**EXPLICACIÓN DE CODIGO**

Se hizo un servidor de streaming utilizando express que crea un sistema de archivos y lo devuelve al cliente. Nuestro servidor nos permitirá seleccionar marcos de tiempo a lo largo del video y decidir qué tan grande es la carga a enviar en este caso es de 100 KB.

app.get("/", function (req, res) {

    res.sendFile(\_\_dirname + "/index.html");

});

Utilizamos un archivo con formato html para visualizar el video:

< <!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="UTF-8" />

        <meta http-equiv="X-UA-Compatible" content="IE=edge" />

        <meta name="viewport" content="width=device-width, initial-scale=1.0" />

        <title>Streaming con Node</title>

        <style>

            body {

                margin: 2% auto;

                max-width: 100%;

                background-color: rgb(250, 247, 247);

                padding-top: 1%;

                padding-left: 5%;

            }

        </style>

    </head>

    <body>

        <video id="videoPlayer" width="20%" controls autoplay muted >

            <source src="/video" type="video/mp4" />

        </video>

    </body>

</html>

 const CHUNK\_SIZE = 10 \*\* 5;

Definimos un tamaño CHUNK de 100 KB

        const start = Number(range.replace(/\D/g, ""));

        const end = Math.min(start + CHUNK\_SIZE, videoSize - 1);

        console.log("inicio:" + start + "  fin:" + end);

Aqui se calcula los rangos de inicio y fin y se van mostrando en la consola

const headers = {

        "Content-Range": `bytes ${start}-${end}/${videoSize}`,

        "Accept-Ranges": "bytes",

        "Content-Length": contentLength,

        "Content-Type": "video/mp4",

    };

En esta parte el reproductor de video sabe qué tan avanzado está en función del tamaño del video

res.writeHead(206, headers);

ahora enviamos una respuesta al request utilizando el estado 206 que indica que se envía contenido parcial

const videoStream = fs.createReadStream(videoPath, { start, end });

Utilizamos el file system para crear el read stream, utilizando la ruta del video como argumento y el inicio y fin como campos del objeto options

videoStream.pipe(res);

Finalmente hacemos un pipe

app.listen(8000, function () {

    console.log("Listening on port 8000!");

});

Creamos un servidor que escuche en puerto 8000

En nuestro archivo package.json configuramos el script start:

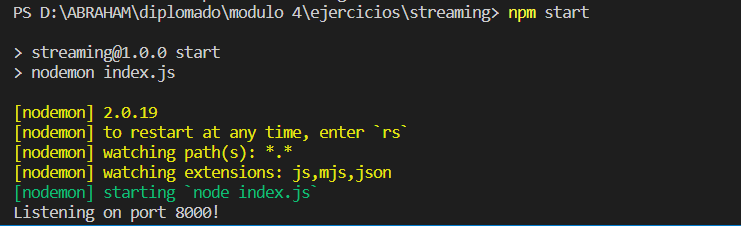
"scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "nodemon index.js"

  },

Iniciamos el servidor con npm start



Finalmente en un browser accedemos a hhtp://localhost:8000

