Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

Laboratorio: Computación en cliente web

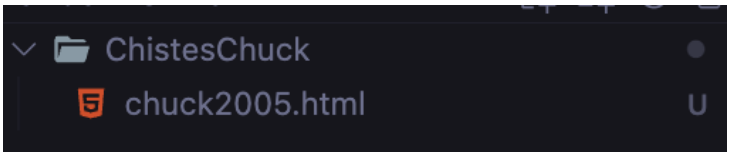
Objetivos

Trabajar con estándares web relacionados con conexiones AJAX, funciones asíncronas y componentes.

Desarrollar el laboratorio tal y como se explica durante la sesión y narrar cómo se ha hecho, siguiendo para ello las instrucciones de este documento. Si no puedes asistir a la sesión en directo, recuerda que siempre la tienes a tu disposición como grabación.

Comienzo del ejercicio

Se procedió con la creación del directorio raíz con nombre: ***ChistesChuck.*** Este mismo contendrá todo el código para la correcta ejecución del trabajo.



Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

Resolución del ejercicio a la manera de 2005

Continuando con el desarrollo de nuestro ejercicio, se creó el archivo: ***chuck2005.html*** con las dependencias necesarias para utilizar Bootstrap V5. El template inicial luce de la siguiente manera:

chuck2005.html U

ChistesChuck > chuck2005.html > ...

```html
<!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>Chuck 2005</title>

    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
      crossorigin="anonymous"
    />
  </head>
  <body>
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
      crossorigin="anonymous"
    ></script>
  </body>
</html>
```

Fig. 1 - HTML template con Bootstrap V5

Haremos una pequeña interfaz para que nuestro usuario con tan solo picar un botón, pueda hacer request de la manera 2005 de algún chiste:

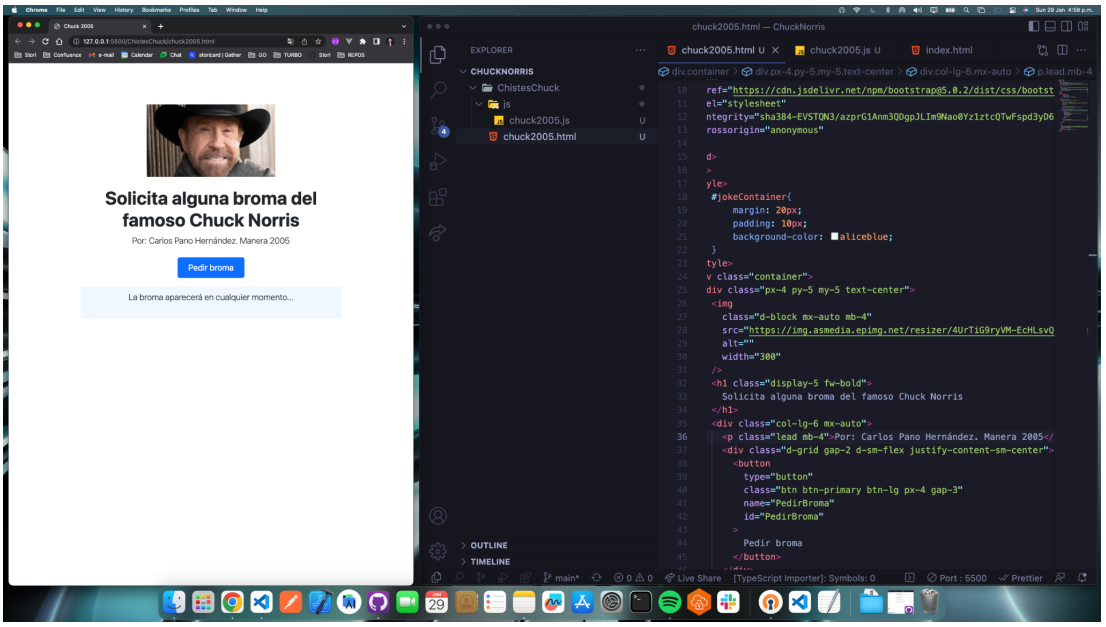


Fig. 2 - API JOKES UI

Una vez terminado nuestro programa y unos cuantos estilos, se procedió a hacer la petición de nuestra broma. Se tuvieron algunos inconvenientes con la URL proporcionada por el profesor, por lo que se decidió buscar un API funcional en Internet: https://api.chucknorris.io/

Misma que proporciona un JSON con alguna broma Random. Para mantener orden en nuestro repositorio, se creó una carpeta llamada: **"js".** Misma que contendrá todo el funcionamiento de nuestra aplicación.

De nuestra interfaz gráfica, se obtuvo el valor de nuestro botón y texto que será sustituido con la llamada. El código se muestra a continuación:

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

```javascript
document.getElementById("PedirBroma").addEventListener("click", requestJoke);
let textoBroma = document.getElementById("textoBroma");

function requestJoke() {
  xmlhttp = new XMLHttpRequest();

  xmlhttp.open("GET", "https://api.chucknorris.io/jokes/random", true);

  xmlhttp.onreadystatechange = function () {
    let textoChiste = JSON.parse(this.response).value;
    console.log("Chiste recibido: " + textoChiste);

    textoBroma.innerHTML = textoChiste;
  };
  xmlhttp.send();
}
```

Fig. 3 - XMLHttpRequest

Pruebas de funcionamiento

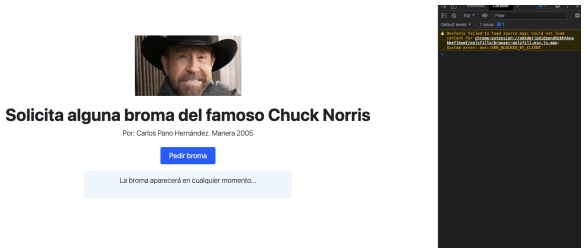
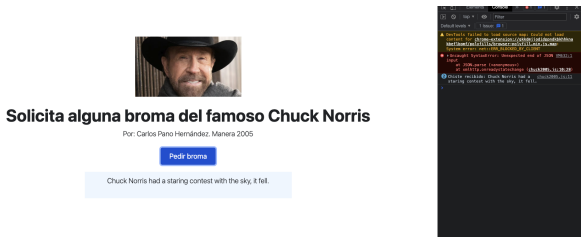
Estado	Resultado
Inicial Descripción en: *"La broma aparecerá en cualquier momento…"*	
Prueba 1:	
Prueba 2:	

Fig. 4 - Testing

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

De igual manera se muestra un screen de nuestros logs:

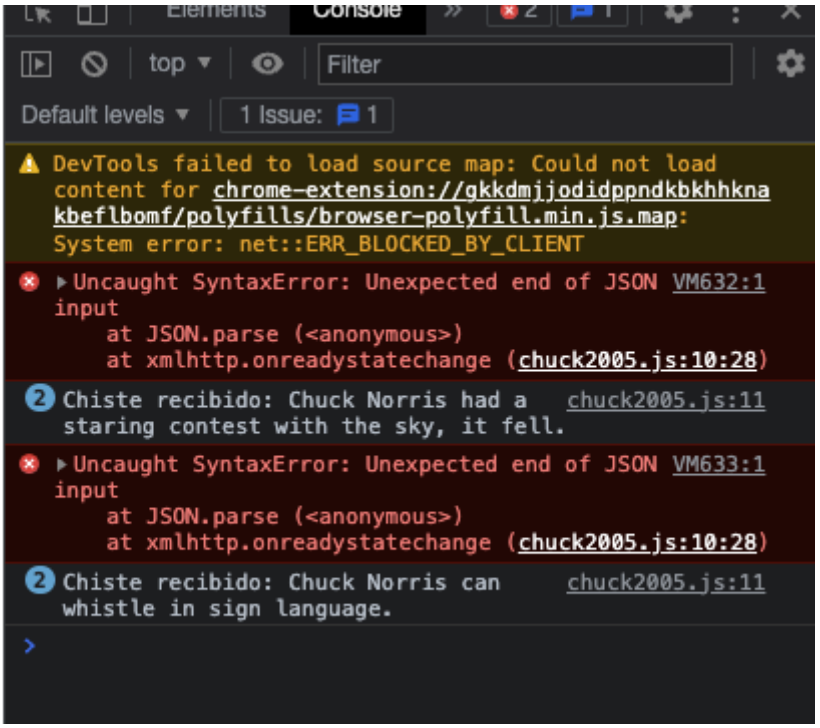


Fig. 4 - Logs 2005

Podemos concluir que este ejercicio fue realizado con éxito.

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

Resolución del ejercicio a la manera de 2006

Para este ejercicio, se utilizó la misma plantilla del 2005 sumándole la librería Jquery como se muestra a continuación:

```
55
56      <script
57        src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstra
58        integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfA
59        crossorigin="anonymous"
60      ></script>
61      <script
62        src="https://code.jquery.com/jquery-3.6.3.min.js"
63        integrity="sha256-pvPw+upLPUjgMXY0G+8O0xUf+/Im1MZjXxxgOcBQBXU="
64        crossorigin="anonymous"
65      ></script>
66      <script src="js/chuck2006.js"></script>
67    </body>
68  </html>
69
```

Fig. 5 - JQuery Library

Dentro de nuestro archivo “chuck2006.js”, se podrá encontrar el siguiente código:

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

```
EXPLORER                    chuck2006.html U    chuck2006.js U ×
CHUCKNORRIS                 ChistesChuck > js > chuck2006.js > ...
  ChistesChuck
    js
      chuck2005.js U        document.getElementById("PedirBroma").addEventListener("click", requestJoke);
      chuck2006.js U        let textoBroma = document.getElementById("textoBroma");
    chuck2005.html U
    chuck2006.html U
                            function requestJoke() {
                              $.get("https://api.chucknorris.io/jokes/random", (response) => {
                                var textoChiste = response.value;
                                console.log(textoChiste);
                                $("#textoBroma").html(textoChiste);
                              });
                            }
```

Fig. 6 - JQuery code

Mismo que se detona cuando el botón es seleccionado. Con el código terminado, se procede a la fase de pruebas:

Estado	**Resultado**
Inicial Descripción en: *"La broma aparecerá en cualquier momento…"*	**Solicita alguna broma del famoso Chuck Norris** Por: Carlos Pano Hernández. Manera 2006 Pedir broma La broma aparecerá en cualquier momento...
Prueba 1:	**Solicita alguna broma del famoso Chuck Norris** Por: Carlos Pano Hernández. Manera 2006 Pedir broma Chuck Norris can start a fire by rubbing 2 ice cubes together !!
Prueba 2:	**Solicita alguna broma del famoso Chuck Norris** Por: Carlos Pano Hernández. Manera 2006 Pedir broma It is possible to contract hepatitis from smelling one of Chuck Norris's farts.

Fig. 5 - Testing JQuery

Podemos notar a primera vista un log más limpio que el ejercicio anterior, aunado a ello, las pruebas fueron exitosas.

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

Preguntas:

¿Qué diferencias ves con respecto al ejercicio anterior?

De primera instancia vemos una enorme reducción en las líneas de código, ya que las librerías de JQuery simplifican la petición HTTPS. Me agrada el uso de variables ('$') como en PHP, además que el código luce más elegante.

¿Cómo simplifica la vida jQuery?

El uso de librerías por parte de JQuery simplifica tareas comunes como peticiones GET. Esto simplifica enormemente las líneas de código.

¿Qué ocurre si tenemos varios *tags* `h1`?

Si no iteramos con un FOR, el valor del chiste será aplicado únicamente al último H1 del archivo HTML. En este caso, usamos un ID selector para identificar el párrafo que queremos cambiar, por lo que no tuvimos problemas.

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

Resolución con *plugin* de jQuery (hasta 2014)

Este ejercicio no pudo ser realizado debido a que la función "getRandomJoke" ya no se encuentra disponible en ICNDB. Esto se debe a que el mismo dominio ya no existe. El problema se relaciona con el cambio de API en el ejercicio 1.

Espero no afectar mi calificación.

—----------

```
$.icndb.getRandomJoke((response) => { var textoChiste = response.joke; });
```

Que uséis el método $.icndb.getRandomJokes (atentos a la «s» del final) para poblar los elementos de una lista ordenada (tag <ul></ul>). Ejemplo:

```
$.icndb.getRandomJokes({
number: 10,
        success: (response) => {
 response.forEach(element => { $("ul").append('<li>' + element.joke + '</li>'); });
}});
```

—----------

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

Resolución en 2014

Para la solución de este ejercicio, se modificó la interfaz para hacerla mucho más User Friendly (tal como se muestra a continuación):

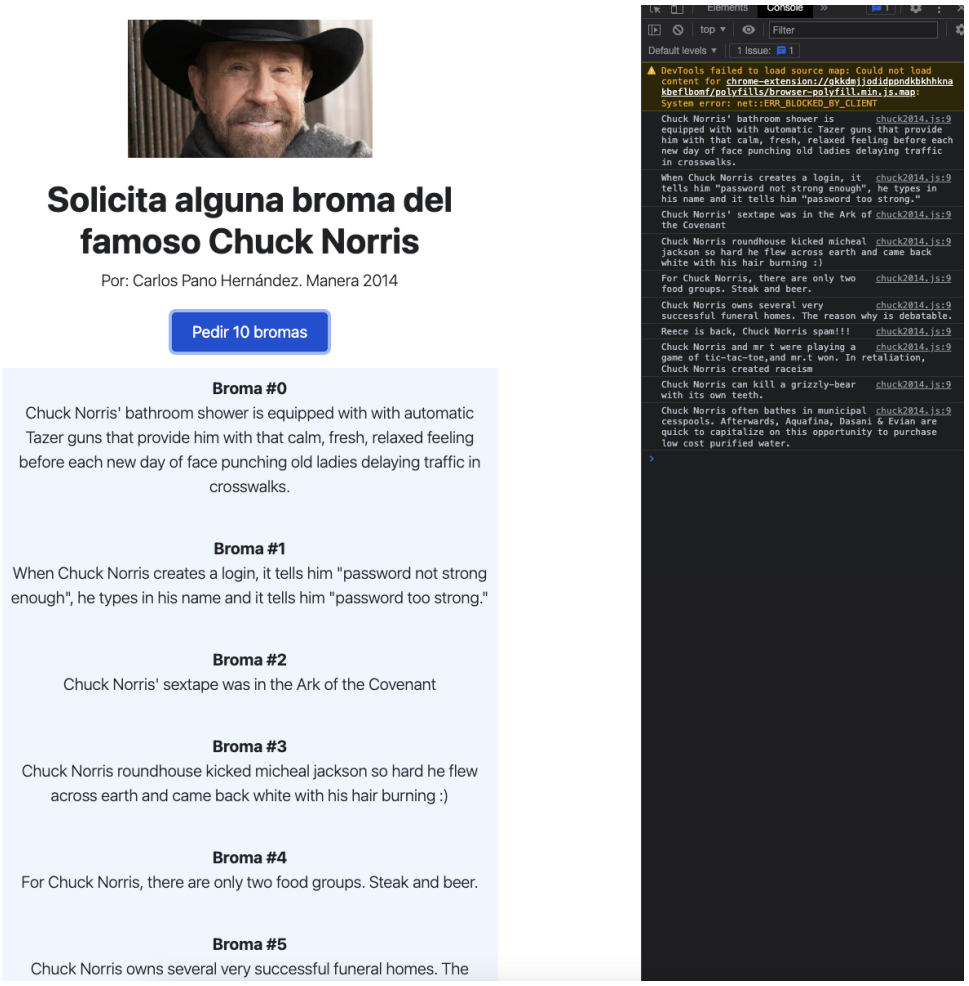
Fig. 6 - UI, FETCH API

Haciendo las modificaciones anteriores, se realizó el código siguiente para la llamada Fetch:

Fig. 7 - FETCH API CODE

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

En la figura 7 vemos cómo llamamos a nuestra función: `callChuckNorrisJoke` 10 veces para hacer la impresión de 10 bromas. El resultado:



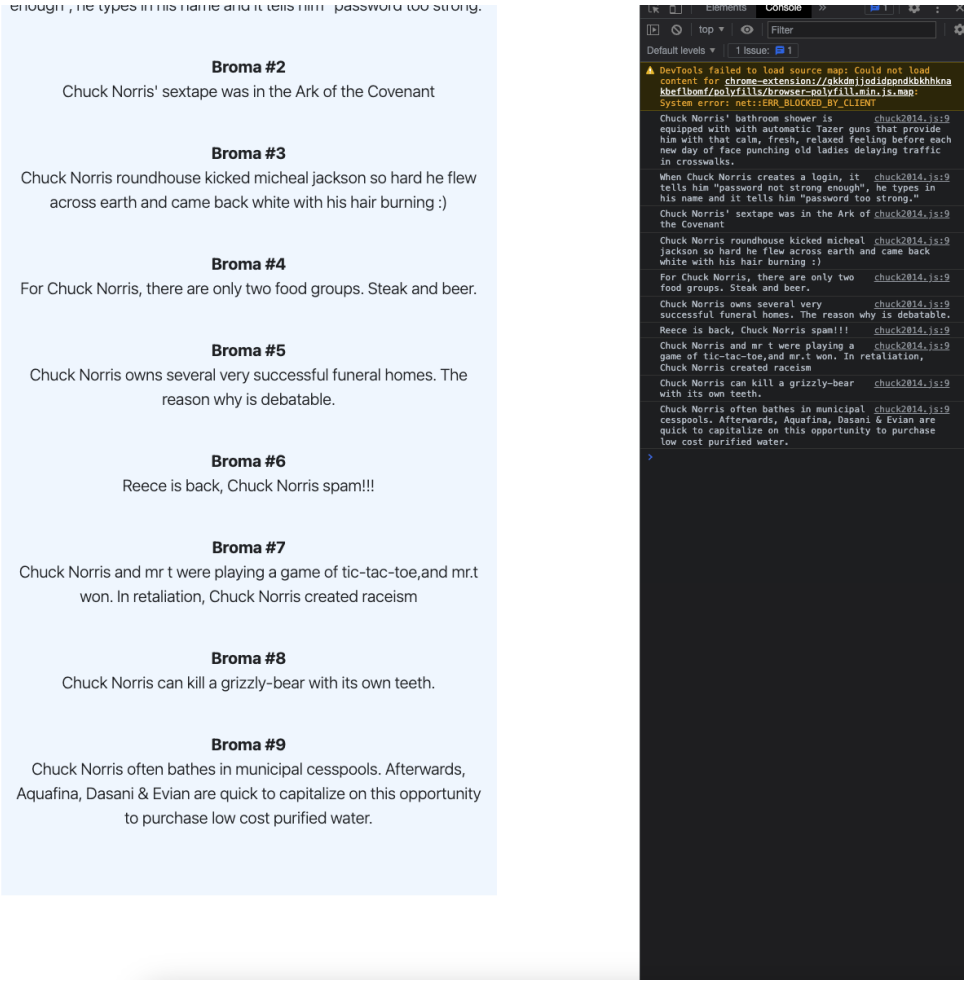


Fig. 8 - FETCH API CODE - Testing

Definitivamente la evolución WEB va tan rápido que ya somos capaces de llamar estos métodos y hacer aplicaciones sencillas a gran velocidad con código elegante.

¿Qué es el WHATWG (https ://whatwg.org/)?

Se trata de una comunidad de código abierto para la creación de estándares WEB y protocolos de comunicación.

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

Se pide también repetir el mismo ejercicio con **node-fetch**, y simplemente imprimir por pantalla el texto de cada chiste.

Para la siguiente sección se realizó un proyecto de Node con Express en formato MVC y dentro de nuestro directorio raíz "ChistesChuck":

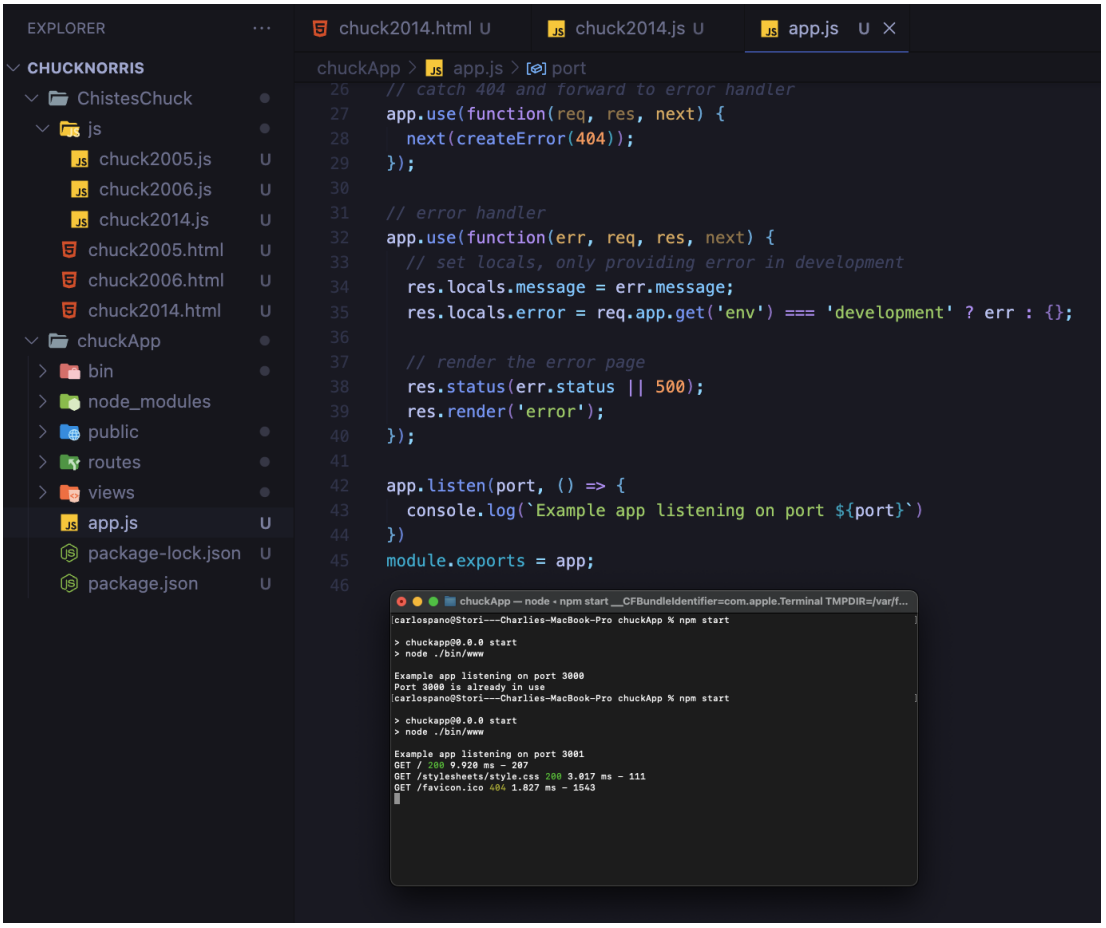


Fig. 9 - Node APP

El proyecto correrá en el puerto 3001:

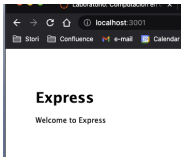


Fig. 10 - Port

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

Así bien, utilizamos la misma plantilla de este mismo ejercicio para la vista HTML/EJS:

Fig. 11 - Main view

Se instaló la paquetería correspondiente con el comando: npm i node-fetch:

```
carlospano@Stori---Charlies-MacBook-Pro chuckApp % npm i node-fetch

added 6 packages, and audited 61 packages in 2s

3 packages are looking for funding
  run `npm fund` for details

4 vulnerabilities (3 high, 1 critical)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Fig. 12 - Instalación de paqueterías

Y se procedió a inicializar el módulo correspondiente en la vista de nuestro controlador:

```
chuckApp > routes > index.js > router.get("/") callback > then() callback
var express = require("express");
const fetch = require("node-fetch");
var router = express.Router();

/* GET home page. */
router.get("/", function (req, res, next) {
  fetch("https://api.chucknorris.io/jokes/random")
    .then((res) => res.json())
    .then((json) => {
      let joke = json.value;
      console.log(joke);
      res.render("index", { joke });
    });


});

module.exports = router;
```

Fig. 13 - Node/Fetch - GET petition

Con lo anterior, cuando cargamos nuestra vista, se hace la petición de nuestra broma y se manda a nuestra view para que sea renderizada:

```
chuckApp — node ◂ npm exec nodemon .app.js __CFBundleIdentifier=com.ap
carlospano@Stori---Charlies-MacBook-Pro chuckApp % npx nodemon .\app.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/www .app.js`
Example app listening on port 3001
Chuck Norris once won American idle by riding a unicycle
GET / 200 789.819 ms - 2103
GET /js/chuck2014.js 404 1.826 ms - 1543
```

Solicita alguna broma del famoso Chuck Norris

Por: Carlos Pano Hernández. Manera 2014 - Node/Fetch

Chuck Norris wins NASCAR races with all right turns.

Fig. 14 - Node/Fetch - GET petition - VIEW

Sin duda utilizar Node-fetch fue mucho más laborioso debido a la configuración previa que se debe hacer para manipular la aplicación. Después de dicha configuración, es un método realmente eficiente y sobre todo, muy eficiente ya que es asíncrono.

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

Resolución del ejercicio con Web Components (presente)

Para esta sección de nuestro trabajo, se instaló de manera global el package manager: “Bower”:

```
chuckApp — -zsh — 93×76
carlospano@Stori---Charlies-MacBook-Pro chuckApp % sudo npm install -g bower
Password:

added 1 package, and audited 2 packages in 3s

found 0 vulnerabilities
carlospano@Stori---Charlies-MacBook-Pro chuckApp %
```

Después de finalizar con las instalaciones correspondientes, nuestro fichero quedó de la siguiente manera:

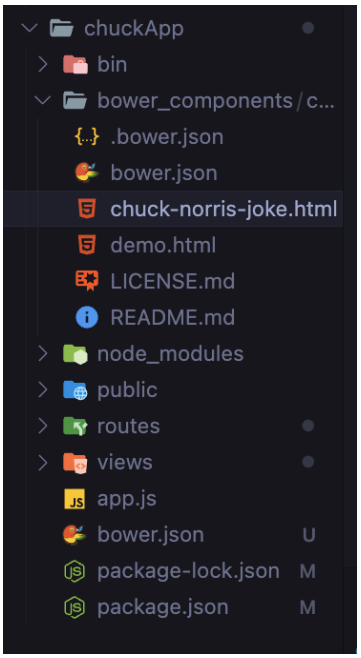


Fig. 14 - Bower, instalación

Se continuó con la inserción de las etiquetas correspondientes en nuestro código HTML como se especificó en el ejercicio:

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

```html
<div class="col-lg-6 mx-auto">
  <p class="lead mb-4">
    Por: Carlos Pano Hernández. Manera 2022 - Bower
  </p>
  <div class="d-grid gap-2 d-sm-flex justify-content-sm-center"></div>
  <div id="jokeContainer">
    <chuck-norris-joke></chuck-norris-joke>
    <p class="lead mb-4" id="textoBroma"></p>
  </div>
```

Fig. 14 - Bower, resultado

Desafortunadamente, tanto el CDN, como la instalación global de nuestras dependencias apuntan a una API que ya no existe:

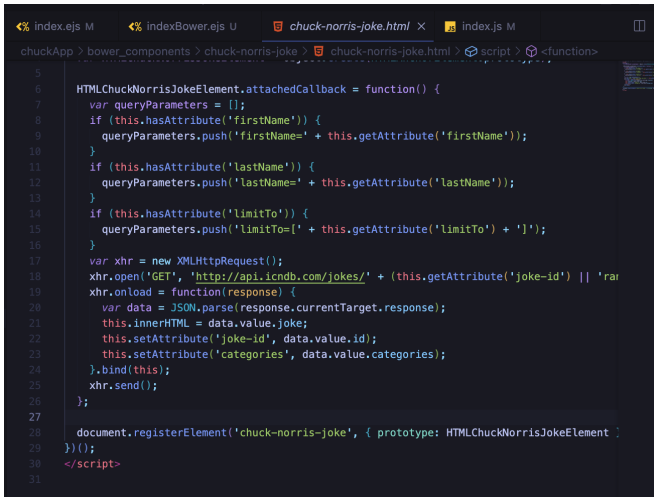


Fig. 14 - Bower, Chuck Norris Code

Asignatura	Datos del alumno	Fecha
Computación en el Cliente Web	Apellidos: **Pano Hernández**	**29/01/2023**
	Nombre: **Carlos**	

Conclusión

Me encuentro agradecido con estas múltiples aplicaciones que vimos para llamar API's. Me pareció muy interesante ver cómo las tecnologías evolucionan a gran velocidad y cada vez las estructuras de datos de nuestros códigos se van componetizando.

Sin duda alguna el método Fetch es el más rápido de todos, puesto que parece ser que es más universal y ampliamente utilizado. La api de ICNDB se encontraba desactualizada, por lo que se requirió de buscar recursos externos.

A pesar de lo anterior, se lograron complir con todos los objetivos de este laboratorio.

Rúbrica

- Las peticiones AJAX se realizan correctamente. Peso: 40%
- Los ejercicios y sus resultados están justificados. Peso: 30%
- La presentación final del ejercicio es limpia, cuidada y acorde a un nivel de master universitario. Peso 30%