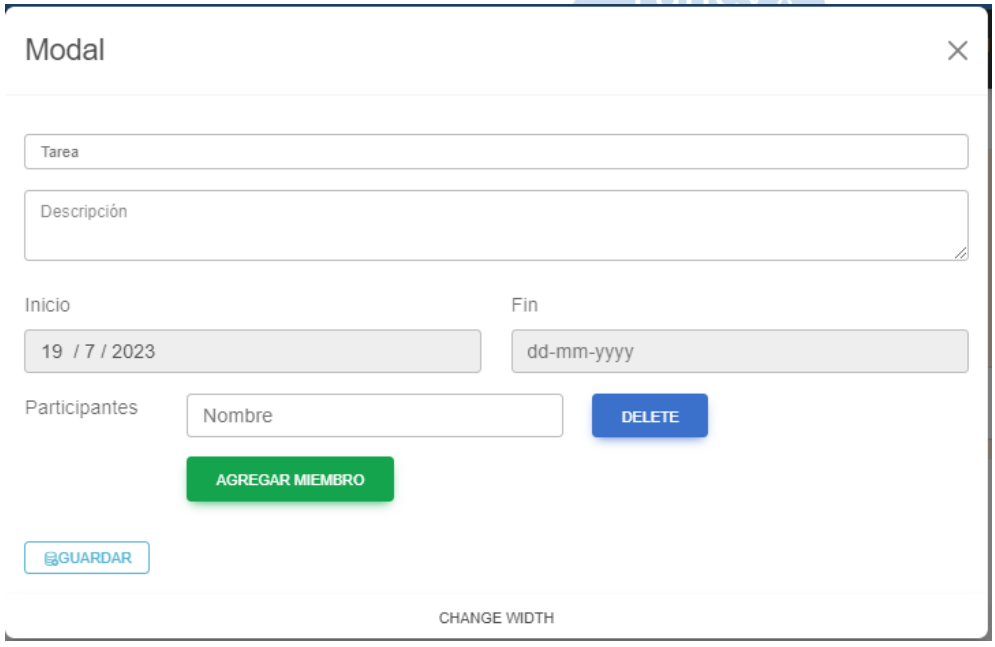
3.1 3.1.1

```
</h2>
<div id="tei" class="accordion-collapse collapse"
  aria-labelledby="headingOne" data-bs-parent="#accordionExample"
  style>
  <div class="accordion-body">
    <div class="form-group form-floating ">
      <small>{{t.descripcion}}</small>
    </div>
    <div class="form-group row">
      <div class="form-group col-md-6">
        <small>Inicio</small>
        <p> {{t.fechaInicio}}</p>
      </div>
      <div class="form-group col-md-6">
        <small class="p">Fin</small>
        <p> {{t.fechaFin}}</p>
      </div>
    </div>
    <div class="row"
      *ngFor="let participante of t.participantes; index as ix">
      <small>{{participante}}</small>
    </div>
```

3.1.1

3.1.1

El modal



```
<ng-template #template>
  <div class="modal-header">
    <h4 class="modal-title pull-left">Modal</h4>
    <button type="button" class="close btn-close pull-right" aria-label="Close"
      (click)="modalRef?.hide()">
      <span aria-hidden="true" class="visually-hidden">&times;</span>
    </button>
  </div>
  <div class="modal-body">…
  </div>
  <button type="button" class="btn" (click)="setModalClass()">Change width</button>
</ng-template>
```

```
<div class="modal-body">
  <form class="mt-3"
    [formGroup]="formTareas"
    (ngSubmit)="guardar()">
    <div class="form-group row">
      <div class="form-group col-md-12">
        <input
          type="text"
          [disabled]="false"
          class="form-control form-control-sm"
          placeholder="Tarea"
          formControlName="tarea" />
      </div>
    </div>
    <div class="form-group form-floating mt-3">
      <textarea
        row="3"
        class="form-control"
        formControlName="descripcion"></textarea>
      <label class="floatingInput">Descripción</label>
    </div>
    <div class="row">…
    </div>
    <div class="row mt-2">…
    </div>
    <button type="submit" class="btn btn-sm btn-outline-info mt-sm-2">
      <i class="bi bi-database-add"></i>Guardar
    </button>
  </form>
</div>
<button type="button" class="btn" (click)="setModalClass()">Change width</button>
</ng-template>
```

```
<div class="row">
  <div class="form-group col-md-6 mt-3">
    <label class="col-form-label col-lg-2">Inicio</label>
    <input
      type="text"
      class="form-control"
      placeholder="dd-mm-yyyy "
      formControlName="fechaInicio" />
  </div>
  <div class="form-group col-md-6 mt-3">
    <label class="col-form-label col-lg-2">Fin</label>
    <input
      type="text"
      class="form-control"
      placeholder="dd-mm-yyyy "
      formControlName="fechaFin" />
  </div>
</div>
<div class="row mt-2">···
</div>
<button type="submit" class="btn btn-sm btn-outline-info mt-sm-2">
  <i class="bi bi-database-add"></i>Guardar
</button>
```

```
</div>
<div class="row mt-2">
  <div class="inner-repeater mb-4">
    <div class="inner mb-0 row" formArrayName="participantes">
      <label class="col-form-label col-lg-2">Participantes</label>
      <div class="inner col-lg-10 ms-md-auto mt-2"
        *ngFor="let data of member.controls; index as i;">
        <div class="mb-3 row align-items-center">
          <div class="col-md-6">
            <input type="text" [formControlName]="i"
              class="inner form-control"
              placeholder="Nombre" />
          </div>
          <div class="col-md-2">
            <div class="mt-2 mt-md-0 d-grid">
              <input type="button" class="btn btn-primary inner"
                value="Delete"
                (click)="deleteMember(i)" />
            </div>
          </div>
        </div>
      </div>
    </div>

    <div class="row justify-content-end">
      <div class="col-lg-10">
        <input data-repeater-create type="button"
          class="btn btn-success inner" value="Agregar Miembro"
          (click)="addMember()" />
      </div>
    </div>
  </div>
</div>
<button type="submit" class="btn btn-sm btn-outline-info mt-sm-2">
  <i class="bi bi-database-add"></i>Guardar
</button>
```

2

- En el Ts

```
export class TareasComponent implements OnInit {

  formTareas!: FormGroup;

  porHacer: ITarea[] = [];
  progreso: ITarea[] = [];
  terminada: ITarea[] = [];

  //*para el modal
  modalRef?: BsModalRef;
  valueWidth = false;
  constructor(private fb: FormBuilder, private modalService: BsModalService) {
  }

  ngOnInit(): void {
    this.initForm();
  }
```

- El formulario

```
  initForm() {
    this.formTareas = this.fb.group(
      {
        tarea: ['', [Validators.required]],
        descripcion: ['', [Validators.required]],
        fechaInicio: [``, [Validators.required]],
        fechaFin: ['', []],
        participantes: new FormArray([
          new FormControl(''),
        ]),
      });
  }
```

```
openModal(template: TemplateRef<any>) {
  this.modalRef = this.modalService.show(
    template,
    Object.assign({}, { class: 'modal-sm' })
  );
  const date = new Date();
  const fechaFormato = `${date.getDate()}  / ${date.getMonth() + 1} / ${date.getFullYear()} `
  this.formTareas.controls['fechaInicio'].setValue(fechaFormato);
  this.formTareas.controls['fechaFin'].setValue('');
  this.formTareas.controls['fechaInicio'].disable();
  this.formTareas.controls['fechaFin'].disable();
}
```

```
//para hacer más grande el modal
setModalClass() {
  this.valueWidth = !this.valueWidth;
  const modalWidth = this.valueWidth ? 'modal-lg' : 'modal-sm';
  this.modalRef?.setClass(modalWidth);
}
```

```
//para los miembros
get member(): FormArray { return this.formTareas.get('participantes') as FormArray; }

addMember() {
  this.member.push(new FormControl());
}
deleteMember(i: number) {
  this.member.removeAt(i);
}
```

```
drop(event: CdkDragDrop<ITarea[]>) {
  if (event.previousContainer === event.container) {
    moveItemInArray(this.porHacer, event.previousIndex, event.currentIndex);
  } else {
    transferArrayItem(event.previousContainer.data, event.container.data, event.previousIndex, event.currentIndex);
  }
  if (event.container.id.match('lista-terminadas')) {
    this.actualizar()
  }

}
actualizar() {
  this.terminada.forEach(x => {
    x.terminado = true;
    const date = new Date();
    const fechaFormato = `${date.getDate()}  / ${date.getMonth() + 1} / ${date.getFullYear()} `
    x.fechaFin = fechaFormato;
  })
}
```

```
guardar() {
  if (this.formTareas.valid) {
    this.addTarea();
    Swal.fire({
      position: 'center',
      title: 'Buen trabajo!',
      text: `submit disparado , formulario es valido`,
      icon: 'info',
    });
    this.formTareas.reset();
    this.modalRef?.hide();

  } else {
    Swal.fire({
      position: 'center',
      title: 'Faltan datos en el formulario',
      text: `submit disparado, formulario No valido`,
      icon: 'warning',
    });
  }
}
```

```
addTarea() {
  this.porHacer.push(
    {
      tarea: `${this.formTareas.controls['tarea'].value}`,
      descripcion: `${this.formTareas.controls['descripcion'].value}`,
      fechaInicio: `${this.formTareas.controls['fechaInicio'].value}`,
      fechaFin: `${this.formTareas.controls['fechaFin'].value}`,
      participantes: this.formTareas.controls['participantes'].getRawValue(),
      terminado: false
    }
  );
  this.formTareas.reset();

}

quitarTarea(i: number) {
  this.porHacer.splice(i, 1);
}
```