DOCUMENTACIÓN TÉCNICA

Fuensanta Sansano Montoya
Tomás Raigal López

2.º Desarrollo de Aplicaciones Web

ÍNDICE

```
const journeyStartTime3 = document.getElementById('journeyStartTime3');
const journeyEndTime3 = document.getElementById('journeyEndTime3');
midJourneyOption2.disabled = true;
fullJourneyOption2.disabled = true;
journeyStartTime2.disabled = true;
journeyEndTime2.disabled = true;
midJourneyOption3.disabled = true;
fullJourneyOption3.disabled = true;
journeyStartTime3.disabled = true;
journeyEndTime3.disabled = true;
});
function showHours2() {
const missingDay2 = document.getElementById('missingDay2');
const midJourneyOption2 = document.getElementById('midJourneyOption2');
const fullJourneyOption2 = document.getElementById('fullJourneyOption2');
const journeyStartTime2 = document.getElementById('journeyStartTime2');
const journeyEndTime2 = document.getElementById('journeyEndTime2');
if(missingDay2.value == '') {
midJourneyOption2.disabled = true;
fullJourneyOption2.disabled = true;
journeyStartTime2.disabled = true;
journeyEndTime2.disabled = true;
} else {
midJourneyOption2.disabled = false;
fullJourneyOption2.disabled = false;
journeyStartTime2.disabled = false;
journeyEndTime2.disabled = false;
}
}
function showHours3() {
const missingDay3 = document.getElementById('missingDay3');
const midJourneyOption3 = document.getElementById('midJourneyOption3');
const fullJourneyOption3 = document.getElementById('fullJourneyOption3');
const journeyStartTime3 = document.getElementById('journeyStartTime3');
const journeyEndTime3 = document.getElementById('journeyEndTime3');
if(missingDay3.value == '') {
midJourneyOption3.disabled = true;
fullJourneyOption3.disabled = true;
journeyStartTime3.disabled = true;
```

```
journeyEndTime3.disabled = true;                                    23
} else {                                                            23
midJourneyOption3.disabled = false;                                 23
fullJourneyOption3.disabled = false;                                23
journeyStartTime3.disabled = false;                                 23
journeyEndTime3.disabled = false;                                   23
}                                                                   23
}                                                                   23
const journeyStartTime1 = document.getElementById('journeyStartTime1');  23
const journeyEndTime1 = document.getElementById('journeyEndTime1');  23
document.getElementById('fullJourneyOption1').addEventListener('click', (e) => {  23
journeyStartTime1.disabled = true;                                  23
journeyEndTime1.disabled = true;                                    23
});                                                                 23
document.getElementById('midJourneyOption1').addEventListener('click', (e) => {  24
journeyStartTime1.disabled = false;                                 24
journeyEndTime1.disabled = false;                                   24
});                                                                 24
const journeyStartTime2 = document.getElementById('journeyStartTime2');  24
const journeyEndTime2 = document.getElementById('journeyEndTime2');  24
document.getElementById('fullJourneyOption2').addEventListener('click', (e) => {  24
journeyStartTime2.disabled = true;                                  24
journeyEndTime2.disabled = true;                                    24
});                                                                 24
document.getElementById('midJourneyOption2').addEventListener('click', (e) => {  24
journeyStartTime2.disabled = false;                                 24
journeyEndTime2.disabled = false;                                   24
});                                                                 24
const journeyStartTime3 = document.getElementById('journeyStartTime3');  24
const journeyEndTime3 = document.getElementById('journeyEndTime3');  24
document.getElementById('fullJourneyOption3').addEventListener('click', (e) => {  24
journeyStartTime3.disabled = true;                                  25
journeyEndTime3.disabled = true;                                    25
});                                                                 25
document.getElementById('midJourneyOption3').addEventListener('click', (e) => {  25
journeyStartTime3.disabled = false;                                 25
journeyEndTime3.disabled = false;                                   25
});                                                                 25
function showHiddenFields() {                                       25
```

INTRODUCCIÓN

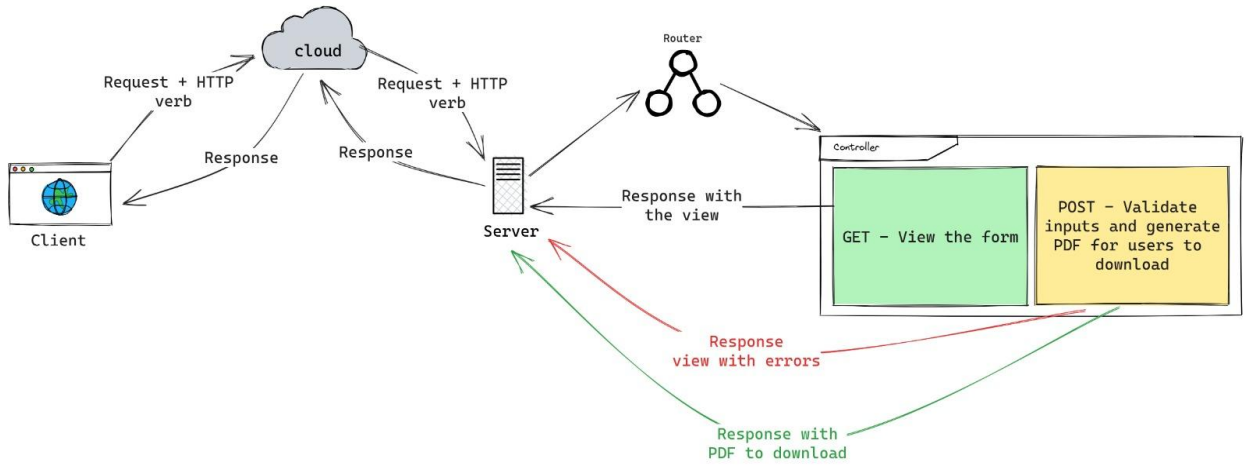
El propósito principal de esta página web es facilitar la creación y descarga de

documentos personalizados en formato PDF, utilizando información ingresada por los usuarios en formularios interactivos. Al proporcionar una interfaz intuitiva y amigable, los usuarios podrán ingresar sus datos de manera sencilla y obtener rápidamente documentos generados automáticamente.

A lo largo de esta documentación, se explorarán en detalle los componentes esenciales de la página web, desde el *front-end* hasta el *back-end*. Se describe la arquitectura general del sistema, las tecnologías utilizadas y las funcionalidades.

Esta documentación técnica está diseñada para brindar un recurso valioso a todas las personas interesadas en comprender cómo funciona esta página web, que se especializa en el rellenado de formularios y la generación de archivos PDF.

Para poner un poco en contexto, el flujo de la aplicación está representado gráficamente en la siguiente imagen.



RUTAS

Para esta aplicación se han creado 7 rutas.

La primera hace una petición de tipo *GET* a la página principal y solo necesita que retorne una vista.

```
Route::get('/', function () {

   return view('welcome');

});
```

Estas 3 de tipo *GET* referencian a los 3 formularios existentes en la aplicación.

```
Route::get('/extracurricular-activity', static function () {

   return view('livewire/extracurricularActivity');

});

Route::get('/family-authorization', static function () {

   return view('livewire/familyAuthorization');

});

Route::get('/proof-missing-teacher', static function () {

   return view('livewire/proofMissingTeacher');

});
```

Y estas 3 citan a los controladores que se encargan de la generación de los pdfs. Son de tipo *POST* ya que las vistas asociadas a cada clase necesitan de unos datos que imprimir.

```
Route::post('/extracurricular-activity', [ExtracurricularActivity::class,
'store']);

Route::post('/family-authorization', [FamilyAuthorization::class,
'store']);

Route::post('/proof-missing-teacher', [ProofMissingTeacher::class,
'store']);
```

ESTILOS

- ## FORMULARIOS

Para los estilos de los formularios se ha usado ***Bootstrap*** en su última versión, la 5.3. Al principio se tuvo problemas con la importación a través de su *CDN* y decidimos que sería de mucha más ayuda descargarla y añadirla directamente como parte de los archivos del proyecto.

- ## PDFs

Para los estilos se quería usar *Bootstrap*, al igual que en los formularios, pero había un problema y es que, a la hora de la generación del PDF, *Bootstrap* no se renderizaba y se quedaba el archivo solo con el texto, sin estilos.

La solución, después de buscar información y probar distintas cosas, se vió que lo que funcionaba bien era *CSS* puro, por lo tanto, cada PDF, que es una vista *HTML*, tiene su propia hoja de estilos asociada.

DOMPDF

DOMPDF es un motor de maquetación y renderizado *HTML* escrito en *PHP*. Es un renderizador orientado al estilo: descargará y leerá hojas de estilo externas, etiquetas de estilo en línea y los atributos de estilo de elementos *HTML* individuales. También soporta la mayoría de los atributos de presentación *HTML*.

Decidimos usar esta librería porque es la más famosa de todas las que hacen esta función y se sigue actualizando a día de hoy. Además de esto, es la que más funcionalidades tiene y la que mejor nos venía al estar usando *PHP*.

Para instalarlo mediante *composer* (También se puede hacer descargando y metiendolo al proyecto directamente o a través de *git*) se usa este comando y ya se puede empezar a usarlo.

```
composer require dompdf/dompdf
```

Y para usarlo solo hace falta importarlo.

```
use Dompdf\Dompdf;
```

TRAIT GENERACIÓN PDF

Al principio se tenía en cada controlador un método para la generación de cada PDF, pero, a la hora de refactorizar código, se vió que tenían cosas en común y que se podía extraer a un único método.

Este es el trait encargado de generar los PDFs y va a recibir 3 cosas: el nombre de la vista a la que tiene que mandar los datos recibidos del formulario, el nombre que se le quiere dar al archivo .pdf que se descarga y los datos en sí.

Esta primera parte va a coger la imagen que va a estar en la cabecera, esta se encuentra en la carpeta *public* del proyecto y va a ser codificada en *base64* y añadida a los datos que le llegan a la vista. Se optó por la opción de convertir la imagen a *base64* porque *DomPDF* no puede acceder bien a las rutas públicas de *Laravel Forge* (sitio donde se desplegó la aplicación) y de esta forma, se solucionaba el problema.

```
use Dompdf\Dompdf;

use Dompdf\Options;

trait PDF

{

   public function generatePDF($pdfViewName, $downloadFileName,
$data)

   {

       $headerPath = public_path('/images/cabecerav6.png');

        $typeOfHeader = pathinfo($headerPath, PATHINFO_EXTENSION);

        $imageContent = file_get_contents($headerPath);

        $base64HeaderImage = '' . base64_encode($imageContent);

        $data['header'] = $base64HeaderImage;
```

Options son las opciones de *DOMPDF*. En este caso, "*isRemoteEnabled*" se establece a true para proteger información potencialmente sensible y "*isHtml5ParserEnabled*" es para que pueda interpretar bien todas las etiquetas de *HTML5*.

```
        $options = new Options();

        $options->set('isRemoteEnabled', true);

        $options->set('isHtml5ParserEnabled', true);
```

Se carga la vista que se le ha pasado y con los datos que ha recibido. Se establece el tamaño y la orientación de la página, A4 y horizontal en este caso. Se renderiza y se descarga el archivo pdf con el nombre que se le ha definido.

```
        $html = view($pdfViewName, compact('data'))->render();

        $dompdf = new Dompdf($options);

        $dompdf->loadHtml($html);

        $dompdf->setPaper('A4', 'portrait');

        $dompdf->render();

        return $dompdf->stream($downloadFileName);

    }

}
```

REQUESTS

Se va a presentar las request de la aplicación de las cuáles, únicamente, se va a poner una imagen de la estructura.

En esta parte vemos todas las reglas de validación de las que van a hacer uso los campos del formulario correspondiente.

```
class ExtracurricularActivityRequest extends FormRequest
{
   public function rules()
   {
       return [
           'activity_name' => 'required|string',
           'activity_place' => 'required|string',
           'activity_module' => 'required|string',
           'activity_departament' => 'required|string|max:100',
           'teachers' => 'required|string',
           'student_groups' => 'required|string',
           'date' => 'required|date_format:Y-m-d',
           'departure_time' => 'required|date_format:H:i',
           'arrive_time' => 'required|date_format:H:i',
           'activity_price' =>
'required|regex:/^\d+([.,])?\d{0,2}$/',
           'transport' => 'required|string',
           'activity_responsible_teacher' =>
'required|string|max:50',
       ];
   }

   public function messages()
   {
       return [
           'activity_name.required' => 'El nombre de la actividad es
     obligatorio',
           'activity_place.required' => 'El lugar de la actividad
```

```
debe
     ser obligatorio',
          'activity_module.required' => 'El modulo al que pertenece
la actividad debe ser obligatorio',
          'activity_departament.required' => 'El departamento al que
pertenece la actividad debe ser obligatorio',
          'activity_departament.max' => 'El departamento no puede
tener más de 100 carácteres',
          'teachers.required' => 'El campo profesores que van a
asistir a la actividad debe ser obligatorio',
          'student_groups.required' => 'El campo grupo de
estudiantes debe ser obligatorio',
          'date.required' => 'El campo fecha de la actividad debe
ser obligatorio',
          'date.date_format' => 'El campo fecha de la actividad debe
ser en formato día/mes/año',
          'departure_time.required' => 'El campo hora de salida debe
ser obligatorio',
          'departure_time.date_format' => 'El campo hora de salida
debe de ser en formato hh:mm',
          'arrive_time.required' => 'El campo hora de llegada debe
ser obligatorio',
          'arrive_time.date_format' => 'El campo hora de llegada
debe de ser en formato hh:mm',
          'activity_price.required' => 'El campo precio de la
actividad debe ser obligatorio',
          'activity_price.regex' => 'El campo precio debe de ser
numérico y positivo',
          'transport.required' => 'El campo transporte debe ser
obligatorio',
          'activity_responsible_teacher.required' => 'El campo
responsable de la actividad debe ser obligatorio',
          'activity_responsible_teacher.max' => 'El nombre del
profesor no puede tener más de 50 carácteres'
       ];
   }
```

```
}
```

- Autorización familiar

```
class FamilyAuthorizationRequest extends FormRequest
{
   public function rules()
   {
       return [
           'activity' => 'required|string',
           'organizer' => 'required|string|max:100',
           'execution_date' => 'required|date_format:Y-m-d',
           'departure_time' => 'required|date_format:H:i',
           'goals' => 'required|string|max:250',
           'deadline' => 'required|date_format:Y-m-d',
           'parents' => 'required|string|max:50',
           'student' => 'required|string|max:50',
           'course' => 'required|string',
           'authorization' => 'required',
           'dni' => ['required',
'regex:/([a-z]|[A-Z]|[0-9])[0-9]{7}([a-z]|[A-Z]|[0-9])/'],
       ];
   }

   public function messages()
   {
       return [
           'activity.required' => 'La actividad es obligatoria',
           'organizer.required' => 'El nombre del organizador es
obligatorio',
           'organizer.max' => 'El nombre del organizador no puede
tener más de 100 carácteres',
           'execution_date.required' => 'La fecha de la actividad es
obligatoria',
           'execution_date.date_format' => 'La fecha de la actividad
debe tener un formato válido',
```

```
            'departure_time.required' => 'La hora de salida es
obligatoria',
            'departure_time.date_format' => 'La hora debe tener un
formato válido',
            'goals.required' => 'Los objetivos son obligatorios',
            'goals.max' => 'Los objetivos y contenidos no pueden tener
más de 250 carácteres',
            'deadline.required' => 'La fecha de entrega de la hoja es
obligatoria',
            'deadline.date_format' => 'La fecha de entrega de la hoja
debe tener un formato válido',
            'parents.required' => 'El nombre del padre/madre/tutor es
obligatorio',
            'parents.max' => 'El nombre del padre/madre/tutor no puede
tener más de 50 carácteres',
            'student.required' => 'El nombre del alumno es
obligatorio',
            'student.max' => 'El nombre del alumno no puede tener más
de 50 carácteres',
            'course.required' => 'El curso del alumno es obligatorio',
            'authorization.required' => 'La autorización es
obligatoria',
            'dni.required' => 'El DNI del padre/madre/tutor es
obligatorio',
            'dni.regex' => 'El DNI debe tener un formato válido',
        ];
    }
}
```

- Justificante falta profesorado

```
class ProofMissingTeacherRequest extends FormRequest
{
   public function rules()
   {
       return [
           'name' => 'required|string|max:50',
           'department' => 'required|string|max:100',
           'dni' => ['required',
'regex:/([a-z]|[A-Z]|[0-9])[0-9]{7}([a-z]|[A-Z]|[0-9])/'],

           'missingDay1' => 'required|date_format:Y-m-d',
           'journeyType1' => 'required',
           'journeyStartTime1' =>
'exclude_if:journeyType1,fullJourneyOption1|required|date_format:H:i',
           'journeyEndTime1' =>
'exclude_if:journeyType1,fullJourneyOption1|required|date_format:H:i',

           'missingDay2' => 'nullable|date_format:Y-m-d',
           'journeyType2' => 'nullable',
           'journeyStartTime2' =>
'required_if:journeyType2,date|date_format:H:i',
           'journeyEndTime2' =>
'required_if:journeyType2,date|date_format:H:i',

           'missingDay3' => 'nullable|date_format:Y-m-d',
           'journeyType3' => 'nullable',
           'journeyStartTime3' =>
'required_if:journeyType3,date|date_format:H:i',
           'journeyEndTime3' =>
'required_if:journeyType3,date|date_format:H:i',

           'permissionsSelect' => 'required',
```

```
            'reason' => 'nullable|requiredIf:in:'.implode(',',
array_keys(config('specialMedicalReasons'))) .'|max:100',
            'anotherReason' => 'required_with:reasons:'.implode(',',
array_keys(config('specialMedicalReasons'))) .'|max:100'
        ];
    }

    public function messages()
    {
        return [
            'name.required' => 'El nombre del profesor es obligatorio',
            'name.max' => 'El nombre no puede tener más de 50 carácteres',
            'department.required' => 'El departamento es obligatorio',
            'department.max' =>  'El departamento no puede tener más de 100
carácteres',
            'dni.required' => 'El DNI es obligatorio',
            'dni.regex' => 'Tiene que tener ser un NIF o NIE válido',

            'missingDay1.required' => 'El día de falta es obligatorio',
            'missingDay1.date_format' => 'El día de falta debe tener un
formato válido',
            'journeyType1.required' => 'Es obligatorio elegir una opción de
las dos',
            'journeyStartTime1.required' => 'Ambas horas son obligatorias
si se ha marcado la segunda opción',
            'journeyStartTime1.date_format' => 'Ambas horas deben tener un
formato válido',
            'journeyEndTime1.required' => 'Ambas horas son obligatorias si
se ha marcado la segunda opción',
            'journeyEndTime1.date_format' => 'Ambas horas deben tener un
formato válido',

            'missingDay2.date_format' => 'El día de falta debe tener un
formato válido',
            'journeyStartTime2.date_format' => 'Ambas horas deben tener un
formato válido',
            'journeyStartTime2.required' => 'Ambas horas son obligatorias
```

```
si se ha rellenado la fecha',
            'journeyEndTime2.date_format' => 'Ambas horas deben tener un
formato válido',
            'journeyEndTime2.required' => 'Ambas horas son obligatorias si
se ha rellenado la fecha',

            'missingDay3.date_format' => 'El día de falta debe tener un
formato válido',
            'journeyStartTime3.date_format' => 'Ambas horas deben tener un
formato válido',
            'journeyStartTime3.required' => 'Ambas horas son obligatorias
si se ha rellenado la fecha',
            'journeyEndTime3.date_format' => 'Ambas horas deben tener un
formato válido',
            'journeyEndTime3.required' => 'Ambas horas son obligatorias si
se ha rellenado la fecha',

            'permissionsSelect.required' => 'Debe seleccionar un motivo',
            'anotherReason.required' => 'Este campo es obligatorio',
            'anotherReason.max' => 'El motivo no puede tener más de 100
carácteres'
        ];
    }
}
```

CONTROLADORES

Para explicar los controladores, debido a que la funcionalidad de todos ellos es la misma, se va a explicar uno de ellos, el de justificante de falta de profesorado, ya que es el más completo.

Todos los controladores hacen uso del *trait* mencionado anteriormente.

```
class ProofMissingTeacher extends Component
{
  use PDF;
```

Se establece la vista del formulario a la que está asociado para que esta se renderice.

```
  public function render()
  {
     return view('livewire.proofMissingTeacher');
  }
```

Este es el método, que tiene cada controlador, para recibir los datos del formulario y mandarselos al *trait* para generar el pdf.

```
  public function store(ProofMissingTeacherRequest $request)
  {
```

Esta sección inicial tiene como propósito presentar una manera más legible en la que se visualizará la fecha en los documentos en formato PDF. Para lograr esto, se establece la zona horaria y se obtiene la fecha actual correspondiente.

```
date_default_timezone_set('Europe/Madrid');

$actual_date = getdate();
```

A continuación, se procede a crear un *array asociativo*, en el cual cada número se asigna a un mes específico del calendario.

```
$month = [
    1 => 'Enero',
    2 => 'Febrero',
    3 => 'Marzo',
    4 => 'Abril',
    5 => 'Mayo',
    6 => 'Junio',
    7 => 'Julio',
    8 => 'Agosto',
    9 => 'Septiembre',
    10 => 'Octubre',
    11 => 'Noviembre',
    12 => 'Diciembre'
];
```

Dado que los meses se obtienen por defecto como valores numéricos a través de la función "*getdate()*", es necesario realizar una conversión adecuada. Para lograrlo, se establece la correspondencia entre el número del mes obtenido por defecto y su respectivo *clave-valor* en el arreglo asociativo.

```
$actual_date['mon'] = $month[$actual_date['mon']];
```

En este *array asociativo* se almacenan todos los datos provenientes del formulario, una vez que han sido validados en la *request* correspondiente. Como se puede observar, existen tres variables que se envían como argumentos a un método denominado *dateFormat()*. El propósito de este método es formatear la fecha de manera que se imprima en un formato más reconocible en España.

```
$data = [
    'name' => $request->name,
    'department' => $request->department,
    'dni' => $request->dni,
    'missingDay1' => $this->dateFormat($request->missingDay1),
    'journeyType1' => $request->journeyType1,
    'journeyStartTime1' => $request->journeyStartTime1,
    'journeyEndTime1' => $request->journeyEndTime1,
    'missingDay2' => $this->dateFormat($request->missingDay2),
    'journeyType2' => $request->journeyType2,
    'journeyStartTime2' => $request->journeyStartTime2,
    'journeyEndTime2' => $request->journeyEndTime2,
    'missingDay3' => $this->dateFormat($request->missingDay3),
    'journeyType3' => $request->journeyType3,
```

```
        'journeyStartTime3' => $request->journeyStartTime3,
        'journeyEndTime3' => $request->journeyEndTime3,
        'day' => $actual_date['mday'],
        'month' => $actual_date['mon'],
        'year' => $actual_date['year'],
        'permissionsSelect' => $request->permissionsSelect,
        'reason' => $request->reason,
        'anotherReason' => $request->anotherReason,
    ];
            return $this->generatePDF('pdfs.proofMissingTeacherPDF', 'justificante falta profesorado.pdf', $data);
  }

  public function dateFormat($date) {
    if($date == null) {
      return null;
    }
    return date('d-m-Y', strtotime($date));
  }
}
```

VISTAS

Al igual que los controladores, se va a presentar únicamente una vista de cada de apartado, ya que, en general, son iguales salvando diferencias de estilos y campos.

También se debe mencionar que los formularios están dotados de memoria, es decir, si el usuario se equivoca en algún campo, los datos que rellenó seguirán estando en el formulario, no se habrán borrado.

- ## Vistas de formularios
    - ### Justificante de falta de profesor

Algo que se debe de mencionar de este formulario es que tiene JavaScript que se ejecuta al cargar la página para habilitar o deshabilitar los radio buttons y las horas y, además, si se selecciona la segunda opción de los botones de selección, es decir, “no he faltado la jornada completa” también se deshabilitan las horas.

También se usa JavaScript para que si se elige la opción L2 o P14 como motivo en el desplegable aparecen una serie de nuevas opciones, como la elección de radio buttons o la entrada de texto mediante un campo de tipo texto si ninguna de las opciones ofrecidas en los radio buttons satisfacen la necesidad del usuario.

```
<script>
   function displayAnotherReason() {
       const anotherReason =
document.getElementById('anotherReason');
       anotherReason.style.display = 'block';
       anotherReason.disabled = false;
   }

   function hideAnotherReason() {
       const anotherReason =
document.getElementById('anotherReason');
       anotherReason.style.display = 'none';
       anotherReason.disabled = false;
   }

   function updateAnotherValue() {
       const another = document.getElementById('another');
       const anotherReason =
document.getElementById('anotherReason');
       another.value = anotherReason.value;
   }

   window.addEventListener('load', (e) => {
       const midJourneyOption2 =
```

```
document.getElementById('midJourneyOption2');

        const fullJourneyOption2 =
document.getElementById('fullJourneyOption2');

        const midJourneyOption3 =
document.getElementById('midJourneyOption3');

        const fullJourneyOption3 =
document.getElementById('fullJourneyOption3');

        const journeyStartTime2 =
document.getElementById('journeyStartTime2');

        const journeyEndTime2 =
document.getElementById('journeyEndTime2');

        const journeyStartTime3 =
document.getElementById('journeyStartTime3');

        const journeyEndTime3 =
document.getElementById('journeyEndTime3');



        midJourneyOption2.disabled = true;

        fullJourneyOption2.disabled = true;

        journeyStartTime2.disabled = true;

        journeyEndTime2.disabled = true;



        midJourneyOption3.disabled = true;

        fullJourneyOption3.disabled = true;

        journeyStartTime3.disabled = true;

        journeyEndTime3.disabled = true;

    });
```

```
   function showHours2() {
       const missingDay2 = document.getElementById('missingDay2');
       const midJourneyOption2 =
document.getElementById('midJourneyOption2');
       const fullJourneyOption2 =
document.getElementById('fullJourneyOption2');
       const journeyStartTime2 =
document.getElementById('journeyStartTime2');
       const journeyEndTime2 =
document.getElementById('journeyEndTime2');

       if(missingDay2.value == '') {
           midJourneyOption2.disabled = true;
           fullJourneyOption2.disabled = true;
           journeyStartTime2.disabled = true;
           journeyEndTime2.disabled = true;
       } else {
           midJourneyOption2.disabled = false;
           fullJourneyOption2.disabled = false;
           journeyStartTime2.disabled = false;
           journeyEndTime2.disabled = false;
       }
   }
```

```
   function showHours3() {
       const missingDay3 = document.getElementById('missingDay3');
       const midJourneyOption3 =
document.getElementById('midJourneyOption3');
       const fullJourneyOption3 =
document.getElementById('fullJourneyOption3');
       const journeyStartTime3 =
document.getElementById('journeyStartTime3');
       const journeyEndTime3 =
document.getElementById('journeyEndTime3');

       if(missingDay3.value == '') {
           midJourneyOption3.disabled = true;
           fullJourneyOption3.disabled = true;
           journeyStartTime3.disabled = true;
           journeyEndTime3.disabled = true;
       } else {
           midJourneyOption3.disabled = false;
           fullJourneyOption3.disabled = false;
           journeyStartTime3.disabled = false;
           journeyEndTime3.disabled = false;
       }
   }
```

```
   const journeyStartTime1 =
document.getElementById('journeyStartTime1');
   const journeyEndTime1 =
document.getElementById('journeyEndTime1');

document.getElementById('fullJourneyOption1').addEventListener('click
', (e) => {
       journeyStartTime1.disabled = true;
       journeyEndTime1.disabled = true;
   });

document.getElementById('midJourneyOption1').addEventListener('click'
, (e) => {
       journeyStartTime1.disabled = false;
       journeyEndTime1.disabled = false;
   });

   const journeyStartTime2 =
document.getElementById('journeyStartTime2');
   const journeyEndTime2 =
document.getElementById('journeyEndTime2');

document.getElementById('fullJourneyOption2').addEventListener('click
', (e) => {
```

```
        journeyStartTime2.disabled = true;

        journeyEndTime2.disabled = true;

    });



document.getElementById('midJourneyOption2').addEventListener('click'
, (e) => {

        journeyStartTime2.disabled = false;

        journeyEndTime2.disabled = false;

    });


    const journeyStartTime3 =
document.getElementById('journeyStartTime3');

    const journeyEndTime3 =
document.getElementById('journeyEndTime3');


document.getElementById('fullJourneyOption3').addEventListener('click
', (e) => {

        journeyStartTime3.disabled = true;

        journeyEndTime3.disabled = true;

    });



document.getElementById('midJourneyOption3').addEventListener('click'
, (e) => {

        journeyStartTime3.disabled = false;
```

```
        journeyEndTime3.disabled = false;
    });


    function showHiddenFields() {
        const permissionsSelect =
document.querySelector('#permissionsSelect');
        const selectedOption =
permissionsSelect.options[permissionsSelect.selectedIndex].id;
        const medicalProof = document.querySelector('#medicalProof');


        if (selectedOption === 'L2' || selectedOption === 'P14') {
            medicalProof.style.display = "block";
        } else {
            medicalProof.style.display = "none";
        }
    }
</script>
```

- Vistas de PDFs
    - Justificante de falta de profesor

Región de Murcia
Consejería de Educación, Formación Profesional y Empleo
Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"
Formación Profesional DUAL
Erasmus+
IES ingeniero de la cierva
C/La Iglesia, s/n
30012 Patiño (Murcia) 968266922
30010978@murciaeduca.es

JUSTIFICANTE DE FALTA DEL PROFESORADO

Nombre	Pedro
Departamento	Informática

JUSTIFICA que no pudo asistir al centro de trabajo los días:

13-06-2023	La **jornada completa**
14-06-2023	De **08:00 a 11:00**
15-06-2023	De **11:00 a 14:00**

Por el siguiente motivo: **L2: Ausencia por enfermedad no superior a tres días. (10)** debido a que hay otro motivo justificado

Murcia, a **12** de **Junio** de **2023**

Firmado por:_____ con DNI: **12345678A**