UNIVERSIDAD NACIONAL DE SAN AGUSTÍN
FACULTAD DE INGENIERIA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

CURSO:
LABORATORIO PROGRAMACION WEB 2

DOCENTE:
CORRALES DELGADO CARLOS JOSE

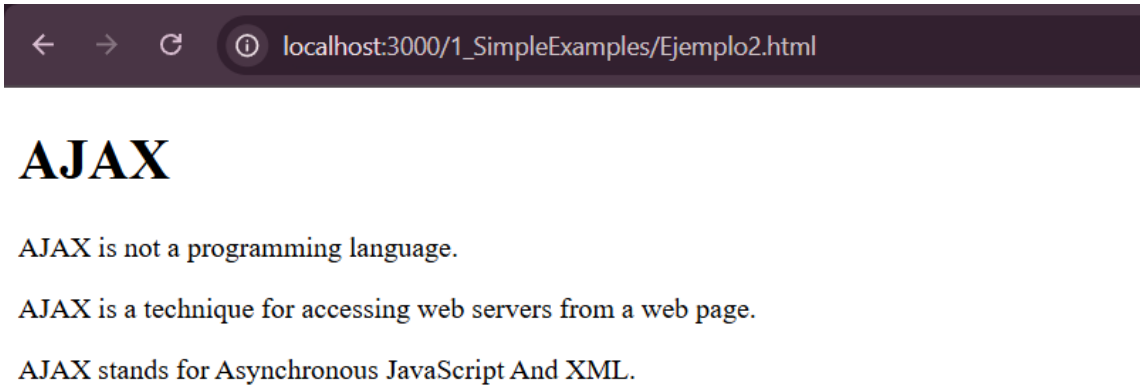
TEMA:
INFORME EJEMPLOS W3SCHOOLS

PRESENTADO POR:
SANTA CRUZ VILLA DIEGO SEBASTIAN

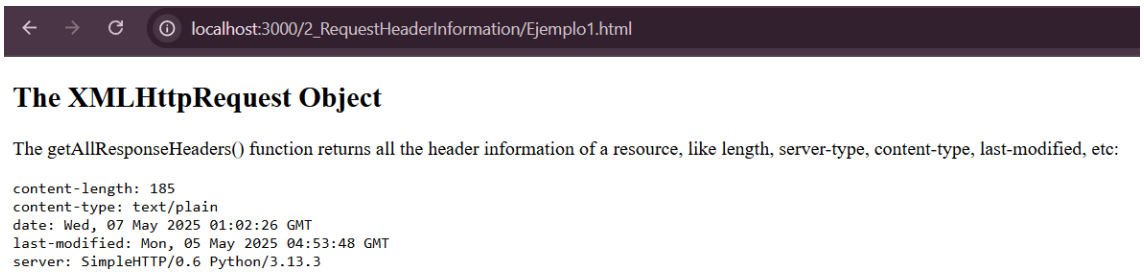
RESULTADOS:

ESTRUCTURA DE CARPETAS:

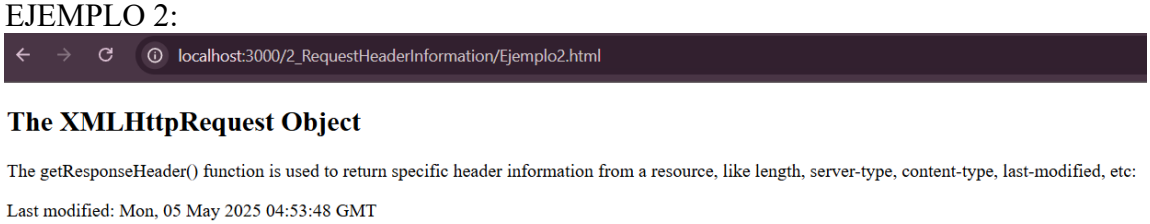
Seccion: SimplesExamples
EJEMPLO 1 Y 2:



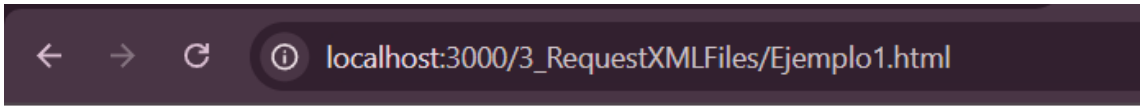
Seccion: RequestHeaderInformation
EJEMPLO 1:



EJEMPLO 2:



Seccion: RequestXMLFiles
EJEMPLO 1:



The XMLHttpRequest Object

Retrieve data from XML file

Status: 200

Status text: OK

Response: Tove Jani Reminder Don't forget me this weekend!

Get XML data

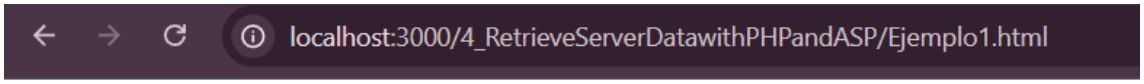
EJEMPLO 2:

The XMLHttpRequest Object

Get my CD collection

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros
Bee Gees	One night only
Dr.Hook	Sylvias Mother
Rod Stewart	Maggie May
Andrea Bocelli	Romanza
Percy Sledge	When a man loves a woman
Savage Rose	Black angel
Many	1999 Grammy Nominees
Kenny Rogers	For the good times
Will Smith	Big Willie style
Van Morrison	Tupelo Honey
Jorn Hoel	Soulsville
Cat Stevens	The very best of
Sam Brown	Stop
T'Pau	Bridge of Spies
Tina Turner	Private Dancer

SECCION: RetrieveServerDatawithPHPandASP
EJEMPLO 1:



The XMLHttpRequest Object

Start typing a name in the input field below:

Suggestions:

Error response

Error code: 404

Message: File not found.

Error code explanation: 404 - Nothing matches the given URI.

First name: Diego

EJEMPLO 2:

The XMLHttpRequest Object

Start typing a name in the input field below:

Suggestions:

Error response

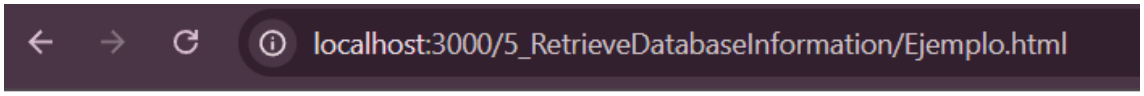
Error code: 404

Message: File not found.

Error code explanation: 404 - Nothing matches the given URI.

First name: Diego

Seccion: RetrieveDatabaseInformation
EJEMPLO:



The XMLHttpRequest Object

North/South

Error response

Error code: 404

Message: File not found.

Error code explanation: 404 - Nothing matches the given URI.

Seccion: AJAXApplications
EJEMPLO 1:

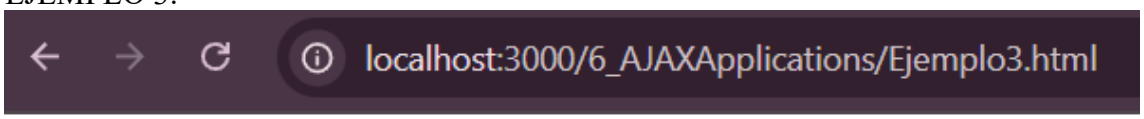
Get my CD collection

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros
Bee Gees	One night only
Dr.Hook	Sylvias Mother
Rod Stewart	Maggie May
Andrea Bocelli	Romanza
Percy Sledge	When a man loves a woman
Savage Rose	Black angel
Many	1999 Grammy Nominees
Kenny Rogers	For the good times
Will Smith	Big Willie style
Van Morrison	Tupelo Honey
Jorn Hoel	Soulsville
Cat Stevens	The very best of
Sam Brown	Stop
T'Pau	Bridge of Spies
Tina Turner	Private Dancer
Kim Larsen	Midt om natten
Luciano Pavarotti	Pavarotti Gala Concert
Otis Redding	The dock of the bay
Simply Red	Picture book

EJEMPLO 2:

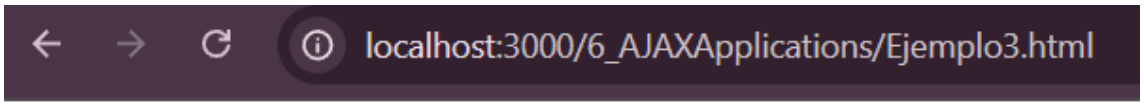
Artist: Bob Dylan
Title: Empire Burlesque
Year: 1985

EJEMPLO 3:



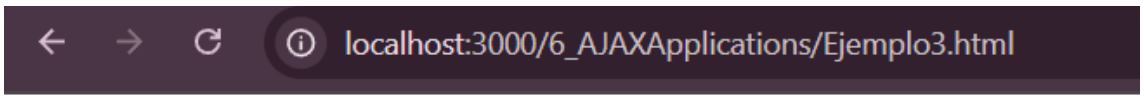
Artist: Bonnie Tyler
Title: Hide your heart
Year: 1988

<< >>



Artist: Dolly Parton
Title: Greatest Hits
Year: 1982

<< >>



Artist: Bob Dylan
Title: Empire Burlesque
Year: 1985

<< >>

EJEMPLO 4:

Click on a CD to display album information.

Artist: Bob Dylan
Title: Empire Burlesque
Year: 1985

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros
Bee Gees	One night only
Dr.Hook	Sylvias Mother
Rod Stewart	Maggie May
Andrea Bocelli	Romanza
Percy Sledge	When a man loves a woman
Savage Rose	Black angel
Many	1999 Grammy Nominees
Kenny Rogers	For the good times
Will Smith	Big Willie style
Van Morrison	Tupelo Honey
Jorn Hoel	Soulsville
Cat Stevens	The very best of

DESARROLLO:

Seccion SimplesExamples

En la seccion de SimplesExamples nos enseña el uso de AJAX con XMLHttpRequest para poder remplazar el texto de una etiqueta con la de un archivo fuera del html.
Para la diferencia del ejemplo 1 y 2 solo es el uso de funcion que toma la url en vez de pasarla directamente.

```
function loadDoc(url, xFunction) {
    const xhttp=new XMLHttpRequest();
    xhttp.onload = function() {xFunction(this);}
    xhttp.open("GET", url);
    xhttp.send();
}
function myFunction(xhttp) {
    document.getElementById("demo").innerHTML =  xhttp.responseText;
}
```

Seccion RequestHeaderInformation

En el ejemplo 1 nos muestra cómo ver la información que el servidor envía junto con una respuesta a una petición AJAX.
Cuando se pide el archivo ajax_info.txt, se usa la función getAllResponseHeaders() para ver detalles como el tipo de archivo, la fecha de modificación, el servidor, entre otros.

```
<script>
    const xhttp = new XMLHttpRequest();
    xhttp.onload = function() {
        document.getElementById("demo").innerHTML =
        this.getAllResponseHeaders();
    }
    xhttp.open("GET", "ajax_info.txt");
    xhttp.send();
</script>
```

En el ejemplo nos enseña cómo obtener un solo dato específico de la respuesta del servidor usando getResponseHeader().
En este caso, se consigue la fecha de la última vez que se modificó el archivo (Last-Modified) y se muestra en la página.

```
<script>
    const xhttp=new XMLHttpRequest();
    xhttp.onload = function() {
        document.getElementById("demo").innerHTML =
        this.getResponseHeader("Last-Modified");
    }
    xhttp.open("GET", "ajax_info.txt");
    xhttp.send();
</script>
```

Seccion RequestXMLFiles

En el ejemplo 1 nos muestra cómo usar AJAX para obtener información de un servidor. Uno permite ver todas las cabeceras que acompañan una respuesta, como el tipo de contenido o la fecha de modificación. Otro muestra cómo obtener solo una cabecera específica, como la fecha de la última vez que se actualizó un archivo. También hay un ejemplo que recupera un archivo XML y muestra el código de estado, su mensaje y el contenido del archivo como texto.

```
<html>
    <body>

        <h2>The XMLHttpRequest Object</h2>
        <h2>Retrieve data from XML file</h2>
        <p><b>Status:</b> <span id="A1"></span></p>
        <p><b>Status text:</b> <span id="A2"></span></p>
        <p><b>Response:</b> <span id="A3"></span></p>

        <button onclick="loadDoc('note.xml')">Get XML data</button>

        <script>
            function loadDoc(url) {
                const xhttp = new XMLHttpRequest();
                xhttp.onload = function() {
                    document.getElementById('A1').innerHTML = this.status;
                    document.getElementById('A2').innerHTML = this.statusText;
                    document.getElementById('A3').innerHTML = this.responseText;
            }
            xhttp.open("GET", url);
            xhttp.send();
            }
        </script>

    </body>
</html>
```

En el ejemplo 2 carga un archivo XML usando AJAX y muestra su contenido en forma de tabla HTML. Extrae los nombres de los artistas y los títulos de los discos del archivo y los organiza en filas. En resumen, se usa AJAX no solo para leer el archivo, sino también para transformar sus datos en una tabla dinámica que se puede ver en la página web.

```
<script>
    function loadDoc() {
        const xhttp = new XMLHttpRequest();
        xhttp.onload = function() {
            myFunction(this);
        }
        xhttp.open("GET", "cd_catalog.xml");
        xhttp.send();
    }
    function myFunction(xml) {
        const xmlDoc = xml.responseXML;
        const x = xmlDoc.getElementsByTagName("CD");
        let table="<tr><th>Artist</th><th>Title</th></tr>";
        for (let i = 0; i <x.length; i++) {
            table += "<tr><td>" +
            x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
            "</td><td>" +
            x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
            "</td></tr>";
        }
        document.getElementById("demo").innerHTML = table;
    }
</script>
```

Sección RetrieveServerDatawithPHPandASP

No se logro resultados funcionales, solo se logro cargar el html sin embargo el script no tiene de donde sacar información

```
<script>
    function showHint(str) {
    if (str.length == 0) {
        document.getElementById("txtHint").innerHTML = "";
        return;
    }
    const xhttp = new XMLHttpRequest();
    xhttp.onload = function() {
        document.getElementById("txtHint").innerHTML =
        this.responseText;
    }
    xhttp.open("GET", "gethint.asp?q="+str);
    xhttp.send();
    }
</script>
```

Sección RetrieveDatabaseInformation

No se logro resultados funcionales, solo se logro cargar el html sin embargo el script no tiene de donde sacar información

```
<script>
  function showCustomer(str) {
    if (str == "") {
      document.getElementById("txtHint").innerHTML = "";
      return;
    }
    const xhttp = new XMLHttpRequest();
    xhttp.onload = function() {
      document.getElementById("txtHint").innerHTML = this.responseText;
    }
    xhttp.open("GET", "getcustomer.php?q=" + str);
    xhttp.send();
  }
</script>
```

Sección AJAXApplications

En el ejemplo 1 carga un archivo XML con AJAX y muestra la información de una colección de CDs en una tabla HTML. Sin embargo aquí el procesamiento del XML se hace directamente en la función que lo carga, y luego se pasa solo la información necesaria a otra función que crea la tabla.

```
12      <body>
13
14          <button type="button" onclick="loadXMLDoc()">Get my CD collection</button>
15          <br><br>
16          <table id="demo"></table>
17
18          <script>
19              function loadXMLDoc() {
20                  const xhttp = new XMLHttpRequest();
21                  xhttp.onload = function() {
22                      const xmlDoc = xhttp.responseXML;
23                      const cd = xmlDoc.getElementsByTagName("CD");
24                      myFunction(cd)
25                  }
26                  xhttp.open("GET", "cd_catalog.xml");
27                  xhttp.send();
28              }
29
30              function myFunction(cd) {
31                  let table="<tr><th>Artist</th><th>Title</th></tr>";
32                  for (let i = 0; i < cd.length; i++) {
33                      table += "<tr><td>" +
34                      cd[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
35                      "</td><td>" +
36                      cd[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
37                      "</td></tr>";
38                  }
39                  document.getElementById("demo").innerHTML = table;
40              }
41          </script>
42
43      </body>
44  </html>
```

En el ejemplo 2 también usa AJAX para cargar un archivo XML, pero en lugar de mostrar todos los CDs en una tabla, solo presenta la información del primer CD (artista, título y año) dentro de un div. A diferencia de los anteriores, se enfoca en mostrar los detalles de un solo elemento del XML, lo que es útil cuando se quiere destacar un registro específico.

```html
<!DOCTYPE html>
<html>
    <body>

    <div id='showCD'></div>

    <script>
        const xhttp = new XMLHttpRequest();
        xhttp.onload = function() {
            const xmlDoc = xhttp.responseXML;
            const cd = xmlDoc.getElementsByTagName("CD");
            myFunction(cd, 0);
        }
        xhttp.open("GET", "cd_catalog.xml");
        xhttp.send();

        function myFunction(cd, i) {
            document.getElementById("showCD").innerHTML =
            "Artist: " +
            cd[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
            "<br>Title: " +
            cd[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
            "<br>Year: " +
            cd[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue;
        }
    </script>

    </body>
</html>
```

En el ejemplo 3 carga un archivo XML con AJAX y permite ver los CDs uno por uno usando botones de anterior y siguiente. Cada vez que se hace clic, se actualiza la información mostrada (artista, título y año) según el CD actual. Podria compararse como pasar por una base de datos.

```html
<!DOCTYPE html>
<html>
    <body>

        <div id='showCD'></div><br>
        <input type="button" onclick="previous()" value="<<">
        <input type="button" onclick="next()" value=">>">

        <script>
            let i = 0;
            let len;
            let cd;

            const xhttp = new XMLHttpRequest();
            xhttp.onload = function() {
                const xmlDoc = xhttp.responseXML;
                cd = xmlDoc.getElementsByTagName("CD");
                len = cd.length;
                displayCD(i);
            }
            xhttp.open("GET", "cd_catalog.xml");
            xhttp.send();

            function displayCD(i) {
                document.getElementById("showCD").innerHTML =
                "Artist: " +
                cd[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
                "<br>Title: " +
                cd[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
                "<br>Year: " +
                cd[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue;
            }

            function next() {
                if (i < len-1) {
                    i++;
                    displayCD(i);
                }
            }

            function previous() {
                if (i > 0) {
                    i--;
                    displayCD(i);
                }
            }
        </script>

    </body>
</html>
```

Finalmente en el ejemplo 4 carga un archivo XML con AJAX y muestra una tabla con los artistas y títulos de los CDs. Al hacer clic en una fila, se muestran los detalles completos del CD seleccionado, como el artista, el título y el año. Lo nuevo aquí es que cada fila tiene un evento onclick, lo que convierte la tabla en un catálogo interactivo donde el usuario puede explorar los CDs fácilmente.

Click on a CD to display album information.

Artist: Bob Dylan
Title: Empire Burlesque
Year: 1985

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits

```
<script>
    const xhttp = new XMLHttpRequest();
    let cd;
    xhttp.onload = function() {
        const xmlDoc = xhttp.responseXML;
        cd = xmlDoc.getElementsByTagName("CD");
        loadCD();
    }
    xhttp.open("GET", "cd_catalog.xml");
    xhttp.send();

    function loadCD() {
        let table="<tr><th>Artist</th><th>Title</th></tr>";
        for (let i = 0; i < cd.length; i++) {
            table += "<tr onclick='displayCD(" + i + ")'><td>";
            table += cd[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue;
            table += "</td><td>";
            table += cd[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue;
            table += "</td></tr>";
        }
        document.getElementById("demo").innerHTML = table;
    }

    function displayCD(i) {
        document.getElementById("showCD").innerHTML =
        "Artist: " +
        cd[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +
        "<br>Title: " +
        cd[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
        "<br>Year: " +
        cd[i].getElementsByTagName("YEAR")[0].childNodes[0].nodeValue;
    }
</script>
```

CONCLUSIONES:

En conjunto, estos ejemplos muestran cómo usar AJAX para trabajar con archivos XML y obtener información del servidor sin recargar la página. Se aprendió a leer cabeceras HTTP completas o individuales, recuperar y mostrar contenido como texto, y transformar datos XML en elementos visuales como tablas y divisiones. Además, se exploraron diferentes formas de organizar el código, desde mostrar un solo registro hasta recorrer una lista de elementos o construir interfaces interactivas con botones y eventos de clic. En resumen, se comprendió cómo AJAX permite construir aplicaciones web dinámicas que responden de manera más flexible y eficiente a las acciones del usuario.