

**Proiect Aplicatii Multimedia**  
**Video Player**

**Autor: Necula Mihai**

**An: III**

## Cuprins

Introducere .....	3
Prezentarea temei.....	3
Obiectivele proiectului .....	3
Suportul tehnic.....	4
Prezentare tehnica.....	4
Mod de utilizare .....	5
Interactiunea cu utilizatorul.....	5
Configurarea .....	6
Concluzii .....	13
Referinte bibliografice .....	14

## Introducere

Vizionarea conținutului video este una dintre cele mai populare utilizări când vine vorba de căutarea pe diverse pagini web. De-a lungul anilor, au existat o multitudine de versiuni a dispunerii videoclipurilor într-o pagină web, utilizându-se diferite mijloace sau limbaje de programare.

## Prezentarea temei

Tema pe care eu m-am decis să o implementez în cadrul cursului de Aplicații Multimedia este un player video. Am ales această temă, deoarece folosesc tot timpul diverse pagini web care dispun de un player video, fie în scop educațional, fie pentru entertainment. Un alt motiv pentru care am ales această temă este acela că mi-am propus că odată cu ocazia alegerii acestui proiect, voi învăța să lucrez cu diferite programe și tehnologii, cât și limbaje de programare pe care le-am folosit destul de puțin pe parcursul facultății. Proiectul a fost implementat în editorul de cod Visual Studio Code, utilizându-se următoarele limbaje de programare: HTML, CSS și JavaScript.

## Obiectivele proiectului

Player-ul video pe care am decis să îl realizez are stabilite următoarele obiective:

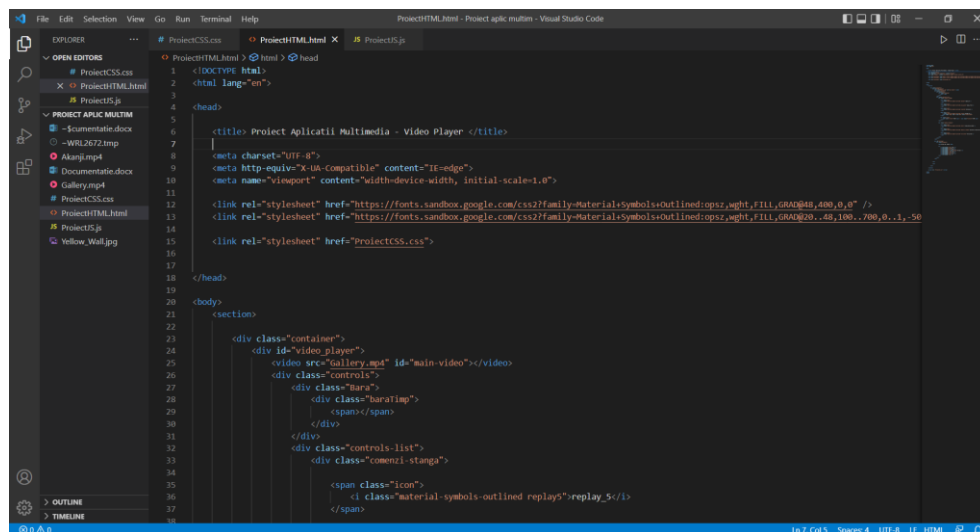
- ➔ Implementarea unui **buton de play**.
- ➔ Crearea butoanelor de **fast-rewind** și **fast-forward** cu 5 secunde.
- ➔ Un **slider de volum** ce va apărea în bara de acțiuni pentru videoclip.
- ➔ O **bara de progres** ce va contoriza progresul clip-ului video.
- ➔ O **caseta de text** ce va fi împartită în două: o parte va contoriza clip-ul derulat, iar în cealaltă se va afișa timpul total al videoclip-ului.
- ➔ **Butonul de setări**, ce va avea o singură utilitate, cea de stabilire a vitezei de redare a clipului video. Această viteză va fi setată by default pe viteza de redare normală (1x). Utilizatorul poate varia viteza începând cu 0.25x și poate ajunge până la o viteză de 2x, valorile fiind incrementate cu 0.25.
- ➔ **Redarea în modul de fundal**, ce odată ce va fi apasată, videoclipul va fi minimizat în colțul din dreapta jos al paginii web, utilizatorul având posibilitatea de a naviga pe alte pagini în timp ce vizualizează în continuare clip-ul video, volumul fiind pornit în continuare.
- ➔ Și nu în ultimul rând, obiectivul final este cel de implementare a unui **buton de fullscreen**. Odată apasat, clip-ul video va fi extins pe toată suprafața ecranului.

## Suportul tehnic

Fiecare document/pagina ce m-a ajutat in implementarea proiectului meu a fost atasata in lista de referinte bibliografice de mai jos. In primul rand, pentru a intelege utilizarea noilor limbaje de programare, am urmarit tutorialele prezente la link-urile [3], [4], [5]. Mai departe, pentru a avea un model de urmarit am urmarit in detaliu referintele [1], [7], [8]. In realizarea aspectului, cat si a comportamentului butoanelor din cadrul programului, am folosit din lista blibliografica [2], [6].

## Prezentare tehnica

Pentru a avea un program in care sa pot introduce toate fisierele pe care le voi utiliza in finalizarea proiectului intr-un workspace, am utilizat **Visual Studio Code**.



Primul pas pe care l-am facut a fost sa realizez pagina HTML. Am plasat in pagina un container, avand o sectiune unde va fi alocat spatiu pentru clip-ul video. Mai departe, am plasat butoanele in pagina, cele pentru play, fast-rewind, fast-forward, volum, redare in fundal, viteza de redare si fullscreen.

“HyperText Markup Language (HTML) este un limbaj de marcare utilizat pentru crearea paginilor web ce pot fi afișate într-un browser (sau navigator). Scopul HTML este mai degrabă prezentarea informațiilor.”

Pentru aspectul paginii si pozitionarea butoanelor precizate mai sus, am folosit CSS. Am cautat sa plasez clip-ul video intr-o pozitie cat mai centrala, pentru a fi cat mai placuta experienta folosirii acestuia de catre un utilizator, iar culorile au fost alese in asa fel incat sa fie realizata o cromatica cat mai buna intre ele.

“CSS sau Cascading Style Sheets este un standard pentru formatarea elementelor unui document HTML. Stilurile se pot atașa elementelor HTML prin intermediul unor fișiere externe sau în cadrul documentului. CSS este unul dintre tehnologiile de bază utilizate în procesul de dezvoltare web, împreună cu HTML și JavaScript.”

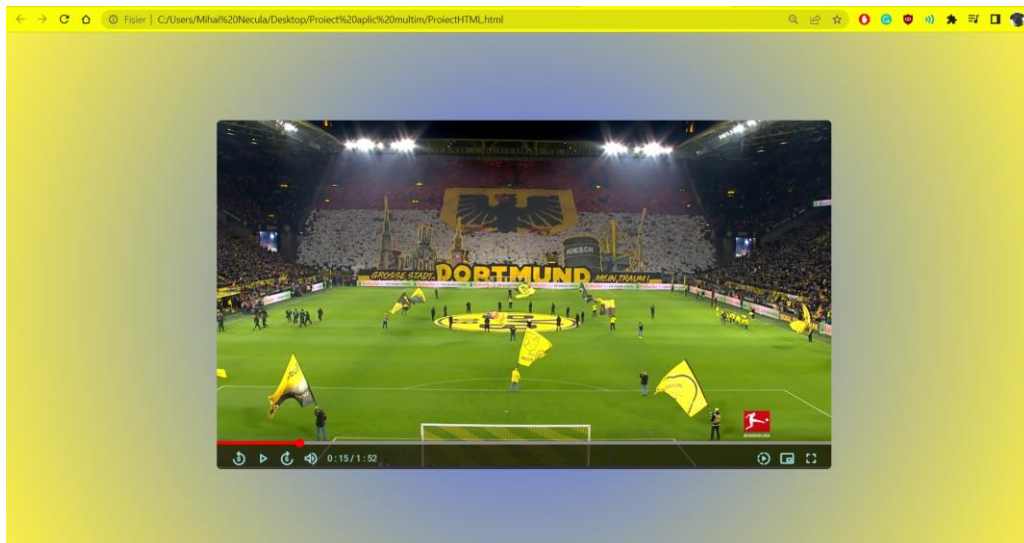
Dupa ce am realizat cele 2 programe mentionate mai sus, am inceput partea de javascript care m-a ajutat in a da utilitate butoanelor pe care le-am adaugat in cadrul player-ului video.

“Javascript este folosit mai ales pentru introducerea unor funcționalități în paginile web, codul JavaScript din aceste pagini fiind rulat de către browser. Limbajul este binecunoscut pentru folosirea sa în construirea site-urilor web, dar este folosit și pentru accesul la obiecte încapsulate în alte aplicații”.

## Mod de utilizare

### Interactiunea cu utilizatorul

In momentul in care utilizatorul va deschide pagina web, urmatoarea pagina va fi afisata:



Interactiunea acestuia va fi una destul de simpla, deoarece am folosit butoane sugestive ce vor permite utilizatorului intelegerea imediata. La fel ca in cod, vom separa comenzile in 2 parti: stanga si dreapta.

În partea din stanga vom avea urmatoarele butoane disponibile si in imaginea de mai jos. Primul buton este cel de fast-rewind, ce va derula clipul video cu 5 secunde inapoi, al doilea buton este cel de play, cat si de pause, acesta avand o utilitate dubla. Urmatorul este cel de fast-forward ce va permite utilizatorului derularea videoclipului cu 5 secunde inainte. Urmatoarea iconita este cea de reglare a volumului, in functie de dorintele utilizatorului, iar ultimul element prezent in partea stanga a player-ului video este cel de calculare a timpului scurs din videoclip, cat si timpul total al clipului.



În partea dreapta, avem butonul de viteza de redare, acesta fiind primul din poza atasata mai jos. În urma apăsării acestuia, pagina va afișa o altă fereastră mică din care utilizatorul își poate selecta viteza de redare dorită. Cea de-a doua pictogramă este cea de rulare în fundal. În urma apăsării acesteia, clipul va rula în colțul din dreapta jos al browser-ului. Ultimul buton este cel de fullscreen, ce permite vizualizarea programului într-o rezoluție mai mare.



În poza de mai jos avem prezentată bara de rulare, ce ne permite să vedem în timp real evoluția videoclipului din punct de vedere al timpului rulat până în acel moment.



## Configurarea

În cadrul configurării, se vor prezenta secvențele de cod pentru fiecare dintre cele 3 programe.

Pentru HTML, am început prin a pune într-o secțiune elementele utilizând atributul de clasă. În cadrul celui de container și video\_player am plasat celelalte butoane, mai târziu, urmând să fie împartite în comenzi de dreapta și de stnga.

```
<div class="container">
  <div id="video_player">
```

Atributul de clasă este adesea folosit pentru a indica un nume de clasă într-un "stylesheet". De asemenea, poate fi folosit de un JavaScript pentru a accesa și manipula elemente cu numele de clasă specific.

```
<div class="comenzi-stanga">
  <span class="icon">
    <i class="material-symbols-outlined replay5">replay_5</i>
  </span>

  <span class="icon">
    <i class="material-symbols-outlined play_pause">play_arrow</i>
  </span>
```

```

<span class="icon">
|   <i class="material-symbols-outlined forward5">forward_5</i>
</span>

<span class="icon">
|   <i class="material-symbols-outlined volume">volume_up</i>
|   <input type="range" min="0" max="100" class="volume_range">
</span>

<div class="timer">
|   <span class="current">0:00</span> / <span class="duration">0:00</span>
</div>

```

Mai sus am atasat 2 poze ce definesc fiecare buton, acestea fiind: replay5, play\_pause, forward5, volume, volume\_range. Numele fiecaruia a fost ales sugestiv pentru a se intelege usor utilitatea lor.

```

<div class="comenzi-dreapta">

  <span class="icon">
|   <i class="material-symbols-outlined setari">slow_motion_video</i>
</span>

  <span class="icon">
|   <i class="material-symbols-outlined redare_in_fundal">picture_in_picture_alt</i>
</span>

  <span class="icon">
|   <i class="material-symbols-outlined fullscreen">fullscreen</i>
</span>

```

Aici sunt prezentate comenzile de dreapta: setari – pentru a stabili viteza de redare, redare\_in\_fundal si fullscreen.

```

<span>Viteza de redare</span>

<ul>
  <li data-speed="0.25">0.25</li>
  <li data-speed="0.5">0.5</li>
  <li data-speed="0.75">0.75</li>
  <li data-speed="1" class="active">Normal</li>
  <li data-speed="1.25">1.25</li>
  <li data-speed="1.5">1.5</li>
  <li data-speed="1.75">1.75</li>
  <li data-speed="2">2</li>
</ul>

```

De asemenea, in urma apasarii butonului de setari (viteza de redare), pagina web va afisa o lista de la 0.25 la 2, valori incrementate cu cate 0.25, ce va permite utilizatorului modificarea vitezei de redare.

Legaturile cu celelalte 2 programe au fost facute prin urmatoarele comenzi:

```
<link rel="stylesheet" href="ProiectCSS.css">
```

```
<script src="ProiectJS.js"></script>
```

Trecand la partea de CSS, aici am inceput prin a stiliza putin pagina, si alegand un anumit font ce va urma sa fie folosit pe parcurs.

```
{  
  font-family: 'Roboto', sans-serif;  
}
```

```
body  
{  
  background: rgb(63,94,251);  
  background: radial-gradient(circle, rgba(63,94,251,1) 0%, rgba(252,240,70,1) 100%);  
}
```

Secventa de mai jos face ca utilizatorului sa-i fie indisponibila selectarea textului sau videoclipului deoarece ar pagina web ar deveni putin inestetica adaugandu-i un astfel de feature.

```
.material-symbols-outlined  
{  
  cursor: pointer;  
  user-select: none;  
  -webkit-user-select: none;  
}
```

Pentru sectiunea in care se va afla player-ul video am plasat elementele cat mai in centru.

```
section  
{  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  min-height: 100vh;  
  width: 100%;  
}
```




Pentru lista in care am organizat vitezele de redare, am folosit urmatoarele caracteristici pentru a-l face sa arate cat mai natural, incorporandu-l cat mai bine in pagina.

```
#video_player #settings .playback ul li
{
  position: relative;
  width: 100%;
  cursor: pointer;
  text-align: left;
  padding: 12px 33px;
  display: block;
  font-size: 14px;
}
```

Pentru o mai buna experienta, am ales ca slider-ul de volum sa fie vizibil doar in momentul in care cel care navigheaza tine cursorul pe iconita respectiva.

```
.controls .icon:hover .volume_range
{
  display: inline-block;
  width: 60px;
}
```

Pentru evolutia corecta a barii de rulare a videoclipului in functie de timpul scurs, am folosit urmatoarea secventa de cod:

```
.controls .Bara .baraTimp::before
{
  content: '';
  position: absolute;
  width: 14px;
  height: 14px;
  border-radius: 50%;
  right: -5px;
  top: 50%;
  transform: translateY(-50%);
  background:  rgb(255, 0, 0);
}
```

Am stabilit o pozitie absoluta a cursorului, acesta reprezentand bulina de pe bara de rulare. Printr-o multitudine de incercari, am reusit sa o plasez intr-o pozitie cat mai aproape de realitate.

In fisierul JavaScript, am cautat ca fiecare buton din cadrul player-ului video sa aiba un comportament corect, pentru a-mi indeplini obiectivele.

```
const video_player = document.querySelector('#video_player');
```

Mai intai, am inceput prin declararea unei constante in care voi avea player-ul video.

```

playxpause = video_player.querySelector('.play_pause');
videoclip = video_player.querySelector('#main-video');

function playVideo()
{
    playxpause.innerHTML = "pause";
    playxpause.title = "pause";
    video_player.classList.add('paused')
    videoclip.play();
}

function pauseVideo()
{
    playxpause.innerHTML = "play_arrow";
    playxpause.title = "play";
    video_player.classList.remove('paused')
    videoclip.pause();
}

```

Butoanele de play, respectiv pause au fost implementate prin secventele de cod de mai sus. Practic, videoclipul va avea default butonul pause apasat, utilizatorul fiind cel care va trebui sa inceapa redarea clip-ului printr-un singur click. In urma apasarii acestuia, iconita de play se va transforma intr-una de pause.

```

playxpause.addEventListener('click',()=>
{
    const isVideoPaused = video_player.classList.contains('paused');
    if(isVideoPaused == 1)
    |   pauseVideo();
    else
    |   playVideo();
})

```

Butoanele de fast-rewind si fast-forward au fost realizate in felul urmatoar:

```

// Inapoi 5 secunde
rewind = video_player.querySelector('.replay5');

rewind.addEventListener('click',()=>
{
    videoclip.currentTime = videoclip.currentTime - 5;
})

// Inainte 5 secunde

forward = video_player.querySelector('.forward5');

forward.addEventListener('click',()=>
{
    videoclip.currentTime = videoclip.currentTime + 5;
})

```

Practic, in urma apasarii acestora, pagina web va decrementa cu 5, respectiv va incrementa cu 5, durata clipului, actiunea fiind disponibila vizibil in caseta de timer.

```
// Afisam in bara de comenzi durata clipului video
videoclip.addEventListener("loadeddata", (e) =>
{
    let videoDuration = e.target.duration;
    let totalMin = Math.floor(videoDuration / 60);
    let totalSec = Math.floor(videoDuration % 60);

    if (totalSec < 10) //Daca sunt mai putin de 10 secunde, a
        totalSec = "0" + totalSec;
    totalDuration.innerHTML = `${totalMin} : ${totalSec}`;
})
```

Este calculata durata totala a clipului video.

```
// Durata clipului curent

progressArea = video_player.querySelector('.Bara');
progress_Bar = video_player.querySelector('.baraTimp');

videoclip.addEventListener('timeupdate', (e) =>
{
    let currentVideoTime = e.target.currentTime;
    let currentMin = Math.floor(currentVideoTime / 60);
    let currentSec = Math.floor(currentVideoTime % 60);

    // Daca sunt mai putin de 10 secunde, adaugam un 0 la inceput
    if (currentSec < 10)
        currentSec = "0" + currentSec;

    current.innerHTML = `${currentMin} : ${currentSec}`;

    let videoDuration = e.target.duration

    let progressWidth = (currentVideoTime / videoDuration) * 100;
    progress_Bar.style.width = `${progressWidth}%`;
})
```

Este calculat timpul parcurs de videoclip.

```
// Volumul

volume = video_player.querySelector('.volume');
volume_range = video_player.querySelector('.volume_range');

function changeVolume()
{
    videoclip.volume = volume_range.value / 100;
    if (volume_range.value == 0)
    {
        volume.innerHTML = "volume_off";
    }
    else if (volume_range.value < 40)
    {
        volume.innerHTML = "volume_down";
    }
    else
    {
        volume.innerHTML = "volume_up";
    }
}
```

Am retinut in variabilele `volume` si `volume_range` attributele stabilite in clasele HTML. In cazul in care valoarea volumul va fi setata pe 0, icoana volumul va aparea ca fiind "taiata". Daca este mai mica decat 40, va aparea o animatie specifica, iar daca este mai mare de 40, la fel va fi afisata, printr-o animatie sugestiva.

```
redare_fundal = video_player.querySelector('.redare_in_fundal');
redare_fundal.addEventListener('click',()=>
{
    videoclip.requestPictureInPicture();
})
```

Prin aceasta este posibil redarea in fundal a clipului video.

```
fullscreen = video_player.querySelector('.fullscreen');
fullscreen.addEventListener('click',()=>
{
    if (!video_player.classList.contains('openFullScreen'))
    {
        video_player.classList.add('openFullScreen');
        fullscreen.innerHTML = "fullscreen_exit";
        video_player.requestFullscreen();
    }
    else
    {
        video_player.classList.remove('openFullScreen');
        fullscreen.innerHTML = "fullscreen";
        document.exitFullscreen();
    }
});
```

In variabila `fullscreen` a fost retinut atributul din clasa HTML. Butonul are 2 scenarii, unul cand nu este apasat si asteapta semnalul utilizatorului pentru a face request paginii web pentru fullscreen, sau cand este deja in faza de fullscreen si asteapta comanda utilizatorului pentru tranzitia in modul "ecran normal".

```
let xhr = new XMLHttpRequest();
xhr.open("GET","Gallery.mp4");
xhr.responseType = "arraybuffer";
xhr.onload = (e)=>
{
    let blob = new Blob([xhr.response]);
    let url = URL.createObjectURL(blob);
    videoclip.src = url;
}
xhr.send();
```

In aceasta secventa, pagina web primeste calea catre clipul video ce se doreste a fi afisat. Proprietatea `XMLHttpRequest responseType` este o valoare de sir enumerata care specifica tipul de date continute. Raspunsul este un `ArrayBuffer` care contine date binare.

## Concluzii

În concluzie, pagina web în care am implementat a atins toate obiectivele pe care le-am stabilit la începutul documentului. În realizarea proiectului s-au folosit următoarele tehnologii: HTML, CSS, cât și JavaScript, toate acestea fiind implementate în IDE-ul Visual Studio Code. Folosind video player-ul, utilizatorul va putea: să dea play sau să pună pauză la un clip video, să deruleze cu 5 secunde înainte sau în spate, să regleze volumul în funcție de propriile dorințe, să stabilească o viteză de redare, să redea clipul video în fundal și, nu în ultimul rând, să poată maximiza videoclipul pe tot ecranul utilizând butonul de fullscreen.

## Referinte bibliografice

- <http://talkerscode.com/webtricks/create-your-own-custom-video-player-using-html5-and-javascript.php> [1]
- <https://stackoverflow.com/questions/8022425/getting-blob-data-from-xhr-request> [2]
- <https://www.w3schools.com/html/default.asp> [3]
- <https://www.w3schools.com/css/default.asp> [4]
- <https://www.w3schools.com/js/default.asp> [5]
- <https://blog.prototypr.io/css-only-multi-color-backgrounds-4d96a5569a20> [6]
- <https://freshman.tech/custom-html5-video/> [7]
- <https://www.youtube.com/watch?v=yY6XnbWnK4o&t=3s> [8]
- <https://fonts.google.com/icons> [9]
- <https://fonts.google.com> [10]
- <https://ro.wikipedia.org/wiki/JavaScript> [11]
- [https://ro.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](https://ro.wikipedia.org/wiki/Cascading_Style_Sheets) [12]