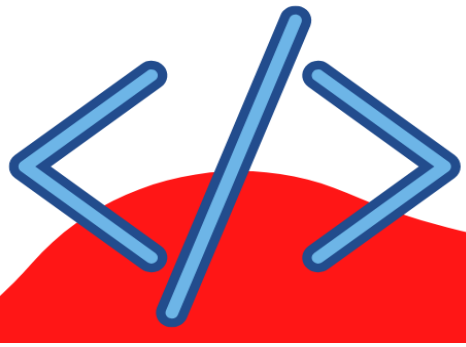
UES-FMP

CURSO DE ESPECIALIZACIÓN MÓDULO FRONT-END

MATERIAL N. 13 UPLOAD Y CALENDAR

****



ING. YANCY ELIZABETH MARTÍNEZ DE MOLINA

Objetivos:

- Implementar la carga de archivos
- Concer fullcalendar

Carga de archivos

Cargar un archivo

- Generar el componenete y el servicio para la acraga de archivos.
- Metodo en el servicio

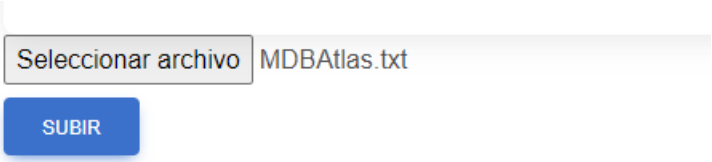
```
export class UploadService {
  url = 'http://localhost:8080/file/upload';
  constructor(private http: HttpClient) { }

  upload(file:File):Observable<Object> {
    const formData = new FormData();
    formData.append('file', file);
    return this.http.post(this.url, formData);
  }

}
```

- En el componente

```
<input type="file" class="file-input"
 (change)="onFileSelected($event)"
  #fileUpload>

<div class="file-upload">
  <button type="button"
  class="btn btn-primary mt-2"
  (click)="subir()">Subir</button>
</div>
```

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

```
subir(){
 this.uploadService.upload(this.file).subscribe((resp) => {
   console.log('repuesta', resp);
 });

}
```



Cargar varios archivos

- Arreglo a utilizar

```
myFiles: string[] = [];
```

- En el Servicio

```
multiple(myFiles: string[]) {
  const formData = new FormData();
  for (var i = 0; i < myFiles.length; i++) {
    formData.append('file', myFiles[i]);
  }
  return this.http.post(`${this.url}/multiple`, formData);
}
```

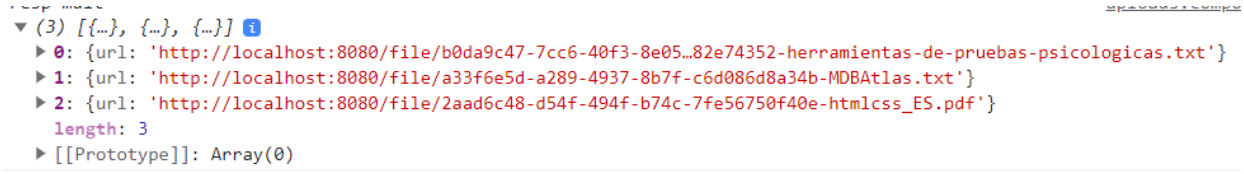
- En el template

```
<h1>Subir varios archivos</h1>

<div class="form-group">
  <label for="file">File</label>
  <input
    formControlName="file"
    id="file"
    type="file"
    multiple
    class="form-control"
    (change)="onFileChange($event)">

  <div class="file-upload mt-2">
    <button class="btn btn-primary"
      type="button"
      (click)="multiple()">Multiple</button>
  </div>
</div>
```

- En el componente

```
onFileChange(event:any) {
  for (let i = 0; i < event.target.files.length; i++) {
      this.myFiles.push(event.target.files[i]);
  }
}
```

```
multiple() {
  this.uploadService.multiple(this.myFiles).subscribe((resp) => {
    console.log('resp mult', resp);
  });
}
```



Reto para entregar en clases

- Implementar **ngx-dropzone-wrapper**

FULLCALENDAR

https://fullcalendar.io/

El primer paso es instalar las dependencias relacionadas con FullCalendar. Necesitará el núcleo FullCalendar, el adaptador Angular y cualquier complemento que se desee usar

1. Instalacion

npm install --save @fullcalendar/angular @fullcalendar/daygrid @fullcalendar/core --force

npm install@fullcalendar/interaction

1. Generar un ponente para implementar fullcalendar
2. Agregar los import en el mòdulo

```
@NgModule({
  declarations: [
    Base64Component,
    ExportarExcelComponent,
    UploadsComponent,
    FullCalendarComponent
  ],
  imports: [
    CommonModule,
    ExcelRoutingModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule,
    DropzoneModule,
    FullCalendarModule,

  ],
  providers: [
```

3. En el componente

```
<div class="container mt-5">
  <div class="row">
    <div class="col-6">
      <h1>Calendario de Actividades</h1>
      <full-calendar [options]="calendarOptions"></full-calendar>

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

```
import { CalendarOptions } from '@fullcalendar/core';
import dayGridPlugin from '@fullcalendar/daygrid';

@Component({
  selector: 'app-full-calendar',
  templateUrl: './full-calendar.component.html',
  styleUrls: ['./full-calendar.component.scss']
})
export class FullCalendarComponent {

  calendarOptions: CalendarOptions = {
    plugins: [dayGridPlugin,
    ],
    initialView: 'dayGridMonth',
  };
```

Calendario de Actividades

July 2023

today

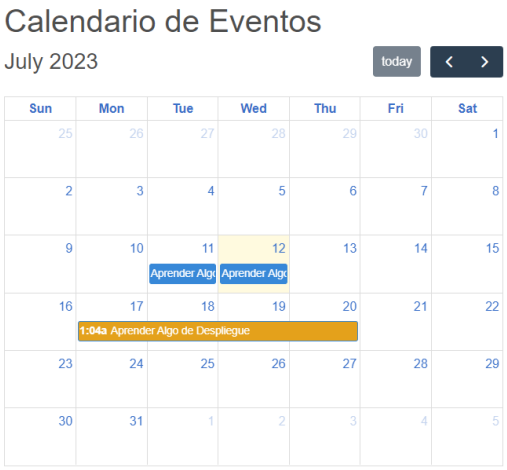
Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

4. Para agregar eventos

```
export class FullCalendarComponent implements OnInit {

  eventos: any = [
    {
      title: 'Aprender Algo de Angular',
      date: '2023-07-11',
      color: '0000FF'
    },
    {…
    },
    {…
    }
  ]

  calendarOptions: CalendarOptions = {
    plugins: [dayGridPlugin,
    ],
    initialView: 'dayGridMonth',
    events: this.eventos,
  };
```

- Tambien los eventos pueden tener la siguiente forma

```
  {
    id: '3',
    title: 'Aprender Algo de Despliegue',
    start: new Date().setDate(new Date().getDate() + 5),
    end: new Date().setDate(new Date().getDate() + 8),
    className: 'bg-warning text-white',
  }
```



5. Instalar npm install @ng-bootstrap/ng-bootstrap@latest
   https://valor-software.com/ngx-bootstrap/#/components

```
import { FullCalendarComponent } from './full-calendar/full-calendar.component';
import { FullCalendarModule } from '@fullcalendar/angular';
import { ModalModule } from 'ngx-bootstrap/modal';

const config: DropzoneConfigInterface = {···
};

@NgModule({
  declarations: [
    Base64Component,
    ExportarExcelComponent,
    UploadsComponent,
    FullCalendarComponent
  ],
  imports: [
    CommonModule,
    ExcelRoutingModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule,
    DropzoneModule,
    FullCalendarModule,
    ModalModule.forRoot()
  ],
```

```
import dayGridPlugin from '@fullcalendar/daygrid';
import { BsModalService, BsModalRef } from 'ngx-bootstrap/modal';
@Component({
  selector: 'app-full-calendar',
  templateUrl: './full-calendar.component.html',
  styleUrls: ['./full-calendar.component.scss']
})
export class FullCalendarComponent implements OnInit {
  modalRef?: BsModalRef;
  eventos: any = [
    {
      title: 'Aprender Algo de Angular',
      date: '2023-07-11',
```

- Agregando el evento click en las opciones del calendario, para levantar el modal

```
calendarOptions: CalendarOptions = {
  plugins: [dayGridPlugin, interactionPlugin],
  initialView: 'dayGridMonth',
  themeSystem: 'bootstrap',
  events: this.eventos,
  dateClick: this.openModal.bind(this),

};
```

```
openModal(event: EventInput) {
  this.nuevoEvento = event;
  this.modalRef = this.modalService.show(this.templateNuevo, this.config);
}
```

```
constructor(private modalService: BsModalService, private fb: FormBuilder) {  }
```

```
  @ViewChild('templateNuevo') templateNuevo!: string;
  config = {
    animated: true
  }
```

```
<ng-template #templateNuevo>
  <div class="modal-header bg-primary text-white">
    <h4 class="modal-title pull-left">Agregar Eventos</h4>
    <button type="button"
      class="btn-close 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>
</ng-template>
```

Al dar click en una opción de fecha del calendario, se habilitarà el modal.

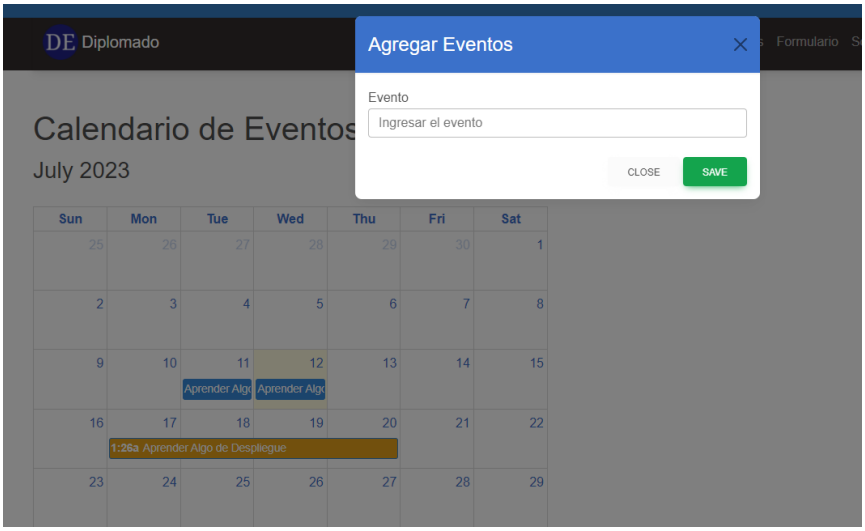
- Agregando formulario al modal

```
ngOnInit(): void {
  this.formEvento = this.fb.group({
    title: ['', [Validators.required]],
  });


}
```

```
  <div class="modal-body">
    <form (ngSubmit)="guardarEvent()" [formGroup]="formEvento">
      <div class="row">
        <div class="col-12">
          <div class="mb-3">
            <label class="control-label">Evento</label>
            <input class="form-control" placeholder="Ingresar el evento"
              type="text" name="title" formControlName="title" />
          </div>
        </div>
      </div>
      <div class="text-end mt-2">
        <button type="button" class="btn btn-light me-1"
          (click)="closeEventModal()">
          Close
        </button>
        <button type="submit" class="btn btn-success" id="btn-save-event">
          Save
        </button>
      </div>
    </form>
  </div>
</ng-template>
```

```
guardarEvent() {
  if (this.formEvento.valid) {
    const title = this.formEvento.get('title')!.value;
    const calendar: Calendar = this.nuevoEvento["view"].calendar;
    calendar.addEvent({
      id: '4',
      title: title,
      start: this.nuevoEvento.date,
      end: this.nuevoEvento.date,
      className: 'bg-success text-white',
    })

    this.formEvento = this.fb.group({
      title: '',
    });
  }
  this.closeEventModal();
  this.modalRef?.hide();
}
```

```
closeEventModal() {
  this.formEvento = this.fb.group({
    title: '',
  });
}
```



Para editar un evento

- Las variables

```
formEditEvento!: FormGroup;
editEvent!: EventImpl;
calendarEvents!: EventInput[];
```

- El ViewChild

```
@ViewChild('templateNuevo') templateNuevo!: string;
@ViewChild('templateEditar') templateEditar!: string;
config = {
  animated: true
}
```

- Agregando el eventoClick en calendarOptions

```
calendarOptions: CalendarOptions = {
  plugins: [dayGridPlugin, interactionPlugin],
  initialView: 'dayGridMonth',
  themeSystem: 'bootstrap',
  events: this.eventos,
  dateClick: this.openModal.bind(this),
  eventClick: this.handleEventClick.bind(this),
};
```

```
handleEventClick(datos: EventClickArg) {
  this.editEvent = datos.event
  this.formEditEvento = this.fb.group({
    title: `${datos.event.title}`,
  });
  this.modalRef = this.modalService.show(this.templateEditar, this.config);
}
```

- Definiendo el formulario para editar

```
ngOnInit(): void {
  this.formEvento = this.fb.group({
    title: ['', [Validators.required]],
  });
  this.formEditEvento = this.fb.group({
    editTitle: ['', [Validators.required]],
  });
  this.actualizar();
}
```

- Agregando en ng-template

```
<ng-template #templateEditar>
  <div class="modal-header bg-primary text-white">
    <h4 class="modal-title pull-left">Editar Evento</h4>
    <button type="button"
      class="btn-close close pull-right"
      aria-label="Close"
      (click)="modalRef?.hide()">
      <span aria-hidden="true"
        class="visually-hidden">&times;</span>
    </button>
  </div>
  <div class="modal-body">
    <form (ngSubmit)="guardarEdicion()" [formGroup]="formEditEvento">
      <div class="row">
        <div class="col-12">
          <div class="mb-3">
            <label class="control-label">Evento</label>
            <input class="form-control" placeholder="Ingresar el evento"
              type="text" name="title" formControlName="title" />
          </div>
        </div>
      </div>
      <div class="text-end mt-2">
        <button type="button" class="btn btn-danger delete-event ms-1"
          (click)="confirm()">Delete</button>
        <button type="button" class="btn btn-light me-1"
          (click)="closeEventModal()">
          Close
        </button>
        <button type="submit" class="btn btn-success" id="btn-save-event">
          Save
        </button>
      </div>
    </form>
  </div>
</ng-template>
```

- Para actualizar la edición en el calendario

```
guardarEdicion() {
  const editTitle = this.formEditEvento.get('title')!.value;
  const editId = this.calendarEvents.findIndex(
    (x) => {
      x.id + '' === this.editEvent.id + ''
    }
  );
  this.editEvent.setProp('title', editTitle);

  this.calendarEvents[editId] = {
    ...this.editEvent,
    title: editTitle,
    id: this.editEvent.id,
    classNames: 'bg-success text-white',
  };

  this.formEditEvento = this.fb.group({
    editTitle: '',
    editCategory: '',
  });
  this.modalRef?.hide();
}
```

- Eliminando un evento del calendario

```
confirm() {
  Swal.fire({
    title: 'Are you sure?',
    text: 'You won\'t be able to revert this!',
    icon: 'warning',
    showCancelButton: true,
    confirmButtonColor: '#34c38f',
    cancelButtonColor: '#f46a6a',
    confirmButtonText: 'Yes, delete it!',
  }).then((result) => {
    if (result.value) {
      this.deleteEventData();
      Swal.fire('Deleted!', 'Event has been deleted.', 'success');
    }
  });
}
deleteEventData() {
  this.editEvent.remove();
  this.modalRef?.hide();
}
```