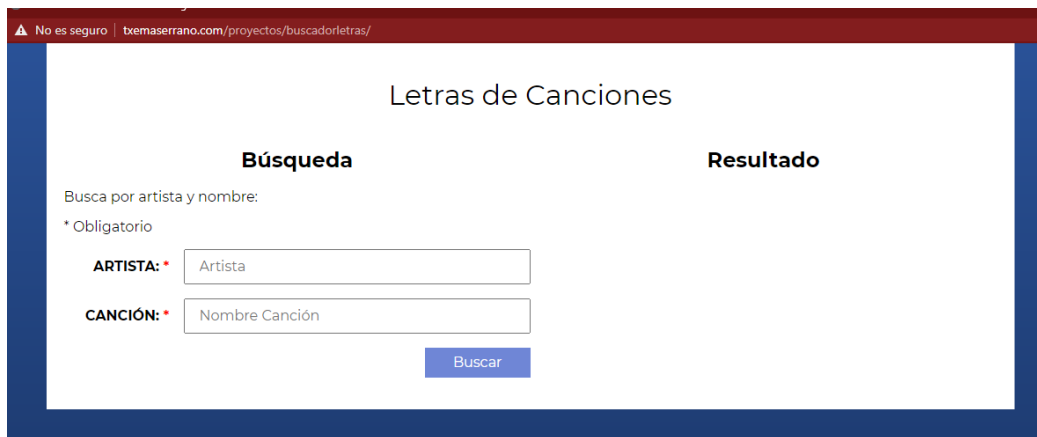
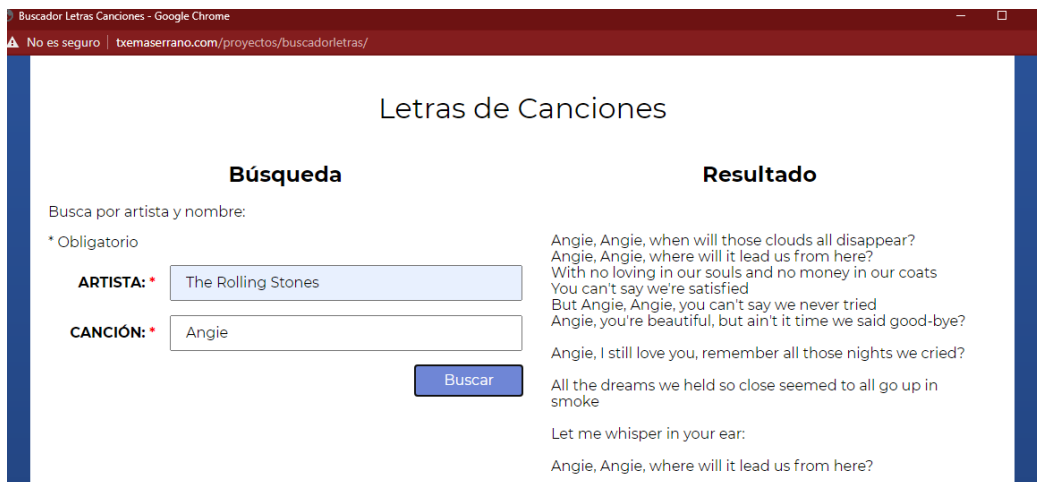


## Buscador de canciones con Fetch-API



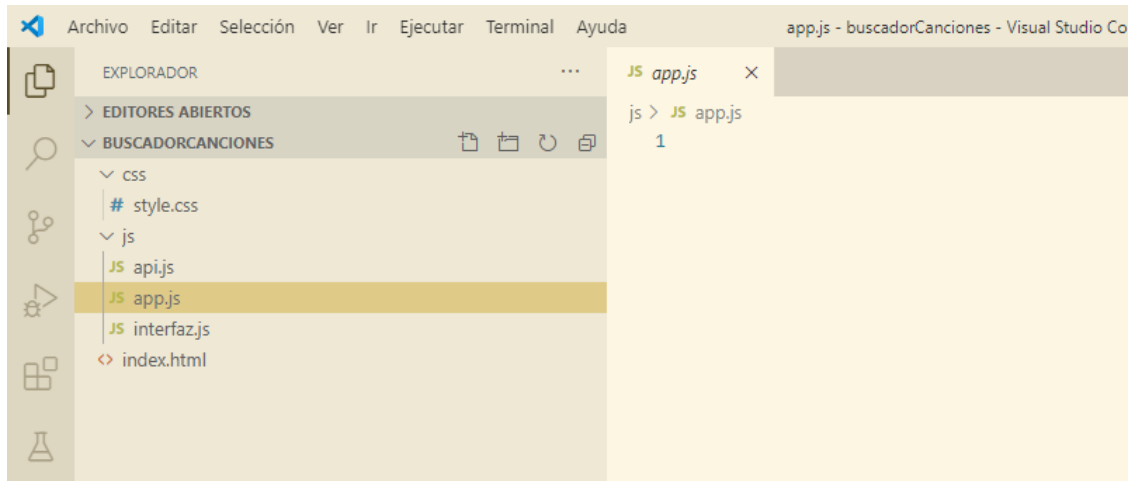
The screenshot shows a web browser window with the URL `txemaserrano.com/proyectos/buscadorletras/`. The page title is "Letras de Canciones". It features two main sections: "Búsqueda" (Search) and "Resultado" (Result). In the "Búsqueda" section, there is a prompt "Busca por artista y nombre:" followed by a note "\* Obligatorio". Below this are two input fields: "ARTISTA: \*" with the placeholder text "Artista" and "CANCIÓN: \*" with the placeholder text "Nombre Canción". A blue "Buscar" button is positioned below the "CANCIÓN" field.

Si rellenamos el formulario:



This screenshot shows the same web application after a search. The "Búsqueda" section now contains the filled-in values: "ARTISTA: \*" is "The Rolling Stones" and "CANCIÓN: \*" is "Angie". The "Resultado" section displays the lyrics for the song "Angie" by The Rolling Stones. The lyrics are: "Angie, Angie, when will those clouds all disappear? Angie, Angie, where will it lead us from here? With no loving in our souls and no money in our coats You can't say we're satisfied But Angie, Angie, you can't say we never tried Angie, you're beautiful, but ain't it time we said good-bye? Angie, I still love you, remember all those nights we cried? All the dreams we held so close seemed to all go up in smoke Let me whisper in your ear: Angie, Angie, where will it lead us from here?".

## Paso 1. Descargar plantilla:



Como vamos a utilizar módulos declaramos el script de tipo “module” en index.html:

```
45
46   <script src="js/app.js" type="module"></script>
47 </body>
48 </html>
```

## Paso 2. Crear la interfaz.

El primer fichero que trabajaremos será interfaz.js, para colocar una serie de selectores:

```
<> index.html  JS interfaz.js
js > JS interfaz.js > ...
1  export const formularioBuscar = document.querySelector("#formulario-buscar"),
2  // document.getElementById("formulario-buscar");
3  divBuscar = document.querySelector("#buscar"),
4  divMensajes = document.querySelector("#mensajes"),
5  divResultado = document.querySelector("#resultado"),
6  headingResultado = document.querySelector(".letra-resultado h2");
```

En **app.js** importaremos esta interfaz:

```
da  app.js - buscadorCanciones - Visual Studio Code
<> index.html  JS interfaz.js  JS app.js  X
js > JS app.js
1  import * as UI from './interfaz.js';
2
3  console.log(UI);
```

Salida:

```
Live reload enabled. index.html:77
▼ Module 1 app.js:3
  divBuscar: (...)
  divMensajes: (...)
  divResultado: (...)
  formularioBuscar: (...)
  headingResultado: (...)
  Symbol(Symbol.toStringTag): "Module"
```

Asociamos una función al formulario para cuando se pulse el botón y extraemos los parámetros de la búsqueda: artista y canción:

```
index.html JS interfaz.js JS api.js JS app.js x
js > JS app.js > buscarCancion
1 import * as UI from './interfaz.js';
2
3 console.log(UI);
4
5 // Utilizamos el formulario buscar:
6
7 UI.formularioBuscar.addEventListener('submit', buscarCancion);
8
9 function buscarCancion(e) {
10   e.preventDefault();
11
12   const artista = document.querySelector("#artista").value;
13   const cancion = document.querySelector("#cancion").value;
14
15   // Validamos los datos
16
17   if (artista == "" || cancion == "") {
18     // El usuario dejó un campo vacío, mostrar mensaje de error
19
20     UI.divMensajes.textContent = "Error: todos los campos son obligatorios";
21     UI.divMensajes.classList.add("error");
22   }
23 }
```

Comprobamos que no estén vacíos:

## Letras de Canciones

### Búsqueda

Busca por artista y nombre:

\* Obligatorio

**ARTISTA:**

**CANCIÓN:**

Buscar

Error: todos los campos son obligatorios

Para hacer que el mensaje se visualice durante 3 segundos únicamente:

```
UI.divMensajes.textContent = "Error: todos los campos son obligatorios";
UI.divMensajes.classList.add("error");

setTimeout(() => {
  UI.divMensajes.innerHTML = '';
  UI.divMensajes.classList.remove('error');
}, 3000);
```

### Paso 3. Conectar con el servicio REST.

Si no hay error, consultamos la API que nos descarga las canciones desde el servicio REST lyrics.com

```
< index.html JS interfaz.js JS api.js JS app.js x
js > JS app.js > buscarCancion
6
7 UI.formularioBuscar.addEventListener('submit', buscarCancion);
8
9 function buscarCancion(e) {
10   e.preventDefault();
11
12   const artista = document.querySelector("#artista").value;
13   const cancion = document.querySelector("#cancion").value;
14
15   // Validamos los datos
16
17   if (artista == "" || cancion == "") {
18     // El usuario dejó un campo vacío, mostrar mensaje de error
19
20     UI.divMensajes.textContent = "Error: todos los campos son obligatorios";
21     UI.divMensajes.classList.add("error");
22
23     setTimeout(() => {
24       UI.divMensajes.innerHTML = '';
25       UI.divMensajes.classList.remove('error');
26     }, 3000);
27   } else {
28
29     // Consultamos la API
30     console.log('Consultando la API');
31
32   }
```

El esquema de nuestra API es:

```
< index.html JS interfaz.js JS api.js JS app.js
js > JS api.js > API
1 // Importamos la interfaz
2
3 import * as UI from './interfaz.js';
4
5 class API {
6
7 }
8
9 export default API;
```

Creamos el constructor de la clase:

```

4
5 class API {
6
7   constructor(artista, cancion){
8     this.artista = artista;
9     this.cancion = cancion;
10  }
11
12 }
13
14 export default API;

```

Esta clase hay que importarla en app.js:

```

index.html  JS interfaz.js  JS api.js  JS app.js
> JS app.js > ...
1  import * as UI from './interfaz.js';
2  import API from './api.js';
3
4  console.log(UI);
5
6  // Utilizamos el formulario buscar:
7
8  UI.formularioBuscar.addEventListener('submit', buscarCancion);

```

Y para realizar la búsqueda con los elementos del formulario, creamos una función **consultarAPI()** en la API:

```

<> index.html  JS interfaz.js  JS api.js  X  JS app.js
js > JS api.js > ...
1  // Importamos la interfaz
2
3  import * as UI from './interfaz.js';
4
5  class API {
6
7    constructor(artista, cancion) {
8      this.artista = artista;
9      this.cancion = cancion;
10   }
11
12   consultarAPI() {
13     console.log('Consultando la API...');
14   }
15
16 }
17
18 export default API;

```

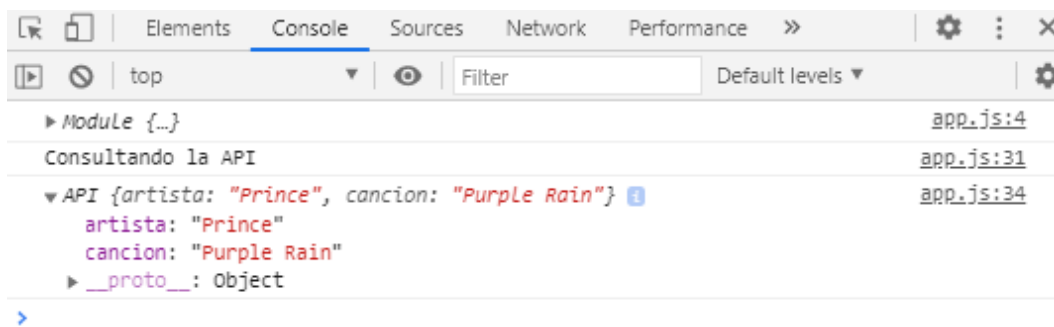
La invocamos con nuestro objeto de búsqueda desde app.js:

```

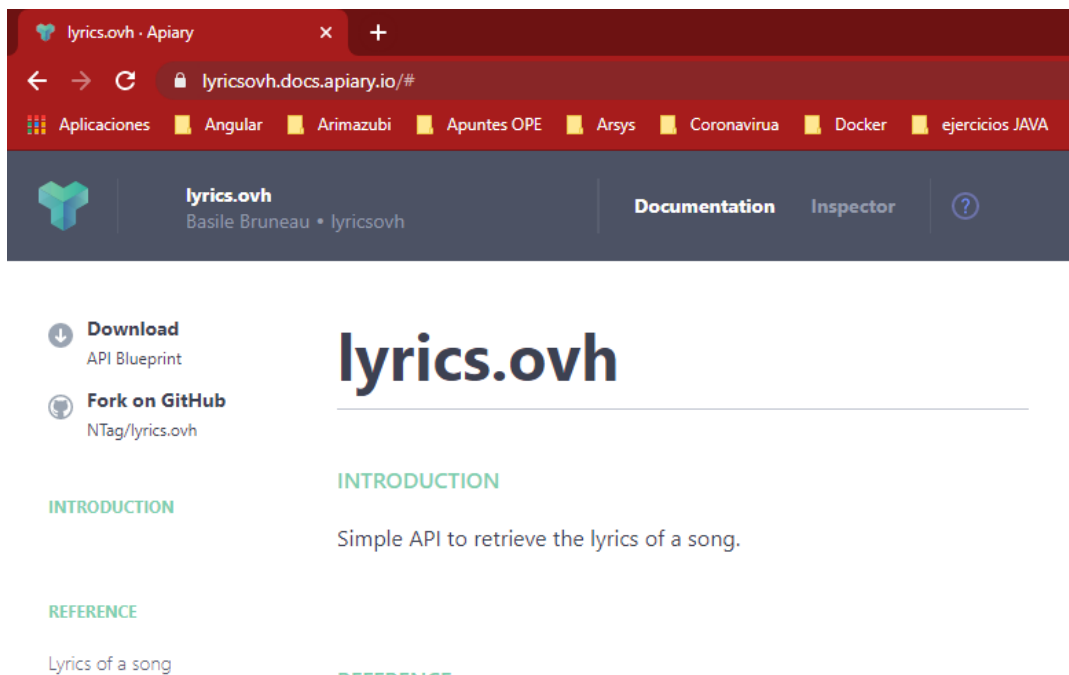
27     }, 3000);
28   } else {
29
30     // Consultamos la API
31     console.log('Consultando la API');
32     const busqueda = new API(artista, cancion);
33
34     busqueda.consultarAPI();
35     console.log(busqueda);
36   }
37 }

```

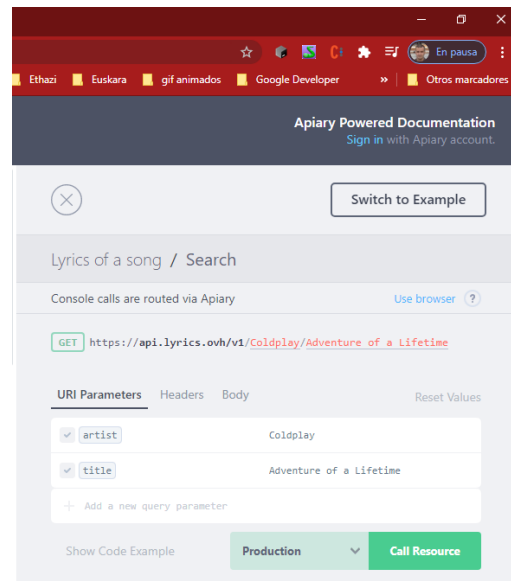
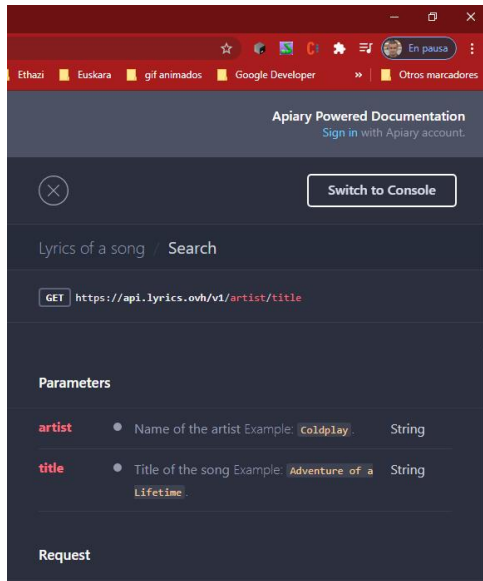
La salida que se obtiene:



Para retornar las letras de las canciones, utilizamos el siguiente servicio REST:



Cuya forma de invocar:



<https://api.lyrics.ovh/v1/artist/title>

Parámetros:

- Artista (string): nombre del artista, ejemplo: Coldplay.
- Title (string): nombre de la canción, ejemplo: Adventure of a Lifetime.

Si lo invocamos:

Request

**GET** `https://private-anon-1e370afdc9-lyricsovh.apiary-proxy.com/v1/Coldplay/Adventure%20of%20a%20Lifetime`

Response

**200**

**Response Headers**

```
content-type: application/json; charset=utf-8
content-length: 1361
x-powered-by: express
access-control-allow-origin: *
etag: w/"551-eftq7nnlbwxj35m+zvvbbloqgxu"
... *
```

**Response Body**

```
{
  "lyrics": "Turn your magic on, Umi she'd say
Everything you want's a dream away
We are legends, every day
That's what she told me
Turn your magic on, to me she'd say
..... *
```

#### Paso 4. Reconfiguramos nuestra API con los parámetros del formulario.

Copiamos la url en nuestra API:

```

5 class API {
6
7   constructor(artista, cancion) {
8     this.artista = artista;
9     this.cancion = cancion;
10  }
11
12   consultarAPI() {
13     const url = `https://api.lyrics.ovh/v1/artist/title`;
14
15     fetch(url)
16       .then(respuesta => respuesta.json());
17   }
18
19 }

```

Sustituimos artista y title por sus valores en la API:

```

2
3 import * as UI from './interfaz.js';
4
5 class API {
6
7   constructor(artista, cancion) {
8     this.artista = artista;
9     this.cancion = cancion;
10  }
11
12   consultarAPI() {
13     const url = `https://api.lyrics.ovh/v1/${this.artista}/${this.cancion}`;
14
15     fetch(url)
16       .then(respuesta => respuesta.json())
17       .then(resultado => {
18         console.log(resultado);
19       });
20   }
21
22 }
23

```

La salida:

Curso: Desarrollo Web en Entorno de Node.js x Buscador de Letras - JSModerno x +

127.0.0.1:5501/index.html?#

Aplicaciones Angular Arimazubi Apuntes OPE Arsys Coronavirus Docker ejercicios JAVA Ethazi Otros marcadores

## Letras de Canciones

### Búsqueda

Busca por artista y nombre:

\* Obligatorio

ARTISTA:

CANCIÓN:

Buscar

Module {...} BRD.js:14

consultando la API BRD.js:131

API {artista: "Prince", cancion: "Purple Rain"} BRD.js:135

{lyrics: "I never meant to cause you any sorrow I never meant to cause you any p..."} BRD.js:138

I never meant to cause you any sorrow I never meant to cause you any pain I only wanted to one time to see you laughing I only wanted to see you laughing in the purple rain Purple rain, purple rain Purple rain, purple rain I only wanted to see you Bathing in the purple rain I never wanted to be your weekend lover I only wanted to be some kind of friend Baby, I could never steal you from another It's such a shame our friendship had to end Purple rain, purple rain Purple rain, purple rain Purple rain, purple rain I only wanted to see you Bathing in the purple rain Honey, I know, I know I know times are changing It's time we all reached out For something new, that means you too You say you want a leader But you can't seem to make up your mind I think you better close it And let me guide you to the purple rain Purple rain, purple rain Purple rain, purple rain Purple rain, purple rain I only wanted to see you Bathing in the purple rain



### Paso 5: mostrar el resultado.

Esta salida la mostraremos en un div:

```
consultarAPI() {
  const url = `https://api.lyrics.ovh/v1/${this.artista}/${this.cancion}`;

  fetch(url)
    .then(respuesta => respuesta.json())
    .then(resultado => {

      const { lyrics } = resultado;

      UI.divResultado.textContent = lyrics;
    });
}
```

La salida:

## Letras de Canciones

### Búsqueda

Busca por artista y nombre:

\* Obligatorio

**ARTISTA:** \*

**CANCIÓN:** \*

Angie, Angie, when will those clouds all disappear?  
 Angie, Angie, where will it lead us from here?  
 With no loving in our souls and no money in our coats  
 You can't say we're satisfied  
 But Angie, Angie, you can't say we never tried  
 Angie, you're beautiful, but ain't it time we said good-bye?

Angie, I still love you, remember all those nights we cried?

All the dreams we held so close seemed to all go up in smoke

Let me whisper in your ear:

Angie, Angie, where will it lead us from here?

Module (-)

Consultando la API

API {artista: "The Rolling Stones", cancion: "Angie"}

Si queremos poner un título:

```
const { lyrics } = resultado;

UI.divResultado.textContent = lyrics;
UI.headingResultado.textContent = `${this.cancion} de ${this.artista}`;
```

Salida:

### Búsqueda

Busca por artista y nombre:

\* Obligatorio

**ARTISTA:** \*

**CANCIÓN:** \*

### Purple Rain de Prince

I never meant to cause you any sorrow  
 I never meant to cause you any pain  
 I only wanted to one time to see you laughing  
 I only wanted to see you  
 Laughing in the purple rain

Purple rain, purple rain

Si ponemos una canción que no exista:

## Letras de Canciones

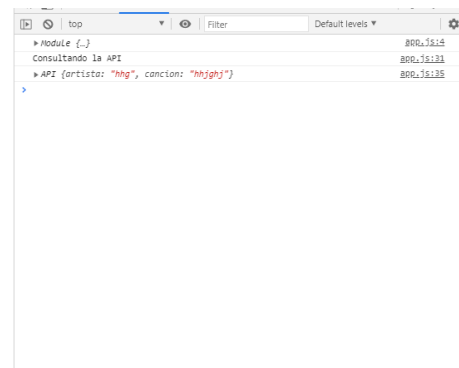
### Búsqueda

Busca por artista y nombre:  
\* Obligatorio

**ARTISTA:**

**CANCIÓN:**

La canción no existe: prueba otra búsqueda



Le ponemos un `timeOut` para mostrar el mensaje sólo 3 segundos y desaparecerá el mensaje limpiando el formulario.