

Documentación Grupo 3

VISOR SEMÁNTICO

 <https://alumnes-ltim.uib.es/gdie2203/>

Aguilar Ferrer, Fèlix Lluís

Torres Torres, Marc

Huzum, Ionut Florin

Índice

Índice	1
Descripción de funcionalidades a nivel de interfaz de usuario	2
Explicación técnica del proceso de codificación y publicación del material multimedia	4
Explicación técnica de funcionalidades desarrolladas	6
Reproductor	6
Iniciar el reproductor:	6
Reproducir Audio:	6
Ocultar/mostrar subtítulos:	7
Carga de metadatos:	7
Iniciar player:	8
Actualización de la barra de progreso:	8
Botones del player:	10
Avanzar/retroceder el player:	11
Editor	11
Seleccionar archivo:	11
Abrir un archivo:	13
Guardar un archivo:	14
Eliminar un archivo:	14
Eliminar Segmento:	16
Añadir Segmento:	17
Web-responsive	19
Versión tablet:	21
Versión Móvil:	23
Problemas y soluciones adoptadas durante la práctica	25
Opinión de la práctica	26
Mejoras adicionales	26
Autoevaluación	26

Descripción de funcionalidades a nivel de interfaz de usuario

Nuestra práctica consta de varias funcionalidades a nivel de interfaz de usuario, siendo las más relevantes el ajuste automático de la dimensión y contenido del video según desde donde se mire (web-responsive), la interacción con un reproductor y la posibilidad de editar los metadatos del vídeo visto en el reproductor.

El reproductor, será la parte más visual de nuestra práctica donde el usuario podrá ver los lugares más emblemáticos de Japón y disfrutar de una serie de funcionalidades:

- **Opciones del reproductor de vídeo**, el usuario podrá adelantar/atrasar, pausar/iniciar y ver la barra de progreso del vídeo con los tiempos actualizándose en tiempo real así como tener la oportunidad de pulsar sobre esta barra para ir a ese punto del vídeo.
- **Google maps**, al tratarse el vídeo de ir mostrando diferentes lugares de japón encontramos que tener un google maps que se va actualizando según las coordenadas del lugar que se muestra en ese preciso instante y que este mapa sea, además, interactivo es una funcionalidad que aporta valor al usuario.
- **Botones del reproductor de vídeo**, el usuario podrá elegir tener una pista de audio en la que se pronuncien los nombres de las regiones mostradas de forma correcta en español, así como la opción de poder desactivar este audio y gozar del vídeo en silencio.

El usuario podrá moverse entre dos páginas, la del reproductor, explicada anteriormente, y la del editor de metadatos, página no tan dirigida a usuarios públicos sino a un usuario con más privilegios capaz de modificar los metadatos de los vídeos, esta página contiene lo siguiente:

Los botones de opción de archivo para editar un archivo de metadatos, se deberá primero, seleccionar un archivo bien en el desplegable de “**Nuevo archivo**” y luego seleccionando “**Abrir archivo**” o bien creándolo con el botón de “**Abrir archivo**”, una vez hayamos

editado el archivo correspondiente deberemos guardarlo con el botón de “**Guardar archivo**” o eliminarlo con el de “**Eliminar archivo**”.

Al haber elegido el archivo que se quiere abrir o se haya creado este, se nos mostrará su contenido en una columna en la parte derecha de nuestra pantalla, este contenido variará según si es un archivo de **subtítulos** (lapso temporal, descripción) o de **metadatos** (lapso temporal, latitud, longitud, descripción), para editar estos archivos tendremos en la parte derecha unos formularios para que el usuario edite el campo correspondiente según el tipo de archivo que tenga abierto.

Indistintamente del archivo que se tenga abierto, el usuario tendrá los mismo tres botones, “**Inicio de segmento**” para indicar donde empieza el segmento, “**Guardar segmento**” para indicar donde finaliza este segmento y hacer que se guarde en los metadatos de la columna de la derecha o “**Eliminar segmento**” en el caso que se quiera eliminar el segmento en cuestión.

Por último tendríamos la propiedad del web-responsive, la cual explicaremos en detalle más adelante.

Explicación técnica del proceso de codificación y publicación del material multimedia

Para todo el proceso de codificación y edición de vídeos de la práctica hemos usado **FFmpeg** una colección de softwares libre desarrollada en GNU/Linux pero compatible en la mayoría de sistemas operativos, este *framework* es capaz de hacer: **decode, encode, transcode, mux, demux, stream, filtrar y ejecutar** casi cualquier tipo de media además de ser una herramienta muy potente, es también extremadamente portable.

Para codificar el video se ha utilizado **mp4** y **webm** ya que son los dos codecs que más navegadores soportan. Para el audio se ha utilizado **mp3** que es soportado por casi todos los navegadores más utilizados hoy en día, exceptuando Opera que es a partir de la versión 15.

Navegador	WEBM	MP4	MP3
Firefox	✓	✓	✓
Safari	✗	✓	✓
Google Chrome	✓	✓	✓
Opera	✓	✓	?
Internet Explorer	✗	✓	✓

Para llevar a cabo esta práctica hemos tenido que usar la herramienta anterior con el fin de convertir nuestro vídeo de japon a un formato más aceptable, en un principio, el vídeo tenia audio, 4k, 60Hz, formato webm y un mensaje de suscribirse en los últimos 10 segundos, por todo esto hemos aplicado los siguientes comandos.

Para convertir el vídeo de webm a mp4:

```
ffmpeg -i input.webm output.mp4
```

Para pasar el video de 60 a 24Hz:

```
ffmpeg -i input.mp4 -r 24 output.mp4
```

Establecer calidad de vídeo a 1080p:

```
ffmpeg -i input.mp4 -s 1980x1080 output.mp4
```

Para quitar la pista de audio del vídeo:

```
ffmpeg -i input.mp4 -map 0:v -c copy output.mp4
```

Para quitar los últimos segundos del vídeo donde aparece el mensaje de suscribirse:

```
ffmpeg -i input.mp4 -ss 00:00:00 -t 00:05:11 output.mp4
```

Para añadir la pista de audio del vídeo:

```
ffmpeg -i input.mp4 -i input.mp3 -c copy -map 0:v:0 -map 1:a:0 output.mp4
```

Para retrasar la pista de audio del vídeo 2 segundos:

```
ffmpeg -i input.mp4 -itsoffset 2 -i input.mp4 -c:a copy -c:v copy -map 0:v:0  
-map 1:a:0 output.mp4
```

Para quitar los primeros segundos del vídeo donde el video no reproduce:

```
ffmpeg -i input.mp4 -ss 00:00:02 -t 00:05:13 output.mp4
```

Para convertir el vídeo de mp4 a webm:

```
ffmpeg -i input.mp4 output.webm
```

Tras aplicar estos comandos nos quedamos con la versión final del vídeo la cual hemos usado para trabajar en el código fuente y hemos subido al servidor proporcionado en clase.

Explicación técnica de funcionalidades desarrolladas

Se han desarrollado diferentes funcionalidades mediante el uso de varios lenguajes, siendo estos han sido **JavaScript**, **Ajax** y **PHP**.

Reproductor

En esta sección describiremos brevemente las funcionalidades referentes al reproductor y su código fuente.

Iniciar el reproductor:

Primeramente hemos de cargar el vídeo y sus tracks, una vez los tengamos realizaremos una búsqueda hasta encontrar los metadata, al encontrarlos los ocultaremos y añadiremos un **EventListener** para que aparezcan cuando sea oportuno.

```
function reproductor_start(){
    vid = document.getElementsByTagName("video")[0];
    tracks = vid.textTracks;

    var i = 0;
    while(tracks[i].kind != 'metadata'){
        i++;
    }
    tracks[i].mode = "hidden";
    tracks[i].addEventListener("cuechange", metadata);
}
```

Reproductor.js 12-22

Reproducir Audio:

Si el usuario quiere mutear o reproducir el audio el programa cambiará el estado del video a muteado o no, además de cambiar el valor del botón.

```
function togglemute(){
    if(vid.muted){
        vid.muted = false;
        document.getElementById("auEsp").value = "Con Volumen";
    } else {
        vid.muted = true;
        document.getElementById("auEsp").value = "Sin Volumen";
    }
}
```

Reproductor.js 68-76

Ocultar/mostrar subtítulos:

En esta función gestionamos el tema de mostrar u ocultar subtítulos, el primer **if** es el caso de pulsar el mismo idioma de subtítulo que ya está activo, en este caso se ocultan todas las pistas de subtítulos, en el caso de ser otro idioma distinto al activo se muestran todas las pistas de subtítulo del lenguaje indicado y se ocultan todas las demás.

```
function subtítulos(leng){
  if (sub == leng){
    sub = null;
    for (var i = 0; i < tracks.length; i++) {
      var track = tracks[i];
      if (track.kind === 'subtitles') {
        track.mode = 'hidden';
      }
    }
  } else {
    for (var i = 0; i < tracks.length; i++) {
      var track = tracks[i];

      // Find the English captions track and mark it as "showing".
      if (track.kind === 'subtitles' && track.language === leng) {
        track.mode = 'showing';
      } else if (track.kind === 'subtitles') {
        track.mode = 'hidden';
      }
    }
    sub = leng;
  }
}
```

Reproductor.js 43-66

Carga de metadatos:

Esta función se encarga de cargar la primera **cue** y luego analizar gramaticalmente texto mediante un **JSON** y finalmente cargar estos datos.

```
function metadata(){
  var cue = this.activeCues[0];
  var obj = JSON.parse(cue.text);
  document.getElementById("descrip").innerHTML = obj['descripcion'];
  setmap(obj['latitude'], obj['longitude'])
}
```

Reproductor.js 24-29

A continuación pasaremos al archivo **player.js** el cual es usado por reproductor.js.

Iniciar player:

Creamos un contenedor con el player y le añadimos los **addEventListener** que describiremos a continuación así como cargar los diferentes componentes (vídeo, barra, svg) y los botones de pantalla.

```
function player_start() {  
    player(contenedor);  
  
    vid = document.getElementsByTagName("video")[0];  
    vid.addEventListener("timeupdate", onTimeChange);  
    vid.addEventListener("durationchange", onDurationChange);  
    vid.addEventListener("ended", updatePlay);  
  
    bar = document.getElementById("bar");  
    bar.addEventListener("click", onProgressBarClick);  
    bar.addEventListener("mousemove", onProgressBarHover);  
  
    window.addEventListener("resize", screenButtons);  
  
    svg = document.getElementById("svg");  
    svg.addEventListener("mousemove", showControls);  
  
    screenButtons();  
}
```

Player.js 15-34

Actualización de la barra de progreso:

Si el vídeo está activo, en el caso de tratarse de un cambio de duración, actualizamos la duración y el tamaño de la barra, si es un cambio de tiempo, actualizamos el tiempo actual y el punto actual de la barra.

```

// Cuando la duracion del video cambia, este se actualiza como el maximo de la
// barra de progreso.
var onDurationChange = function(){
    if(vid.readyState){
        document.getElementById("duration").innerHTML = time(vid.duration);
        document.getElementById("bar").max = vid.duration;
    }
}

// Cuando el tiempo actual del video cambia, este se actualiza junto con la
// barra de progreso.
var onTimeChange = function(){
    if(vid.readyState){
        document.getElementById("current").innerHTML = time(vid.currentTime);
        document.getElementById("bar").value = Math.round(vid.currentTime);
    }
}

```

Player.js 51-67

Botones del player:

Para esto usamos tres funciones, **screenButtons()** coloca y centra los botones de play, avanzar y retroceder, la segunda función **showControls()** muestra u oculta los controles al pasar un intervalo de 5 segundos y por último **updatePlay()** cambia el estado del vídeo según este pausado o no.

```
function screenButtons(){
    var svgY = svg.getBoundingClientRect().bottom;
    var svgX = svg.getBoundingClientRect().right;

    document.getElementById("play").innerHTML = '<path class="play" fill="black" d="M ' +
    (svgX/2 - 10) + ' ' + (svgY/2 - 15) + ' v 30 l 25 -15 Z"/> <path class="pause" stroke-
    width="12" stroke="black" d="M ' + (svgX/2 - 10) + ' ' + (svgY/2 - 15) + ' v 30 m 16 -30 v
    30"/><circle stroke="black" fill="transparent" fill-opacity="0" stroke-width="12" cx="' +
    svgX/2 + '" cy="' + svgY/2 + '" r="' + 40 + '" onclick="toggleVideo();"/> ';

    document.getElementById("foward").innerHTML = '<path fill="black" d="M ' + (svgX*2/3 + 1)
    + ' ' + (svgY/2 - 8) + ' v16 l13 -8 z"/><path fill="black" d="M ' + (svgX*2/3 - 11) + ' '
    + (svgY/2 - 8) + ' v16 l13 -8 z"/><circle stroke="black" fill="transparent" fill-
    opacity="0" stroke-width="6" cx="' + svgX*2/3 + '" cy="' + svgY/2 + '" r="' + 20 + '"
    onclick="fowardsVideo();"/> ';

    document.getElementById("back").innerHTML = '<path fill="black" d="M ' + (svgX/3 - 1) + '
    ' + (svgY/2 - 8) + ' v16 l-13 -8 z"/><path fill="black" d="M ' + (svgX/3 + 11) + ' ' +
    (svgY/2 - 8) + ' v16 l-13 -8 z"/><circle stroke="black" fill="transparent" fill-
    opacity="0" stroke-width="6" cx="' + svgX/3 + '" cy="' + svgY/2 + '" r="' + 20 + '"
    onclick="backVideo();"/> ';

    updatePlay();
}

// Muestra u oculta los botones interactivos dentro de la pantalla. A los 5
// segundos de inactividad del raton los oculta.
function showControls() {
    document.getElementById("play").style.display = "block";
    document.getElementById("foward").style.display = "block";
    document.getElementById("back").style.display = "block";

    clearTimeout(timer);
    timer = setTimeout(() => {
        document.getElementById("play").style.display = "none";
        document.getElementById("foward").style.display = "none";
        document.getElementById("back").style.display = "none";
    }, 5000);
}

// Cambia el estado del boton play segun si el video esta pausado o no.
function updatePlay() {
    document.querySelector("#play .play").style.display = isPlaying(vid) ? "none" : "block";
    document.querySelector("#play .pause").style.display = !isPlaying(vid) ? "none" :
    "block";
}
```

Player.js 103-137

Avanzar/retroceder el player:

Para esto usamos dos funciones, la primera suma 5 segundos al tiempo actual del vídeo y lo aplica si este no supera la duración del vídeo, y de forma similar la segunda función resta 5 segundos y lo aplica de ser este mayor a 0.

```
// Avanza el video 5 segundos si no puede porque es mayor que la duracion,
// impone esta ultima.
function forwardsVideo() {
    var t = vid.currentTime + 5;
    if(t <= vid.duration){
        vid.currentTime = t;
    } else {
        vid.currentTime = vid.duration;
    }
}

// Retrocede el video 5 segundos si no puede porque es menor que 0, impone este
// ultimo.
function backVideo() {
    var t = vid.currentTime - 5;
    if(t >= 0){
        vid.currentTime = t;
    } else {
        vid.currentTime = 0;
    }
}
```

Player.js 157-177

Editor

En el editor hay funcionalidades dirigidas a archivos en el servidor y modificación de estos, estas se han hecho de la siguiente forma:

Seleccionar archivo:

Lo primero que hace el programa es buscar que archivos hay en la carpeta de video con la extensión **.vtt**, para ello se realiza una llamada y espera a que al archivo **dir.php** retorne un **Json** con los archivos que respeten este criterio.

```
// Muestra los archivos vtt que hay en el servidor.
function dirVTT(){
    $.ajax({
        type: "POST",
        url: '../backend/dir.php',
        async: false,
        success: function(data){
            var dir = JSON.parse(data);
            var s = '<option value="-"> nuevo archivo </option>';
            for (var i = 0; i < dir.length; i++){
                s += '<option value="' + dir[i] + '">' + dir[i] + '</option>'
            }
            document.getElementById("selectvtt").innerHTML = s;
        }
    });
}
}
```

Editor.js 181-196

Para ello el servidor realizará un **scandir** en la carpeta de **video** y recorrerá el resultado buscando archivos que contengan la casena **vtt**. Una vez obtenidos, los codifica como **Json** y devolverá este resultado a la llamada.

```
<?php
$dir = scandir('/var/www/html/video/');
$res = array();
for($i = 2; $i < count($dir); ++$i){
    if(strstr($dir[$i], "vtt")){
        array_push($res,$dir[$i]);
    }
}
echo(json_encode($res));
?>
```

Dir.php 1-12

Una vez obtenido el **Json**, este lo deshace en un array y monta las opciones en el menú de **drop-down**.

Abrir un archivo:

A la hora de abrir un archivo, este obtiene el nombre del archivo del **drop-down** menu, si no hay ningún archivo seleccionado este creará un archivo, en este caso, saldrá una ventana emergente en la cual se solicitará el nombre, si hay un archivo seleccionado, abrirá este.

```
// Abre un archivo o crea uno si no hay seleccionados.
function openVTT(){
    file = document.getElementById("selectvtt").value;
    if(file == '-'){
        file = window.prompt('Nombre del archivo: (añade _meta o _idioma)', 'Nombre');
        if(file != null){
            file = file + '.vtt';
            writeVTT(file, 'WEBVTT');
            dirVTT();
            $('#selectvtt option:contains(' + file + ')').prop({selected: true});
        }
    }
    if(file != null){
        document.getElementById("filename").innerHTML = 'Contenido del archivo ' + file;
        type = file.includes("meta");
        readVTT(file);
        setControls();
    }
}
```

Editor.js 133-151

A la hora de crear un archivo nuevo, este llama al archivo **write.php**, el cual se encargará de crear este e incluir la cabecera **WEBVTT**, y actualiza el contenido del **drop-down** menu el cual mostrará el archivo nuevo como seleccionado.

```
<?php
if (isset($_POST["name"]) && isset($_POST["data"])) {
    $file = fopen("/var/www/html/video/" . $_POST['name'], "w") or die("Unable to open file!");
    fwrite($file, $_POST['data']);
    fclose($file);
}
?>
```

Write.php 1-9

Una vez creado el archivo o bien si ya existía se procede a abrir. Para ello se llama a la función de **readVTT** la cual indica de qué tipo de archivo se trata, si es de **metadatos** o es un **subtítulos** ya que los controles dependen de esto y procede a realizar la llamada a **read.php** que devolverá el contenido en **Json**.

```
// Lee archivo lo guarda en la variable json y muestra por el div show.
function readVTT(name){
    $.ajax({
        type: "POST",
        url: '../backend/read.php',
        data: {"name":name},
        async: false,
        success: function(data){
            json = JSON.parse(data);
        }
    });
    document.getElementById("show").innerHTML = showVTT();
}
```

Editor.js 206-217

Una vez haya devuelto el contenido este se mostrará por pantalla y se guardará en memoria dinámica para que se modifique sin acceder constantemente al servidor. Por último el programa montará los controles adecuados para realizar la modificación de este archivo mediante la función **setControls()**.

Guardar un archivo:

Al guardar un archivo el sistema lo que hace es guardar el **Json** en el archivo pertinente. para ello llama a la función **saveVTT** que confirma que haya un archivo abierto y guarda el **Json** en formato **VTT** en el archivo pertinente. el código se ha explicado anteriormente en **abrir un archivo**.

Eliminar un archivo:

Para eliminar un archivo este se obtiene del elemento del drop-down el archivo a eliminar, una vez confirmado que el usuario quiere eliminarlo mediante el popup. se llama a delete.php.

```
// Elimina el archivo seleccionado en el dropdown.
function deleteVTT(){
    var rem = document.getElementById("selectvtt").value;
    if(rem != '-'){
        if(window.confirm('Seguro que quieres eliminar el archivo ' + rem + '?')){
            $.ajax({
                type: "POST",
                url: '../backend/delete.php',
                async: false,
                data: {"file": rem},
                success: function(value){
                    if(value){
                        window.alert("Archivo eliminado");
                        dirVTT();
                        json = undefined;
                        document.getElementById("show").innerHTML = '';
                        document.getElementById("filename").innerHTML = 'Contenido del archivo';
                    } else {
                        window.alert("No se pudo realizar la operacion");
                    }
                }
            });
        } else {
            window.alert("Se cancelo la operacion del archivo");
        }
    }
}
}
```

Editor.js 153-179

Una vez hecha la llamada se elimina el archivo indicado mediante **unlink** y se devuelve el resultado de la operación.

```
<?php
    if(isset($_POST["file"])){
        echo(unlink("/var/www/html/video/" . $_POST["file"]));
    }
?>
```

Delete.php 1-5

Con lo que se ha devuelto si ha sido hecho correctamente se reinicia la ventana con los parámetros por defecto para continuar la edición.

Eliminar Segmento:

Para eliminar un segmento en el archivo **VTT** este llama a la función **deleteSegment** la cual realizará la eliminación del segmento indicado por el tiempo de inicio y final.

```
var i = 0;
while(!found && i < json.length){
    tji = getsecondsVTT(json[i]["inicio"]);
    if(tji >= tsi){
        found = true;
    } else {
        i++;
    }
}
if (found){
    if(tsf > tji){
        tjf = getsecondsVTT(json[i]["fin"]);
        if(tsf >= tjf){
            json.splice(i, 1);
            while(i < json.length && tsf >= getsecondsVTT(json[i]["fin"])){
                json.splice(i, 1);
            }
            if(i < json.length){
                tji = getsecondsVTT(json[i]["inicio"]);
                if(tsf > tji){
                    lapedend = true;
                }
            }
        } else {
            lapedend = true;
        }
    }
}
if(i > 0){
    tjf = getsecondsVTT(json[i-1]["fin"]);
    if(tsi <= tjf){
        lapedstart = true;
    }
}
if(lapedstart){
    json[i-1]["fin"] = "00:" + tini.value + ".000";
}
if(lapedend){
    json[i]["inicio"] = "00:" + tfin.value + ".001";
}
return i;
}
```

Editor.js 90-131

Como se ve en el código anterior, lo primero que se hace es una búsqueda de la posición donde debería ir el segmento mediante comparando el inicio de este con los del **Json**. Si se ha encontrado un segmento mayor que el que se quiere poner entonces procede a ir eliminando todos los segmentos entre el inicial y final que entran completamente dentro del segmento a eliminar.

Una vez que se ha hecho esto si aun hay un segmento que el inicio se encuentra en el segmento pero el final no se guarda que hay que modificar este para más tarde.

Por último se revisa si se solapa el inicio de este segmento con el final del anterior en el **Json**. si es así se guarda el estado.

Por último si se solapa por arriba o abajo se modifican los segmentos para que el segmento indicado haya sido eliminado.

Añadir Segmento:

Para añadir un segmento en los archivos **VTT** primero hay que iniciar el segmento, esto lo que hace es recoge donde se encuentra el cursor del video, guarda como inicio del segmento y inicia un **Listener** para el final del segmento que ira modificando con el video.

```
// Inicia el temporizador del segmento.  
function newSegment(){  
  tini.value = time(vid.currentTime);  
  tfin.value = time(vid.currentTime);  
  vid.addEventListener("timeupdate", getEndSegmentTime);  
}
```

Editor.js 23-27

Una vez hecho esto si el usuario pulsa el botón de guardar segmento realizará una búsqueda en el **Json** donde debería ir este segmento mediante la función **deleteSegment**.

```

// Añade un nuevo segmento en el json.
function addSegment(){
    pause();
    if(tini.value < tfin.value){
        if(type){
            if(tlat.value != "" && tlon.value != "" && tdes.value != "") {
                var i = deleteSegment();

                var ti = "00:" + tini.value + ".001";
                var tf = "00:" + tfin.value + ".000";
                var tl = tlat.value;
                var tg = tlon.value;
                var td = "'" + tdes.value + "'";
                json.splice(i, 0, {inicio:ti, fin:tf, latitud:tl, longitud:tg, descripcion:td});
                document.getElementById("show").innerHTML = showVTT();
            } else {
                window.alert("Introduce metadatos.");
            }
        }
        }else{
            if(ttex.value != ""){
                var i = deleteSegment();

                var ti = "00:" + tini.value + ".001";
                var tf = "00:" + tfin.value + ".000";
                var tt = ttex.value;
                json.splice(i, 0, {inicio:ti, fin:tf, texto:tt});
                document.getElementById("show").innerHTML = showVTT();
            } else {
                window.alert("Introduce subtítulo.");
            }
        }
    }
    }else{
        window.alert("El inicio no puede ser mayor o igual que el final.")
    }
}
}

```

Editor.js 34-70

Una vez hecho espacio mediante esta función, según si es un archivo de **metadatos** o bien de **subtítulos** se introducirá la información en el sitio pertinente dentro del **Json**.

Web-responsive

Para elaborar la web, no se ha hecho uso de ningún tipo de plantilla proporcionada por bootstrap de forma que todos los aspectos relacionados con el diseño se han creado manualmente, incluyendo también que la página sea responsive y se adapte a diferentes dispositivos.

Para que la web sea responsive se ha hecho mediante las hojas de estilos css (tanto del reproductor como el editor), en los cuales se ha aplicado diferentes tamaños a los elementos que forman la web. También se ha tenido en cuenta el tamaño de la pantalla en píxeles del dispositivo para cambiar la proporción de los elementos en caso de que la pantalla sea de un móvil u ordenador.

Esto ha sido posible usando la css grid, dividiendo la pantalla en una tabla con ciertas columnas y filas de cierto tamaño según el contenido que tengan que acomodar, cada hojas de estilo de css mencionadas anteriormente tienen una css grid que se activará según el tamaño de la pantalla, creando efectivamente un css grid con diferente distribución para cada dispositivo y tamaño que hemos definido.

En el ejemplo de abajo podemos observar el .grid para la versión de escritorio.

```
@media screen and (min-width: 736px) {  
  .grid {  
    display: grid;  
    grid-template-columns: 1fr 1fr;  
    grid-template-rows: 1fr;  
    gap: 0px 0px;  
    grid-auto-flow: row;  
    grid-template-areas: "navbar navbar" "controlplayer map" "controlplayer text" "footer footer";  
  }  
  .footer {  
    grid-area: footer;  
    position: relative;  
    bottom: 0;  
    margin-top: 1.8%;  
  }  
  .map {  
    grid-area: map;  
    margin-top: 40px;  
    margin-left: 40px;  
    display: block;  
  }  
}
```

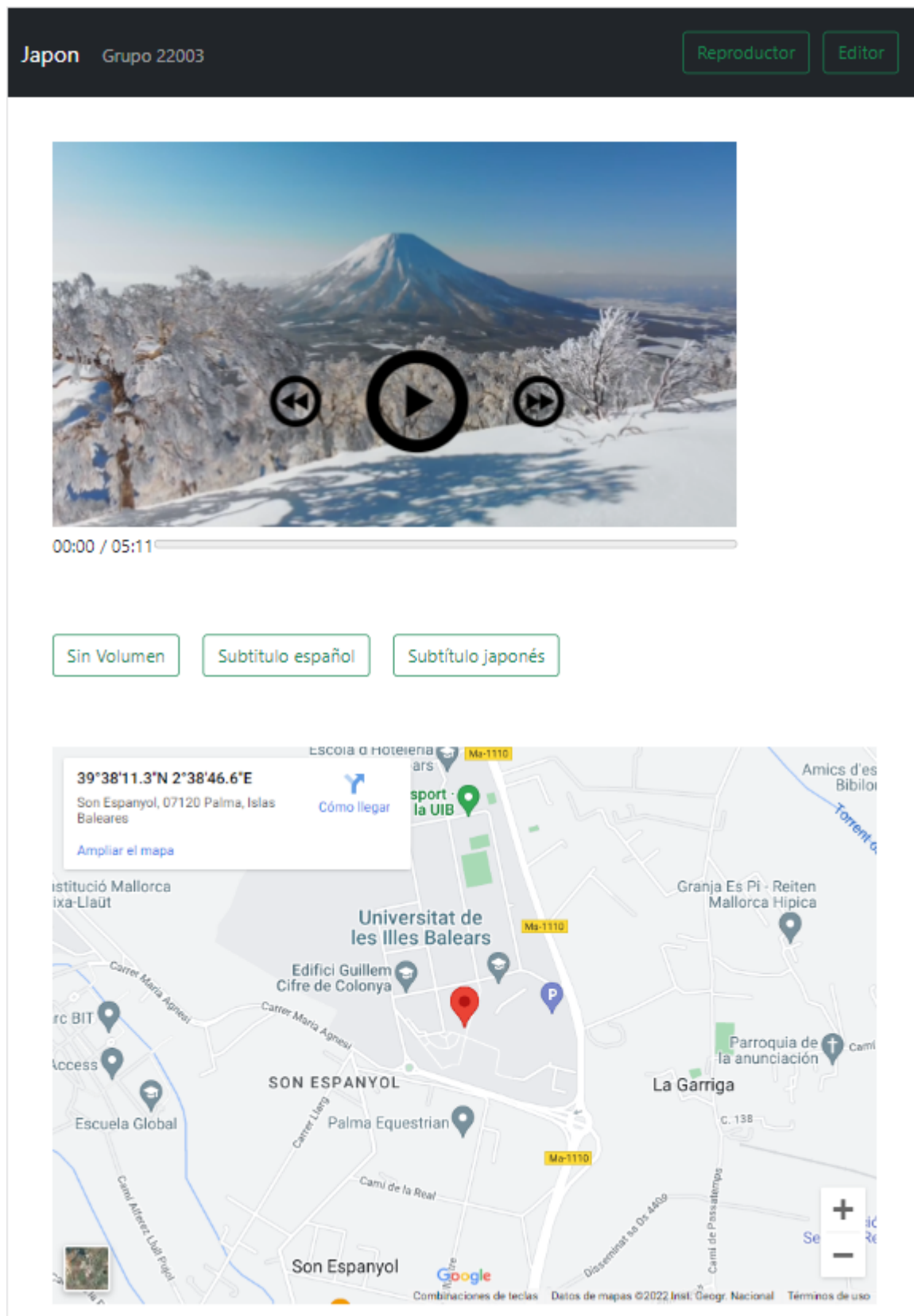
Se ha comprobado la visualización de esta en las siguientes plataformas y navegadores:

Navegador	Correctitud
Firefox	✓
Safari	✓(*)
Google Chrome	✓
Opera	✓
Internet Explorer	✓

(*) En Safari la barra de reproducción se desliza junto con el video quedando en el centro del contenedor el video y la barra al final de este, sin embargo la funcionalidad de estos es la pertinente.


Versión tablet:

Podemos ver que el reproductor se pone en la parte superior con el mapa debajo, todas las funcionalidades del reproductor van correctamente en el dispositivo.



Para el editor en la tableta se ve igual que en la versión de escritorio y mantiene las mismas funcionalidades.

Japon Grupo 22003 Reproductor Editor



00:00 / 05:11

Inicio segmento:

Fin segmento:

Latitud:

Longitud:

Descripción:

Inicio Segmento Guardar Segmento Eliminar Segmento

japon_meta.vtt ▼

Abrir Archivo

Guardar Archivo Eliminar Archivo

Contenido del archivo japon_meta.vtt

00:00:00.001 --> 00:00:08.000
latitud: 35.360625
longitud: 138.7273634
descripcion: "El monte Fuji es el pico más alto de la isla de Honshu y de todo Japón, con 3776 metros de altitud. Se encuentra entre las prefecturas de Shizuoka y Yamanashi en el Japón central y justo al oeste de Tokio, desde donde se puede observar en un día despejado. El Fuji es un estratovolcán y es el símbolo de Japón. Considerado sagrado desde la Antigüedad, les estaba prohibido a las mujeres llegar a la cima hasta la era Meiji (finales del siglo xix). Actualmente es un conocido destino turístico, así como un destino popular para practicar el alpinismo. La temporada «oficial» para practicar el alpinismo dura desde principios de julio hasta finales de agosto. Son mayoría los que escalan por la noche para apreciar la salida del Sol."

00:00:08.001 --> 00:00:19.000
latitud: 33.8455768
longitud: 132.7655346
descripcion: "El Castillo de Matsuyama es un castillo japonés que se encuentra en el centro de la Ciudad de Matsuyama de la prefectura de Ehime, en la cima del Monte Katsu También es conocido como Castillo de Kinki o como Castillo de Katsuyama. Consta de un cuerpo principal comunicado al cuerpo secundario sumiyagura sur sumiyagura sur a través de pasillos. Por ello se trata de un castillo de cuerpos interconectados que junto a los castillos de Himeji y Wakayama conforman los tres principales castillos de este tipo de distribución."


00:00:19.001 --> 00:00:24.000
latitud: 36.7578079
longitud: 139.5971689
descripcion: "El Santuario de Toshogu es un santuario sintoísta que se encuentra en Nikko. Forma parte del conjunto de los «Santuarios y templos de

Versión Móvil:

En el móvil, el editor si se pone de forma horizontal sale igual que en el escritorio, en vertical se mueven los elementos a uno debajo del otro, todo el rato mantiene la misma funcionalidad que en la versión de escritorio, el unico problema seria los controles del video que se desplazan y ya no están centrados.

Japon Grupo 22003

Reproductor Editor



00:00 / 05:11

Inicio segmento:

japon_meta.vtt

Abrir Archivo

Guardar Archivo

Eliminar Archivo

Contenido del archivo japon_meta.vtt

00:00:00.001 --> 00:00:08.000
latitud: 35.360625
longitud: 138.7273634
descripcion: "El monte Fuji es el pico más alto de la isla de Honshu y de todo Japón. con 3776 metros de

Inicio segmento:

Fin segmento:

Latitud:

Longitud:

Descripcion:


Inicio Segmento Guardar Segmento

descripcion: "El monte Fuji es el pico más alto de la isla de Honshu y de todo Japón, con 3776 metros de altitud. Se encuentra entre las prefecturas de Shizuoka y Yamanashi en el Japón central y justo al oeste de Tokio, desde donde se puede observar en un día despejado. El Fuji es un estratovolcán y es el símbolo de Japón. Considerado sagrado desde la Antigüedad, les estaba prohibido a las mujeres llegar a la cima hasta la era Meiji (finales del siglo XIX). Actualmente es un conocido destino turístico, así como un destino popular para practicar el alpinismo. La temporada «oficial» para practicar el alpinismo dura desde principios de julio hasta finales de agosto. Son mayoría los que escalan por la noche para apreciar la salida del Sol."

00:00:08.001 --> 00:00:19.000
latitud: 33.8455768

En el reproductor el video se pone en la parte superior y los elementos debajo de este, de la misma manera que se ha explicado anteriormente los controles del video se desplazan y no quedan centrados con el video, sin embargo las funcionalidades se mantienen.

Japon Grupo 22003 Reproductor Editor



00:00 / 05:11

Inicio segmento:

Fin segmento:

Latitud:

Longitud:

Descripcion:


Inicio Segmento Guardar Segmento

Eliminar Segmento

japon_meta.vtt ▾

Abrir Archivo

Japon Grupo 22003 Reproductor Editor



00:00 / 05:11

Sin Volumen

Subtitulo español


Subtítulo japonés

Escola d'Hoteleria

Ma-1110

39°38'11.3"N 2°38'...

[Ampliar el mapa](#)



Problemas y soluciones adoptadas durante la práctica

A la hora de pensar que codificadores utilizar pensamos en mp4 y ogg, sin embargo la calidad de ogg dejaba mucho que desear así que se optó por utilizar webm como codificación adicional a mp4.

Para el servidor hemos encontrado funciones que no son válidas (**str_contains**, **str_ends_with**) por lo cual, las hemos tenido que cambiar por (**str_str**).

Otras de las dificultades que nos hemos encontrado ha sido el hecho de hacer responsive por nosotros mismos la web, esto debido a que hay ciertos estilos entre las diferentes hojas y por eso a veces se sobreponen uno sobre otro. Además hemos tenido que tener en cuenta que cada cambio realizado en el tamaño de los divs o los márgenes también afectan a la visualización en otros dispositivos, por lo cual hemos buscado soluciones intermedias.

También hemos tenido problemas con el Header y el Footer por el planteamiento inicial que tuvimos de la página. Inicialmente, habíamos pensado crear dos documentos HTML que correspondan al footer y header respectivamente, de forma que los dos sean llamados desde la hojas Editor.html y Reproductor.html. De esta forma los dos compartirán estos documentos y los cambios realizados tanto en el header como el footer se verían reflejados en el editor y reproductor sin tener que hacer dos veces. El problema que tuvimos consiste en que a la hora de cambiar de página entre el reproductor o editor mediante el navegador, lo que hacía era cambiar el div correspondiente al navegador e insertar ahí la página seleccionada. Para solucionarlo, llegamos a la conclusión de que sería más sencillo insertar el navegador y el footer en los dos sitios y asegurarnos realizar siempre los mismo cambios. Al final, esta segunda forma ha sido más ventajosa ya que como las dos ventanas del reproductor como del editor presentan elementos diferentes, se han podido adoptar individualmente a cada página.

Además, también hemos tenido algunos problemas con situar correctamente el navegador y el footer, tanto en la web vista desde un ordenador como vista desde un móvil, debido a que o bien los botones de navegador no se situaban en la parte derecha, o el footer no se quedaba en la parte inferior de la pantalla.

Opinión de la práctica

En general nos ha parecido una buena primera práctica debido a que además de repasar ciertos aspectos relacionados con la web, también hemos descubierto nuevas herramientas que nos parecen bastante interesantes como es el `ffmpeg`, porque a pesar de que es un poco complicado de instalar, nos ha parecido una bastante potente y satisfactoria de utilizar, debido a que es muy fácil modificar y adaptar los vídeos con una serie de comandos si ya te sabes estos últimos.

Mejoras adicionales

Una de los aspectos a mejorar de la práctica es el aspecto visual tanto para ordenador como móviles. De forma que esta se vea un poco menos simple y darle un toque más personal. Adaptar la página para tener una mejor visualización para dispositivos tableta y móvil.

En el apartado del header añadir una sección con un drop down el cual contenga los vídeos que hay y que permita añadir nuevos sin tener que hacer cambios en el código, siendo así más dinámico a la hora de añadir metadatos nuevos.

Autoevaluación

Un aspecto que valoramos de nuestra práctica es que la mayoría de elementos que la conforman han sido creados por nosotros mismos. Nos referimos tanto la web que la hemos creado desde cero sin utilizar ningún tipo de plantilla como los audios grabados que se utilizan en los vídeos. Además, consideramos que todo lo relacionado con el back-end de la práctica está bastante bien, aunque hay algunos aspectos a mejorar como hemos visto en el apartado de “Mejoras adicionales” al igual que en el front-end. Debido a esto nos damos una nota de **7,5**.