



# HACK A MUSIC ~ Entrega #1

## ◆ 1. Objetivo

Construir una plataforma de música utilizando **Vuejs** y la **API** de **Lastfm**.

## ◆ 2. Registro en la API

Esta API necesita que te registres en su página. Su página es <https://www.last.fm/api/?lang=es&> .



## Last.fm Web Services

[API Introduction](#)

The Last.fm API allows anyone to build their own programs using Last.fm data, whether they're on the web, the desktop or mobile devices. [Find out more](#) about how you can start exploring **the social music playground** or just browse the list of methods below.

### Getting Started

Anyone is free to use the Last.fm API. Here's what you need to get going:

1. [Get an API account](#)
2. [Read the documentation](#)
3. [Join the API group](#)

### Featured Applications

[See more at build.last.fm »](#)



#### vBulletin Last.FM page

Embeds a Last.FM Group playlist into your vBulletin forum, on a new page.



#### Chrome Last.fm Scrobber

Scrobble music videos all around the web with this Google Chrome extension!



#### Music Quilt Screensaver

Get your own personalized screensaver here – brought to you by Motorola.

## Login

Username

Password

Let me in!

[Forgot your username or password?](#)

Don't have a profile? [Sign up for FREE](#)

Creáis vuestra cuenta.

Os aparecerá la siguiente página donde no podéis elegir nada, NO LA CERRÉIS.

## Create API account

---

Contact email

desireemcarmona@gmail.com

Application name

Application description

Verificad la cuenta a través del mail.

## Account verified

Thank you for verifying your email address. You can now [continue to use Last.fm](#).

Actualizad la página de antes para que os deje escribir y rellenad:

**\*No escribáis el nombre de la aplicación con ningún espacio.**

Application name	hackamusic
Application description	<div>Hack a music application for educational purposes.</div>
Callback URL	<div>https://hackamusic.com</div> <p>When using our <a href="#">web-based authentication</a> this callback URL is sent authentication tokens (<a href="#">more info</a>). This field isn't used for desktop or mobile authentication.</p>
Application homepage	<div>https://hackamusic.com</div>

Manage Cookies

Y clicáis '**Submit**'.

Os saldrá algo como lo siguiente:

# API account created

---

Here are the details of your new API account.

Application name	hackamusic
API key	faf3387042a6e4bb39034a9eed88d84d
Shared secret	bd05acd04b11dc86b737fa49e3440d11
Registered to	desiremcarmona

Quédate con esa página, copia-pegas esos datos en un `.txt` o lo que sea, los necesitarás para utilizar la API. Más adelante vemos dónde ponerlo.

## ◆ 3. Setup del proyecto

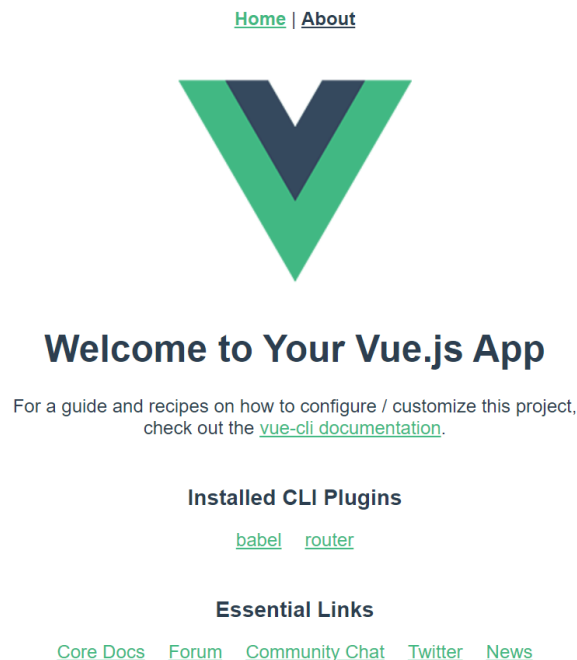
Crea el nuevo proyecto de Vuejs, se llamará HACK A MUSIC ( `vue create hackamusic` ).

Elegirás:

- Manually select.
- Babel + Router.
- n
- In dedicated config files
- n

Una vez finalice la instalación, levanta el server de la aplicación con `npm run serve` y visita la ruta correcta que aparece en consola para ver tu proyecto (en mi caso `localhost:8080` ).

Debería aparecerte:



## ◆ 4. Importando la API y sus datos al proyecto

Dentro del proyecto, dentro de la carpeta `src`, vas a crear una nueva carpeta llamada `api`.

Dentro, crearás un fichero llamado `config.js` donde vas a especificar y exportar tus datos de autenticación de la api. Primero, escribe la siguiente estructura:

```
export default {  
  appName: '', // NOMBRE DE LA APP  
  apiKey: '', // TU KEY PERSONAL  
  secret: '', // TU KEY SECRETA  
  registeredTo: '' // TUS DATOS  
}
```

Estos datos tendrás que rellenarlos con los datos que te dio la API de Lastfm (no te preocupes, están por orden), así que:

```
export default {  
  appName: 'hackamusic',  
  apiKey: 'e59d26bb7f3aee6eeb4ca7f0e2c35f60',  
  secret: '85a85d0f6fee8f4c051f248f41bbec77',  
  registeredTo: 'desiremcarmona'  
}
```


*Recordatorio: esos son mis datos, tú debes rellenarlos con los datos que tú has conseguido al registrarte en la API, que probablemente serán algo diferentes.*

Volviendo a la API, primero vamos a pedir que la API nos envíe los **TOP ARTISTS** por región (en este caso España), la documentación la tienes aquí:

<https://www.last.fm/api/show/geo.getTopArtists> .

Para acceder a esa llamada de la API vas a necesitar su **key**, la cual acabamos de indicar en el código de arriba.

Dentro de la carpeta `api` que has creado, crea un nuevo archivo llamado `index.js` . Es donde crearemos las llamadas. Primero debes importar la configuración del archivo `config` , y definir una constante donde guardarás tu **API KEY**:



```
import config from './config.js'

const apiKey = config.apiKey
```

La ruta de la API que vamos a llamar es: [https://ws.audioscrobbler.com/2.0/?method=geo.gettopartists&country=spain&api\\_key=YOUR\\_API\\_KEY&format=json](https://ws.audioscrobbler.com/2.0/?method=geo.gettopartists&country=spain&api_key=YOUR_API_KEY&format=json) , y como ves resaltado en **color**, debemos cambiar `YOUR_API_KEY` por nuestra constante creada en `apiKey` , de modo que la URL de la API quedaría así:

[https://ws.audioscrobbler.com/2.0/?method=geo.gettopartists&country=spain&api\\_key="+apiKey+"&format=json](https://ws.audioscrobbler.com/2.0/?method=geo.gettopartists&country=spain&api_key=)

Crea una constante para guardar esta URL en una constante llamada `URL_GEO` :



```
import config from './config.js'

const apiKey = config.apiKey
const URL_GEO = "https://ws.audioscrobbler.com/2.0/?
method=geo.gettopartists&country=spain&api_key="+apiKey+"&format=json"
```

(No debería haber espacios, se ve así porque la URL es demasiado larga).

Vamos a utilizar de nuevo la librería **AXIOS** para hacer llamadas. Instala AXIOS a través de la consola matando el server y escribiendo `npm i --save axios`.

A continuación debes importarla también en `index.js`:


```
import config from './config'
const axios = require('axios').default;
```

Una vez importado **AXIOS**, vamos a **crear una función para coger los topartists de Lastfm de España**, por ahora para que salga en la consola:



```
function getArtists() {  
  axios.get(`${URL_GEO}`)  
    .then(function (response) {  
      // handle success  
      console.log(response);  
    })  
    .catch(function (error) {  
      // handle error  
      console.log(error);  
    })  
}
```

A continuación, exporta esta función:



```
export default {  
  getArtists  
}
```

Por ejemplo en la vista `Home.vue`, importa tu api:

```
import api from '@api/index.js'
```

Y finalmente, crea un hook `created` y llama a la función:

```
created( ){  
  api.getArtists( )  
}
```

Actualiza tu aplicación y comprueba que en consola te sale la llamada.

Si no te sale repasa los pasos, si te sale, continua al siguiente punto.

## ◆ 5. Mejorando la API

Vamos a optimizar un poco el código de la API.

Te habrás fijado en que en la URL de `URL_GEO` estamos añadiendo la URL base de la API (`ws.audioscrobbler...`), cosa que no debería ser así. Vamos a coger la URL base y la vamos a separar en otra variable `const` llamada `URL_BASE`, mientras que borramos esa parte de la `const URL_GEO`:

```
import config from './config.js'
const axios = require('axios').default;

const apiKey = config.apiKey
const BASE_URL = "https://ws.audioscrobbler.com/"
const URL_GEO = "2.0/?method=geo.gettopartists&country=spain&api_key="+apiKey+"&format=json"
```

Convierte en asíncrona la función `getArtists` y añade un `try` - `catch` para mejorar su funcionamiento:

```
async function getArtists() {
  try {
    const response = await axios.get(`${BASE_URL}${URL_GEO}`);
    console.log(response);
  } catch (error) {
    console.error(error);
  }
}
```

Comprueba que no te has cargado nada y que sigue apareciendo la respuesta de la API en consola.

## ◆ 6. Mostrando los artistas en la página

Primero, es necesario pedir a la función de la API que esta vez nos haga un `return` de la respuesta:

```
async function getArtists() {
  try {
    const response = await axios.get(`${BASE_URL}${URL_GEO}`);
    return response
  } catch (error) {
    console.error(error);
  }
}
```

Volvemos a `Home.vue`, y dentro del objeto `data` crearemos un array de artistas:

```
data(){
  return {
    artists: []
  }
}
```

Y es momento de recoger los datos de la API y meterlos dentro de `artists`. Iremos a la llamada que hemos creado en `created` y añadiremos la lógica para guardar los artistas en la variable del front:

```

created(){
  api.getArtists().then(response => (this.artists = response.data.topartists.artist))
}

```

Accedemos a `response.data.topartists.artist` porque ahí, de entre toda la respuesta de la API, es donde está la información que nos interesa. Esto siempre puedes comprobarlo a través del `console.log` que pedimos a la llamada a la API:



Para mostrarlos por pantalla será necesario utilizar la directiva `v-for`. Úsala como quieras, en un `p`, `ul`, `ol`, etc.

```

<ul>
  <li v-for="artist in artists" :key="artist.id">
    ARTISTA: {{ artist.name }}
    OYENTES: {{ artist.listeners }}
  </li>
</ul>

```

En pantalla deberías estar viendo una lista de nombre de artistas y sus oyentes.

¡Genial! ¡Ya estás mostrando los datos de la API en tu página!

## ◆ 7. Recopilando

- Has configurado la API junto a tu APIKEY de autenticación.
- Has optimizado el código de la API.
- Has creado la primera llamada a la API del proyecto ( `getArtists()` ).
- Has mostrado ambas respuestas por pantalla: importando la api, llamando sus métodos en el hook `created` y guardando la información en una variable para mostrarla por pantalla a través de una directiva `v-for`.

## ◆ 8. Problema con las imágenes

Lastfm hace un tiempo dejó de proporcionar imágenes de los artistas. En su lugar, sale una fotografía de una estrella blanca. Si colocas la línea de código

```

```

por ejemplo en el `v-for` de los artistas, verás que sale esa imagen.

Para solucionar esto puedes hacer manualmente un array de imágenes de cada artista (fijándote en el orden que salen en la API) y renderizar un `v-for` de ese array de imágenes manual dentro del `v-for` de artistas.

El problema de esto es que los datos de la API (al ser top artistas del momento y demás), pueden cambiar rápidamente, y es posible que de un día para otro las imágenes que has puesto ya no se correspondan con el orden de la respuesta de la API.

Para el ejercicio vale con que salga la foto de la estrella blanca, ¡tú decides!

## ◆ 9. Ejercicios del proyecto (OBLIGATORIOS)



Crea 2 funciones más en el archivo `index.js` de la api, una para coger los **toptracks** `getTopTracks()` ( <https://www.last.fm/api/show/geo.getTopTracks> ) y otra para coger los **toptags** `getTopTags()` ( <https://www.last.fm/api/show/chart.getTopTags> ).

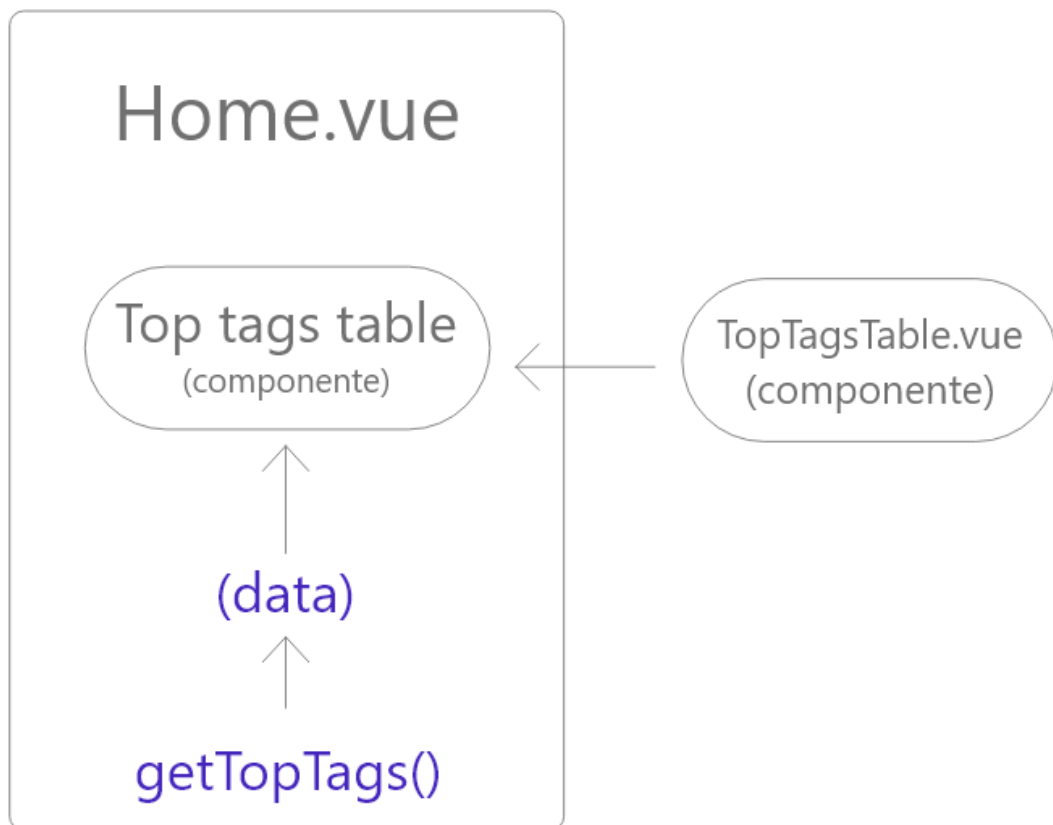


Las vistas obligatorias del proyecto son: `Home.vue` , `About.vue` , `TopArtists.vue` y `TopTracks.vue` .



En `Home.vue` deben mostrarse los **Top tags** de **Lastfm**, recuerda utilizar un componente para mostrarlos, y que los datos los conseguirás a través de la llamada a la API de `getTopTags()` . 🙋

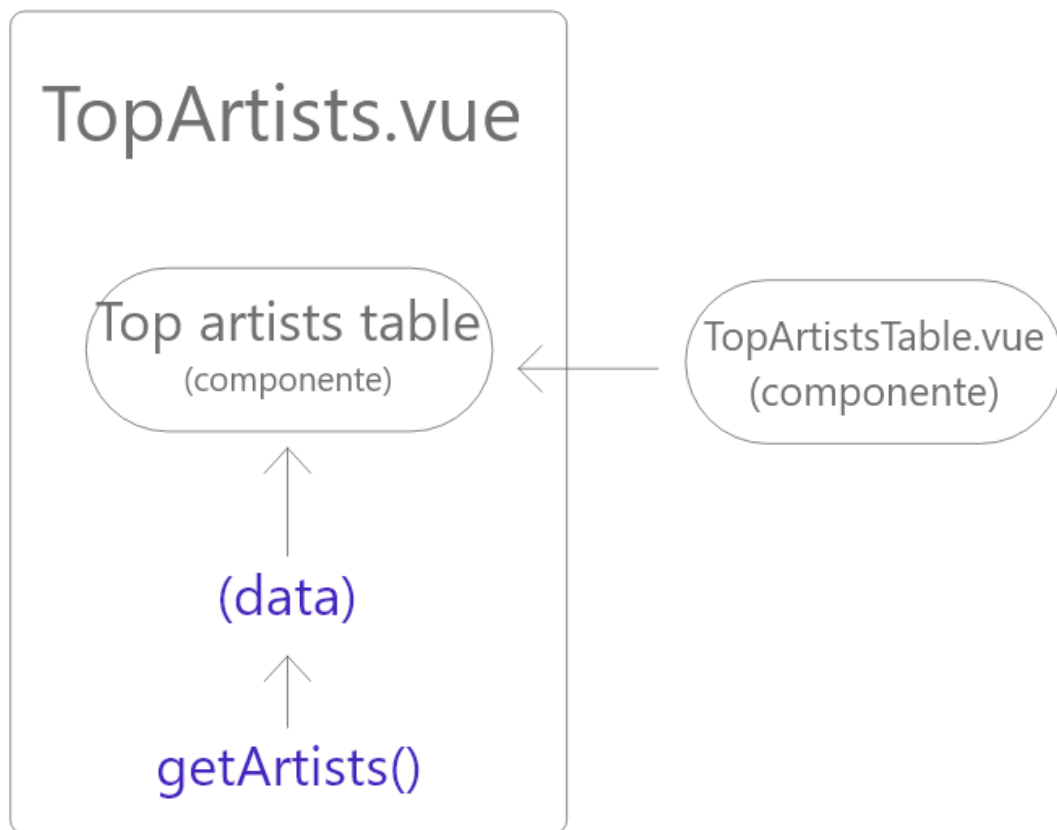




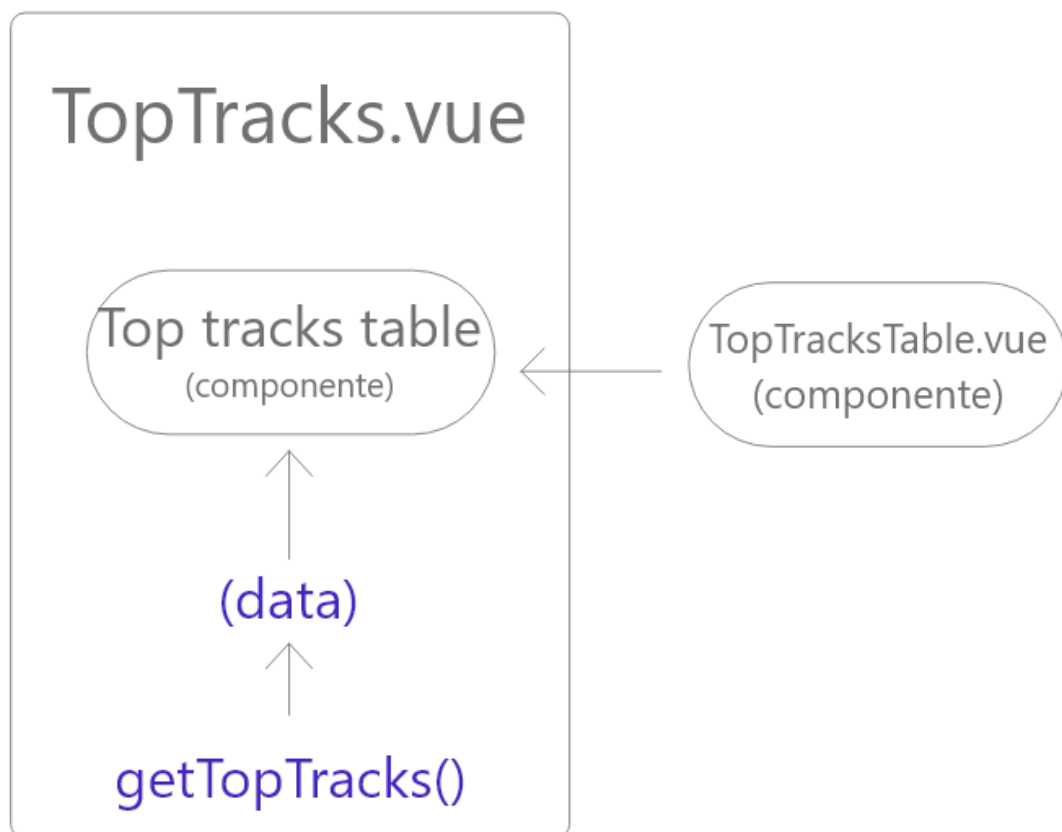
En `About.vue` debes mostrar información sobre ti y sobre con qué has hecho el proyecto (la API de Lastfm, librería AXIOS, SweetAlert, Vue-Headful...), en definitiva todo lo que uses.



En `TopArtists.vue` debes mostrar los **artistas top** de **LastFm** pero utilizando un **componente** adecuado (debes crearlo...). Utiliza las props necesarias. La lógica debe estar en la vista, no en el componente. 📝



En `TopTracks.vue` debes mostrar los **tracks top** de **LastFm**, pero utilizando un componente adecuado (debes crearlo...). Utiliza las props necesarias. La lógica debe estar en la vista, no en el componente. 📌



Limpia el proyecto de vistas y componentes que no se utilicen.  
(Ningún *HelloWorld* sin usar por ahí...) 🙄.



Intenta mostrar tantos datos de los artistas y tags como puedas.  
Consulta la respuesta que te llega de la API para saber qué más  
puedes mostrar aparte de lo que has visto en la guía (Nombre, Artista,  
Oyentes, Imagen...).



Comenta y documenta tu código.



Configura de manera correcta el Router y los links (router-links) del proyecto para que funcionen correctamente y de forma dinámica con el 'name' y no con el path.



No hay componentes obligatorios, debes utilizar componentes y variables, como tú creas que es mejor su disposición y organización. Intenta optimizar tanto como puedas los HTML. 🌿



Utiliza un diseño original y practica CSS. Puedes inspirarte en otras plataformas de música como Spotify o alguna otra de radio online. HAZ EL DISEÑO LO ÚLTIMO, PRIMERO HAZ QUE TODO FUNCIONE 😊



Optimiza el código tanto como sea posible y siempre que te sea posible.



Configura una página de **error 404** en tu proyecto para recoger todas las rutas que no están creadas en el router.

## ◆ 10. Herramientas e implementaciones **opcionales**

Puedes utilizar las siguientes *herramientas opcionales/sugerencias* para completar tu proyecto:

**SweetAlert2** → <https://sweetalert2.github.io/> para crear alguna alerta interesante para la persona usuaria.

**Vue-Headful** → <https://www.npmjs.com/package/vue-headful> para cambiar dinámicamente los títulos de cada página del proyecto.

**CSS Loader Spinner** → <https://loading.io/css/> para crear una animación de "loading" mientras los datos de la API van cargando en la vista. Cada animación tiene su código css y html, podéis implementarlos en el proyecto utilizando un componente y alguna directiva como `v-if` o `v-show` , ¡investigad! 🤔

💡 Implementa alguna función para buscar por artista a través de una barra de búsqueda (*input*) y devolver la información por pantalla. Puedes colocar esta función en la vista `TopArtists.vue` .