

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



## PRÁCTICA DE LABORATORIO

**CARRERA:** ING. DE SISTEMAS

**ASIGNATURA:** APLICACIONES DISTRIBUIDAS

**NRO. PRÁCTICA:** 1 **TÍTULO PRÁCTICA:** Consumo de APIs web (plataformas en la nube)

### OBJETIVOS

- Conocer las arquitecturas y patrones arquitectónicos web para el diseño de aplicaciones web
- Interactuar con servicios web de plataformas en la nube

### INSTRUCCIONES

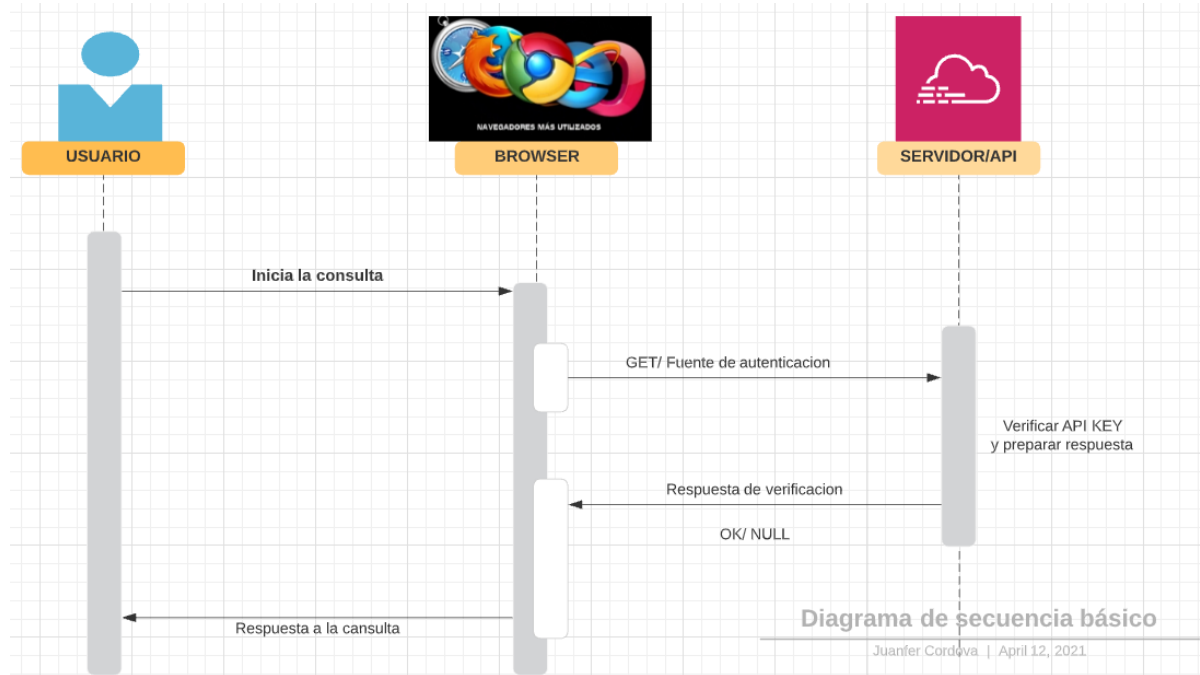
Desarrollar una aplicación web usando una de las API gratuitas de [API List Fun](#). Tener en cuenta que se deben aplicar buenas prácticas para el desarrollo de la interfaz gráfica de usuario, para la cuál se permite utilizar plantilla de [Bootstrap](#).

#### Requisitos:

- La aplicación Web debe permitir buscar la información a través de un nombre.
- Además, se deberá visualizar toda la información disponible de la base de datos.

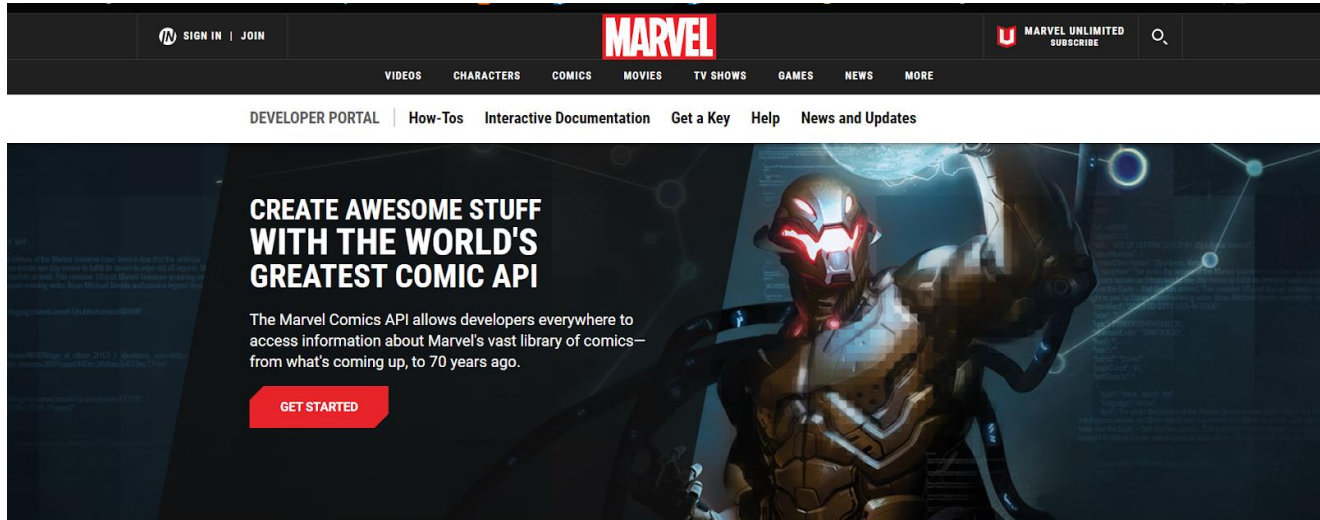
### ACTIVIDADES POR DESARROLLAR

1. Identificar gráficamente la arquitectura web de la aplicación a desarrollar.

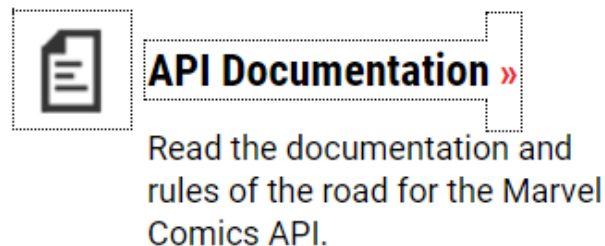


2. Generar una llave para consumir los servicios web de la API (opcional, depende de la API seleccionada).

En este punto trabajaremos con una API gratuita obtenida en [API List Fun](#), la cual trata sobre MARVEL.



Una vez que ya estemos dentro de Marvel, tenemos que dirigirnos hacia API Documentación, para poder crearnos las llaves necesaria y comenzar a desarrollar nuestro trabajo.



Una vez que hayamos ingresado a API Documentación, tendremos unos avisos importantes, uno de los cuales nos sugiere mediante un link que creemos nuestra propia llave.

**Regístrese:** **obtenga una clave API**

**Sea un buen ciudadano de la API:** lea, comprenda y cumpla los **términos de uso** de la API de Marvel Comics

**Vínculo posterior:** observe las **pautas de atribución y vinculación** al mostrar datos de la API

**Manténgase en contacto:** cuéntenos sobre lo que está creando y hable con otros desarrolladores en nuestra página de comunidad

**Construye cosas interesantes**

También tenemos un pequeño texto que nos indica cual es la autenticación para aplicaciones del lado del servidor.

## Autenticación para aplicaciones del lado del servidor

Las aplicaciones del lado del servidor deben pasar dos parámetros además del parámetro apiKey:

**ts** : una marca de tiempo (u otra cadena larga que puede cambiar solicitud por solicitud)

**hash** : un resumen md5 del parámetro ts, su clave privada y su clave pública (por ejemplo, md5 (ts + privateKey + publicKey))

Por ejemplo, un usuario con una clave pública de "1234" y una clave privada de "abcd" podría construir una llamada válida de la siguiente manera: `http://gateway.marvel.com/v1/public/comics?`

`ts=1&apikey=1234&hash=ffd275c5130566a2916217b101f26150` (el valor hash es el resumen md5 de 1abcd1234)

Cuando ingresemos al link de obtener una clave API, se nos abre otra ventana en donde tenemos que registrar nuestros datos y verificarlo por medio del correo electrónico.

CREATE YOUR ACCOUNT  
SIGN IN WITH YOUR EMAIL



Show password

Birth Date



Yes! I would like to receive updates, special offers, and other information from Marvel and The Walt Disney Family of Companies.



Yes, I would like to receive Marvel's Need To Know newsletter for news, video/podcast updates, and more.

By creating an account, I agree to the [Terms of Use](#) and acknowledge that I have read the [Privacy Policy](#).

My home country/region: Ecuador. [Change](#).

CREATE ACCOUNT

Already have an account? [Sign In](#)

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de

Una vez registrado y verificado por medio del correo electrónico, ya tendremos acceso a nuestras llaves privada, y publica.

## MY DEVELOPER ACCOUNT

Hi JCçrdo8318!

Here's your personal Marvel Comics API information:

### Your public key

8529f0db9c296c5d884119cce2cf46eb

### Your private key

e30572272f8f5685602ea88821a57378afc25eeb

Read more about how to use your keys to sign requests. »

Your rate  
limit:

**3000**  
calls/day

Number of calls your  
application can make per day.

### Your authorized referrers

List any domains that can make calls to the Marvel Comics API  
using your API key here:

developer.marvel.com

delete

[add a new referrer](#)

Note: List the domain and path only - don't include "http" or  
other scheme designations. Only use the characters `a-z`,  
`0-9`, `.`, `_`, `-`, and `*`.

Read more about how to authorize referring domains in  
browser-based apps and web sites. »

### Tu clave pública

8529f0db9c296c5d884119cce2cf46eb

### Tu clave privada

e30572272f8f5685602ea88821a57378afc25eeb

3. Crear un repositorio en GitHub con el nombre "Practica01 – Consumo de APIs en la nube"

Juanfercho451 /

Practica01-Consumo-de-APIs-en-la-nube

Unwatch 1

Star 0

Fork 0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

...

main

Go to file

Add file

Code

About



Juanfercho451 Initial commit

hace 33 segundos



README.md

Initial commit

hace 33 segundos

Primer trabajo, aplicaciones distribuidas

Readme

README.md



## Practica01-Consumo-de-APIs-en-la-nube

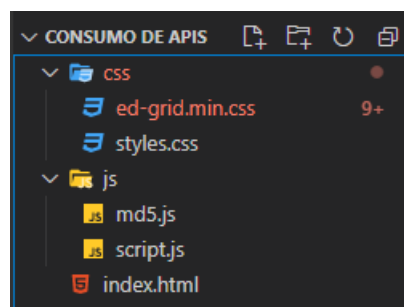
Releases

No releases published  
[Create a new release](#)

4. Desarrollar una aplicación con HTML + CSS + Javascript + Web Services para buscar información y visualizar

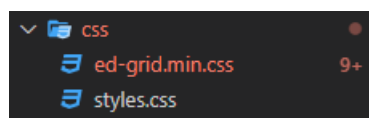
toda la información disponible a través de la API.


Creamos una aplicación llamada consumo de APIs, en donde creamos una carpeta css en donde tendremos todos nuestros estilos, una carpeta js en donde estará todo nuestro código de llamado de APIs, y un index.html, en donde nos reflejara las búsquedas.



Dentro de la carpeta css, tenemos dos archivos un archivo llamado ed-grid-min.css, el cual realizara la función automática de organizar bordes márgenes de todo el documento, este archivo nos descargamos de una base de datos de GRID.

La otra hoja de estilo css, esta nuestros diseños que realizamos a lo largo del documento.



 styles.css

```
body {  
  background-color: black;  
  color: white;  
}  
.main-title {  
  text-align: center;  
  color: white;  
}  
.marvel {  
  text-align: center;  
  color: whitesmoke;  
  text-decoration: double;  
}  
.hero-img {  
  position: relative;  
  overflow: hidden;  
}  
.hero-img:hover .descripcion {  
  top: 0;  
}  
.thumbnail {  
  display: block;  
  width: 100%;  
}  
.descripcion {  
  display: flex;  
  align-items: center;  
  color: white;  
  background: hsla(0, 0%, 20%, .8);  
  position: absolute;  
  width: 100%;  
  height: 100%;  
  top: -100%;  
  margin: 0;  
  padding: 1em;  
}  
#push,  
footer {  
  height: 100px;  
}
```

Dentro de la carpeta js, tenemos nuestras líneas de programación en donde el archivo md5.js, es un algoritmo que se utiliza como una función de codificación o huella digital de un archivo. De esta forma, a la hora de descargar un determinado archivo como puede ser un instalador, el código generado por el algoritmo, también llamado hash, de esta manera codificara nuestras llaves que utilizaremos, el siguiente archivo lo pueden descargar directamente de la web.

md5.js

```
var MD5 = function (string) {

    function RotateLeft(lValue, iShiftBits) {
        return (lValue<<iShiftBits) | (lValue>>>(32-iShiftBits));
    }

    function AddUnsigned(lX,lY) {
        var lX4,lY4,lX8,lY8,lResult;
        lX8 = (lX & 0x80000000);
        lY8 = (lY & 0x80000000);
        lX4 = (lX & 0x40000000);
        lY4 = (lY & 0x40000000);
        lResult = (lX & 0x3FFFFFFF)+(lY & 0x3FFFFFFF);
        if (lX4 & lY4) {
            return (lResult ^ 0x80000000 ^ lX8 ^ lY8);
        }
        if (lX4 | lY4) {
            if (lResult & 0x40000000) {
                return (lResult ^ 0xC0000000 ^ lX8 ^ lY8);
            } else {
                return (lResult ^ 0x40000000 ^ lX8 ^ lY8);
            }
        } else {
            return (lResult ^ lX8 ^ lY8);
        }
    }

    function F(x,y,z) { return (x & y) | ((~x) & z); }
    function G(x,y,z) { return (x & z) | (y & (~z)); }
    function H(x,y,z) { return (x ^ y ^ z); }
    function I(x,y,z) { return (y ^ (x | (~z))); }

    function FF(a,b,c,d,x,s,ac) {
        a = AddUnsigned(a, AddUnsigned(AddUnsigned(F(b, c, d), x), ac));
        return AddUnsigned(RotateLeft(a, s), b);
    };

    function GG(a,b,c,d,x,s,ac) {
        a = AddUnsigned(a, AddUnsigned(AddUnsigned(G(b, c, d), x), ac));
```

```

    return AddUnsigned(RotateLeft(a, s), b);
};

function HH(a,b,c,d,x,s,ac) {
    a = AddUnsigned(a, AddUnsigned(AddUnsigned(H(b, c, d), x), ac));
    return AddUnsigned(RotateLeft(a, s), b);
};

function II(a,b,c,d,x,s,ac) {
    a = AddUnsigned(a, AddUnsigned(AddUnsigned(I(b, c, d), x), ac));
    return AddUnsigned(RotateLeft(a, s), b);
};

function ConvertToWordArray(string) {
    var lWordCount;
    var lMessageLength = string.length;
    var lNumberOfWords_temp1=lMessageLength + 8;
    var lNumberOfWords_temp2=(lNumberOfWords_temp1-
(lNumberOfWords_temp1 % 64))/64;
    var lNumberOfWords = (lNumberOfWords_temp2+1)*16;
    var lWordArray=Array(lNumberOfWords-1);
    var lBytePosition = 0;
    var lByteCount = 0;
    while ( lByteCount < lMessageLength ) {
        lWordCount = (lByteCount-(lByteCount % 4))/4;
        lBytePosition = (lByteCount % 4)*8;
        lWordArray[lWordCount] = (lWordArray[lWordCount] | (string.charCodeAt
(lByteCount)<<lBytePosition));
        lByteCount++;
    }
    lWordCount = (lByteCount-(lByteCount % 4))/4;
    lBytePosition = (lByteCount % 4)*8;
    lWordArray[lWordCount] = lWordArray[lWordCount] | (0x80<<lBytePosition);
    lWordArray[lNumberOfWords-2] = lMessageLength<<3;
    lWordArray[lNumberOfWords-1] = lMessageLength>>>29;
    return lWordArray;
};

function WordToHex(lValue) {
    var WordToHexValue="",WordToHexValue_temp="",lByte,lCount;
    for (lCount = 0;lCount<=3;lCount++) {
        lByte = (lValue>>>(lCount*8)) & 255;
        WordToHexValue_temp = "0" + lByte.toString(16);
        WordToHexValue = WordToHexValue + WordToHexValue_temp.substr(WordToHe

```



```
xValue_temp.length-2,2);
    }
    return WordToHexValue;
};

function Utf8Encode(string) {
    string = string.replace(/\r\n/g, "\n");
    var utftext = "";

    for (var n = 0; n < string.length; n++) {

        var c = string.charCodeAt(n);

        if (c < 128) {
            utftext += String.fromCharCode(c);
        }
        else if ((c > 127) && (c < 2048)) {
            utftext += String.fromCharCode((c >> 6) | 192);
            utftext += String.fromCharCode((c & 63) | 128);
        }
        else {
            utftext += String.fromCharCode((c >> 12) | 224);
            utftext += String.fromCharCode(((c >> 6) & 63) | 128);
            utftext += String.fromCharCode((c & 63) | 128);
        }

    }

    return utftext;
};

var x=Array();
var k,AA,BB,CC,DD,a,b,c,d;
var S11=7, S12=12, S13=17, S14=22;
var S21=5, S22=9 , S23=14, S24=20;
var S31=4, S32=11, S33=16, S34=23;
var S41=6, S42=10, S43=15, S44=21;

string = Utf8Encode(string);

x = ConvertToWordArray(string);

a = 0x67452301; b = 0xEFCDAB89; c = 0x98BADCFE; d = 0x10325476;
```

```
for (k=0;k<x.length;k+=16) {
    AA=a; BB=b; CC=c; DD=d;
    a=FF(a,b,c,d,x[k+0], S11,0xD76AA478);
    d=FF(d,a,b,c,x[k+1], S12,0xE8C7B756);
    c=FF(c,d,a,b,x[k+2], S13,0x242070DB);
    b=FF(b,c,d,a,x[k+3], S14,0xC1BDCEEE);
    a=FF(a,b,c,d,x[k+4], S11,0xF57C0FAF);
    d=FF(d,a,b,c,x[k+5], S12,0x4787C62A);
    c=FF(c,d,a,b,x[k+6], S13,0xA8304613);
    b=FF(b,c,d,a,x[k+7], S14,0xFD469501);
    a=FF(a,b,c,d,x[k+8], S11,0x698098D8);
    d=FF(d,a,b,c,x[k+9], S12,0x8B44F7AF);
    c=FF(c,d,a,b,x[k+10], S13,0xFFFFF5BB1);
    b=FF(b,c,d,a,x[k+11], S14,0x895CD7BE);
    a=FF(a,b,c,d,x[k+12], S11,0x6B901122);
    d=FF(d,a,b,c,x[k+13], S12,0xFD987193);
    c=FF(c,d,a,b,x[k+14], S13,0xA679438E);
    b=FF(b,c,d,a,x[k+15], S14,0x49B40821);
    a=GG(a,b,c,d,x[k+1], S21,0xF61E2562);
    d=GG(d,a,b,c,x[k+6], S22,0xC040B340);
    c=GG(c,d,a,b,x[k+11], S23,0x265E5A51);
    b=GG(b,c,d,a,x[k+0], S24,0xE9B6C7AA);
    a=GG(a,b,c,d,x[k+5], S21,0xD62F105D);
    d=GG(d,a,b,c,x[k+10], S22,0x2441453);
    c=GG(c,d,a,b,x[k+15], S23,0xD8A1E681);
    b=GG(b,c,d,a,x[k+4], S24,0xE7D3FBC8);
    a=GG(a,b,c,d,x[k+9], S21,0x21E1CDE6);
    d=GG(d,a,b,c,x[k+14], S22,0xC33707D6);
    c=GG(c,d,a,b,x[k+3], S23,0xF4D50D87);
    b=GG(b,c,d,a,x[k+8], S24,0x455A14ED);
    a=GG(a,b,c,d,x[k+13], S21,0xA9E3E905);
    d=GG(d,a,b,c,x[k+2], S22,0xFCEFA3F8);
    c=GG(c,d,a,b,x[k+7], S23,0x676F02D9);
    b=GG(b,c,d,a,x[k+12], S24,0x8D2A4C8A);
    a=HH(a,b,c,d,x[k+5], S31,0xFFFA3942);
    d=HH(d,a,b,c,x[k+8], S32,0x8771F681);
    c=HH(c,d,a,b,x[k+11], S33,0x6D9D6122);
    b=HH(b,c,d,a,x[k+14], S34,0xFDE5380C);
    a=HH(a,b,c,d,x[k+1], S31,0xA4BEEA44);
    d=HH(d,a,b,c,x[k+4], S32,0x4BDECF A9);
    c=HH(c,d,a,b,x[k+7], S33,0xF6BB4B60);
    b=HH(b,c,d,a,x[k+10], S34,0xBEBFBC70);
    a=HH(a,b,c,d,x[k+13], S31,0x289B7EC6);
    d=HH(d,a,b,c,x[k+0], S32,0xEAA127FA);
```

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de


```
c=HH(c,d,a,b,x[k+3], S33,0xD4EF3085);
b=HH(b,c,d,a,x[k+6], S34,0x4881D05);
a=HH(a,b,c,d,x[k+9], S31,0xD9D4D039);
d=HH(d,a,b,c,x[k+12], S32,0xE6DB99E5);
c=HH(c,d,a,b,x[k+15], S33,0x1FA27CF8);
b=HH(b,c,d,a,x[k+2], S34,0xC4AC5665);
a=II(a,b,c,d,x[k+0], S41,0xF4292244);
d=II(d,a,b,c,x[k+7], S42,0x432AFF97);
c=II(c,d,a,b,x[k+14], S43,0xAB9423A7);
b=II(b,c,d,a,x[k+5], S44,0xFC93A039);
a=II(a,b,c,d,x[k+12], S41,0x655B59C3);
d=II(d,a,b,c,x[k+3], S42,0x8F0CCC92);
c=II(c,d,a,b,x[k+10], S43,0xFFEFF47D);
b=II(b,c,d,a,x[k+1], S44,0x85845DD1);
a=II(a,b,c,d,x[k+8], S41,0x6FA87E4F);
d=II(d,a,b,c,x[k+15], S42,0xFE2CE6E0);
c=II(c,d,a,b,x[k+6], S43,0xA3014314);
b=II(b,c,d,a,x[k+13], S44,0x4E0811A1);
a=II(a,b,c,d,x[k+4], S41,0xF7537E82);
d=II(d,a,b,c,x[k+11], S42,0xBD3AF235);
c=II(c,d,a,b,x[k+2], S43,0x2AD7D2BB);
b=II(b,c,d,a,x[k+9], S44,0xEB86D391);
a=AddUnsigned(a,AA);
b=AddUnsigned(b,BB);
c=AddUnsigned(c,CC);
d=AddUnsigned(d,DD);
}
```

```
var temp = WordToHex(a)+WordToHex(b)+WordToHex(c)+WordToHex(d);
```

```
return temp.toLowerCase();
```

```
}
```

El otro archivo llamado script.js contiene toda nuestra programación destinada al consumo de APIs.

 script.js X

```
const privatekey = 'e30572272f8f5685602ea88821a57378afc25eeb',
  publickey = '8529f0db9c296c5d884119cce2cf46eb',
  content = document.getElementById('content'),
  search = document.getElementById('search');
//Crearemos una coneccion
const getConection = () => {
  const ts = Date.now();
  const hash = MD5(ts + privatekey + publickey); //encriptamos nuestra llave
  const URL = `http://gateway.marvel.com/v1/public/characters?ts=${ts}&apikey=${publickey}`;
```

```

ey}&hash=${hash}`;
  fetch(URL) //proporciona una interfaz JavaScript para acceder y manipular partes del
canal HTTP
    .then(response => response.json())
    .then(response => {
      response.data.results.forEach(e => {
        drawHero(e); //llamamos a los datos encontrados
      });
    })
    .catch(e => console.log(e)); //en caso de tener algun error lo controlamos
};

const drawHero = e => {
  const image = `${e.thumbnail.path}/portrait_uncanny.${e.thumbnail.extension}`
  //creamos nuestra hoja de estilo para visualizar los datos encontrados
  const hero = `
<div class="hero ed-item 1-1-3" >
<br>
    <h5>${e.name}</h5>
    <div class="hero-img">
      <img class="thumbnail" src ="${image}">
      <p class="descripcion">${e.description}</p>

    </div>
  </div>
`;
  content.insertAdjacentHTML('beforeEnd', hero);
};

//con el siguiente metodo buscamos por medio del nombre a los superheroes
const searchHero = name => {
  const ts = Date.now();
  const hash = MD5(ts + privatekey + publickey);
  const hero = encodeURIComponent(name);
  const URL = `http://gateway.marvel.com/v1/public/characters?nameStartsWith=${hero}&ts
=${ts}&apikey=${publickey}&hash=${hash}`;
  fetch(URL)
    .then(response => response.json())
    .then(json => {
      json.data.results.map(item => {
        drawHero(item);
      });
    })
    .catch(e => console.log(e));
};

//hacemos que mediante un enter se pueda buscar el nombre del superheroe

```

```
search.addEventListener('keyup', e => {
  if (e.keyCode === 13) {
    content.innerHTML = '';
    searchHero(e.target.value.trim());
  }
});
getConection();
```

Y por último creamos nuestro index.html, con ayuda de un nav de bootstrap que nos ayuda con una plantilla para la búsqueda.

index.html X

```
<!DOCTYPE html>
<html lang="es">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/b
bootstrap.min.css" integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous"
>
  <link rel="stylesheet" href="css/ed-grid.min.css">
  <link rel="stylesheet" href="css/styles.css">
  <title>Heroes de MARVEL</title>
</head>

<body>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
  <br>
  <h1 class="main-title">MARVEL</h1>
  <div id="app">
    <nav class="navbar navbar-expand-lg navbar-light bg-dark">
      <div class="container-fluid">
        <a>MARVEL BATALLA DE SUPERHEROES</a>
        <div class="collapse navbar-collapse" id="navbarToggleDemo03">
          <ul class="navbar-nav mr-auto mt-2 mt-lg-0"></ul>
          <form class="form-inline my-2 my-lg-0" @submit.prevent="getHeroes">
            <input type="search" id="search" class="form-control mr-sm-2" v-
model="query" placeholder="Ingrese el superhere..." />
            <button class="btn btn-outline-success my-2 my-sm-
0" type="submit">Buscar Superheroe</button>
          </form>
```

```

    </div>
  </div>
</nav>
</div>
<br>
<div>

</div>
<br>
<div id="content" class="ed-container"></div>
<script src="js/md5.js"></script>
<script src="js/script.js"></script>
<div id="wrapper">
  Realizado por Juan Fernando Cordova
  <div id="push"></div>
</div>
<footer>
</footer>
</body>
</html>

```

- Realizar varios commits en la herramienta GitHub que demuestren el desarrollo de la aplicación.

main 1 branch 0 tags Go to file Add file Code

Juanfercho451 Add files via upload 5a0c098 now 3 commits

Consumo de APIs	Add files via upload	2 minutes ago
README.md	Initial commit	10 minutes ago

README.md

## Practica01-Consumo-de-APIs-en-la-nube

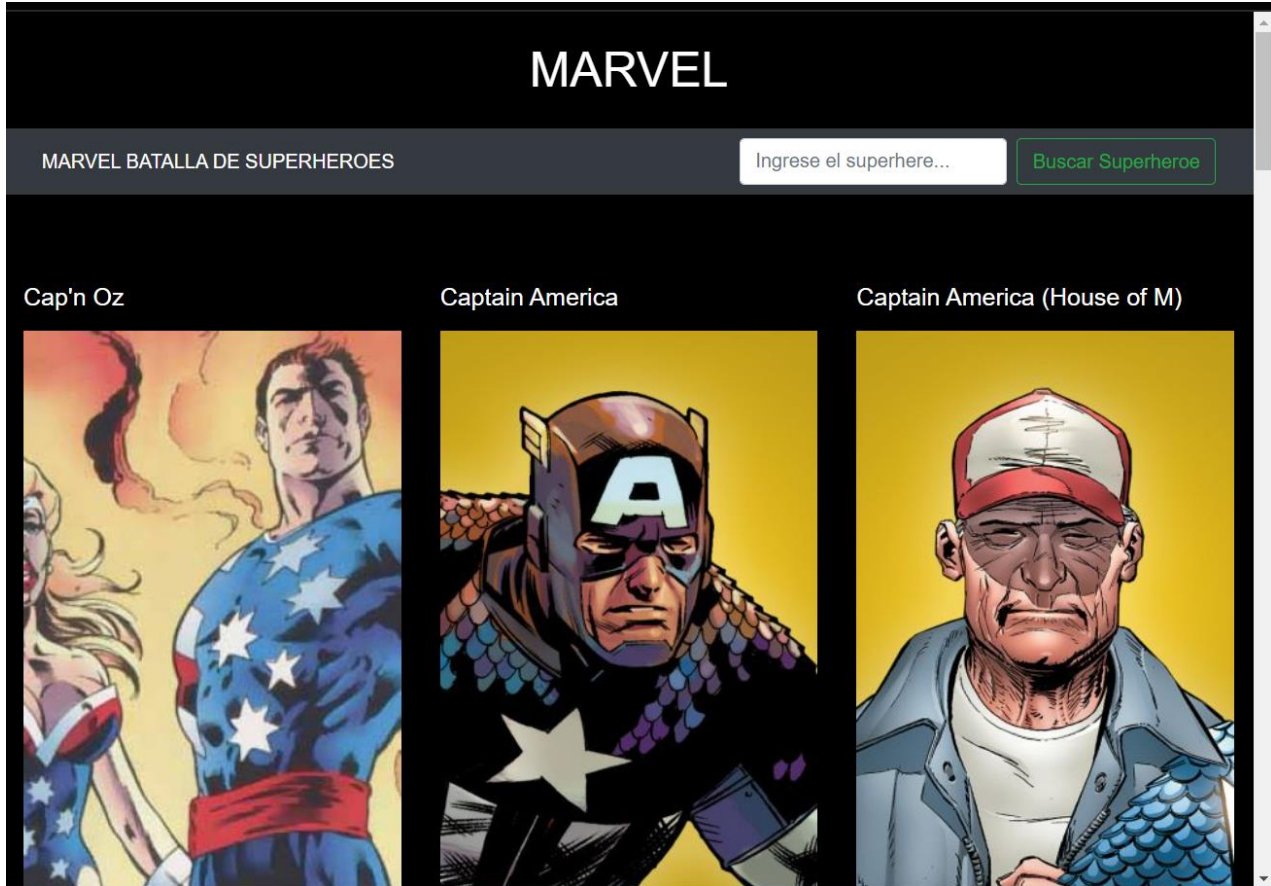
Primer trabajo, aplicaciones distribuidas

- Implementar el README del repositorio del proyecto con la misma información del informe de la práctica
- Subir al AVAC el informe del proyecto en formato \*.pdf. El informe debe contar con conclusiones apropiadas y la firma de cada estudiante

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de  
Simulación

**RESULTADO(S) OBTENIDO(S):**

- Identifica las diferentes arquitecturas Web para el desarrollo de aplicaciones.





# MARVEL

MARVEL BATALLA DE SUPERHEROES

hulk



Buscar Superheroe

Hulk



Hulk (HAS)



Hulk (LEGO Marvel Super Heroes)



## CONCLUSIONES:

- Los estudiantes podrán identificar arquitecturas web utilizando servicios en la nube. Así como también, podrán consumir APIs y manipular objetos JSON.

## RECOMENDACIONES:



	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

- Aplicar conceptos de interacción humano máquina para el desarrollo de la GUI.

**Docente:** Ing. Gabriel León Paredes, PhD.



**Firma:**