



EXERCISES QUE TRABAJAREMOS EN LA CUE

- EXERCISE 1: DESPLIEGUE A PRODUCCIÓN Y DEPLOY DE PROYECTO CREADO CON VUE/CLI.
- EXERCISE 2: DEPLOY DE UN APLICACIÓN SFC CON PARCEL.
- EXERCISE 3: DEPLOY DE UNA APLICACIÓN CON FIREBASE.
- EXERCISE 4: DEPLOY DE UNA APLICACIÓN PARA GITHUB PAGES.

EXERCISE 1: DESPLIEGUE A PRODUCCIÓN Y DEPLOY DE PROYECTO CREADO CON VUE/CLI.

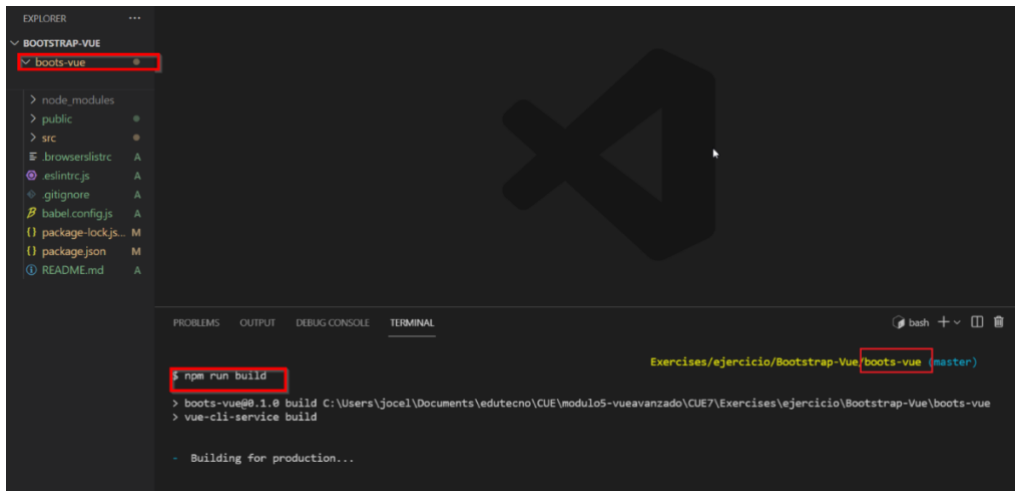
INTRODUCCIÓN

En este ejercicio aprenderemos a desplegar nuestra aplicación o sitio web en producción, para probarla tal como lo haría el usuario final, es decir, hacer un **build** de ella. Para esto, existen herramientas que nos ayudarán en el proceso.

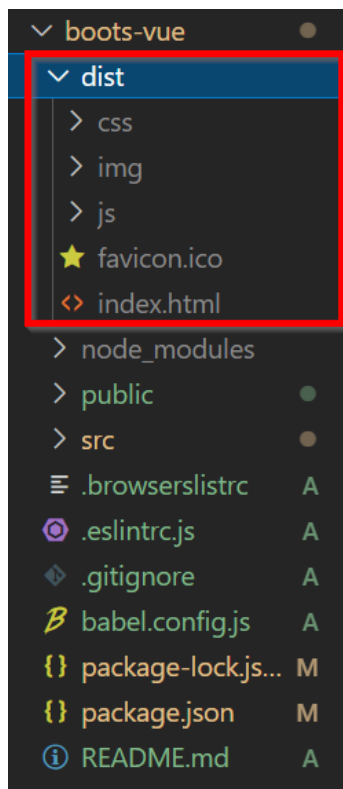
El despliegue en producción es importante, porque nos permite optimizar nuestra aplicación, esto quiere decir que, al momento de hacer el **build**, los archivos del proyecto se van a comprimir/empaquetar al máximo, por lo que la carga en el navegador será lo más rápida posible.

MODO PRODUCCIÓN

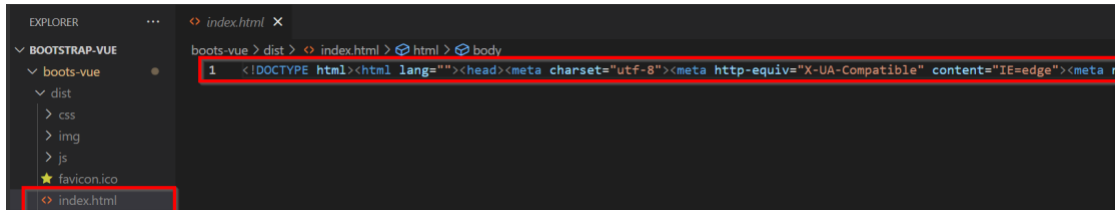
En primer lugar, escogeremos una app que ya hayamos desarrollado con **VUE/CLI**, o podemos crear una nueva, y a continuación realizar los pasos que haremos en este ejercicio. En este caso, se escogió el proyecto desarrollado con **bootstrapVue**. Lo abrimos en VSC, iniciamos la consola, corroboramos que nos encontramos dentro del proyecto correspondiente, escribimos el comando **"npm run build"**, y presionamos "enter".



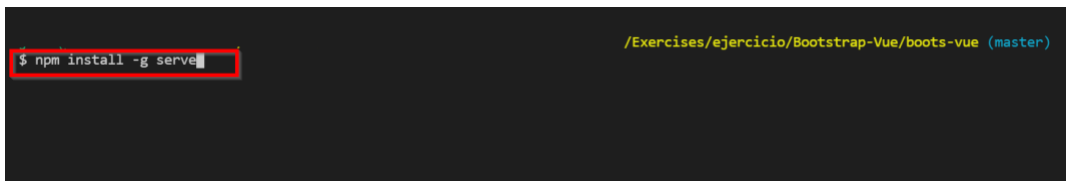
Vemos que se empieza a empaquetar nuestra aplicación. Una vez empaquetada, aparecerá una carpeta nueva llamada **“dist”**.



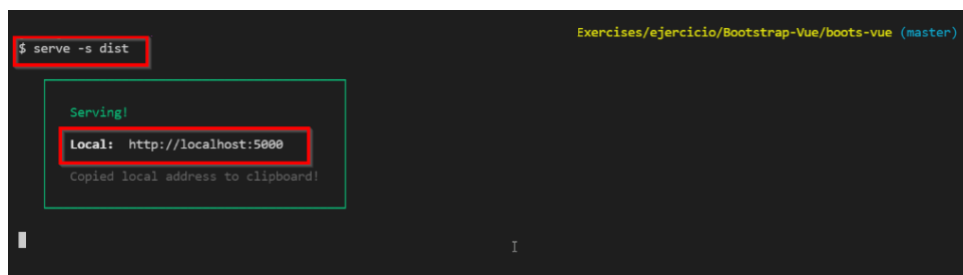
Dentro de la carpeta **"dist"**, se pueden ver otras carpetas que se crearon: **"css"**, **"img"**, **"js"**, y un archivo principal **"index.html"**. Si revisamos este último, notaremos que es un archivo de una sola línea, es decir, está comprimido.



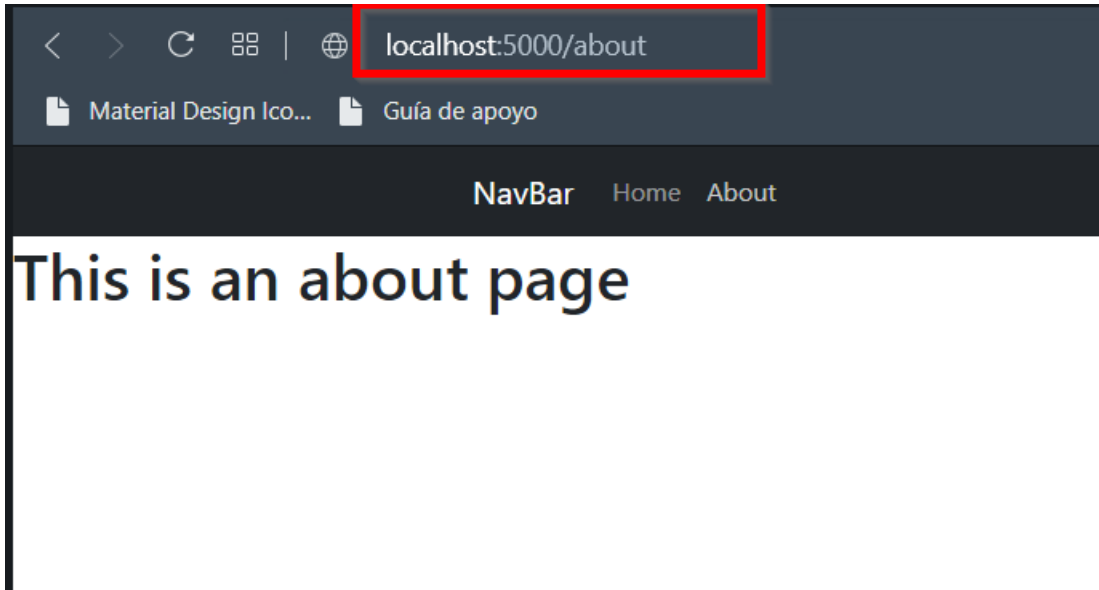
Lo siguiente que haremos, para probar que nuestra aplicación funciona, será instalar la librería **serve**. Ésta nos permite servir sitios estáticos, como si nuestra página estuviera en un servidor. Para ello, en la consola de VSC, escribiremos el comando: **"npm install -g (para instalarlo de forma global) serve"**, y presionaremos "enter".



Una vez que se haya instalado, comprobamos que nuevamente nos encontramos en la raíz de nuestro proyecto, ejecutamos el comando **"serve -s dist"**, y presionamos "enter". Lo que hará es levantar nuestra aplicación como si se tratase de un servidor en el **puerto 5000**.



Nos dirigimos al navegador, y podemos ver finalmente nuestra aplicación empaquetada sin errores.



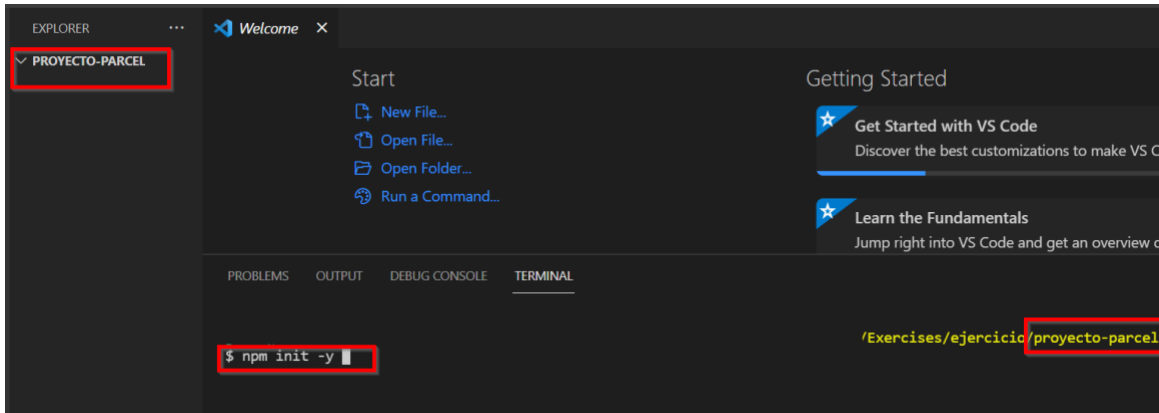
EXERCISE 2: DEPLOY DE UN APLICACIÓN SFC CON PARCEL

INTRODUCCIÓN

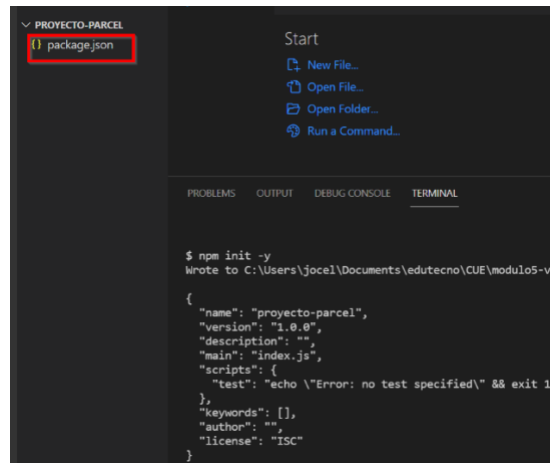
En este ejercicio aprenderemos a usar **Parcel**. Éste es un empaquetador de archivos para aplicaciones web, al igual que **Webpack**. Dispone de soporte para la compilación de archivos como: **css, javascript, html**, entre otros, y nos permitirá levantar un servidor web para probar nuestro proyecto.

INSTALANDO PARCEL

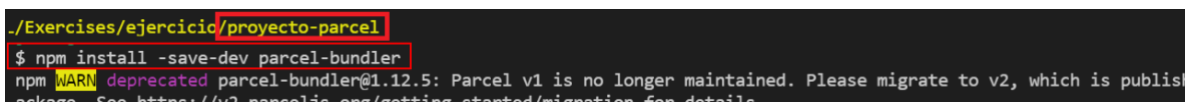
Para empaquetar con **Parcel**, lo primero que haremos es ir a VSC, nos ubicamos en la carpeta donde tenemos nuestro proyecto, en la terminal escribiremos el comando: **"npm init -y"** para iniciar, y le presionamos "enter".



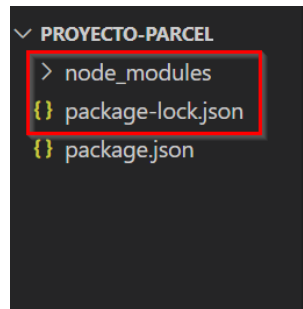
Éste nos va a generar un archivo **"package.json"**, con las configuraciones necesarias.



Ahora, procederemos a instalar **Parcel** como una dependencia de desarrollo del proyecto. Para esto, ejecutaremos en la terminal el comando: **"npm install -save-dev parcel-bundler"**, y presionamos "enter".



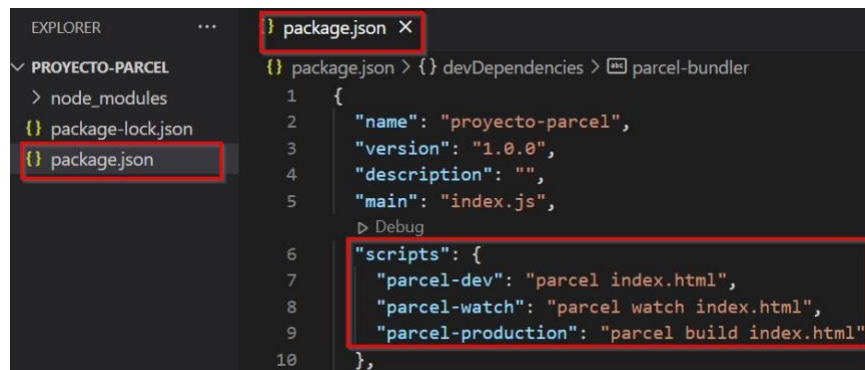
Ya con esto, hemos instalamos el núcleo de **Parcel** como una dependencia de desarrollo. La carpeta creada fue **"node_modules"**, y el archivo **"package-lock.json"**.



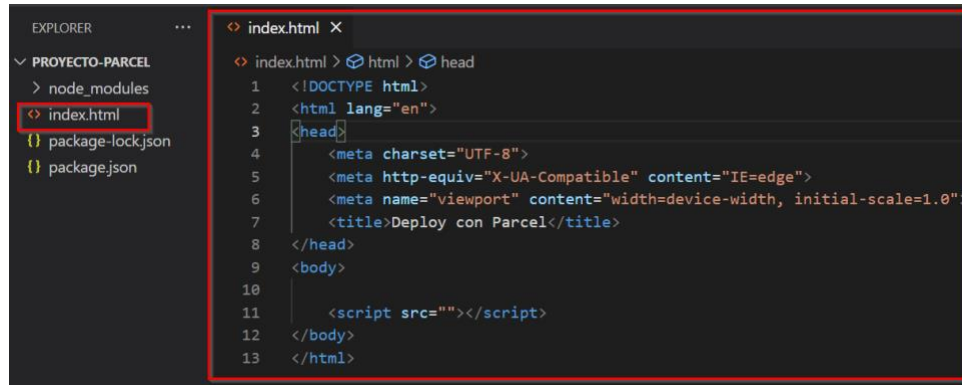
Una vez instalada la dependencia, agregaremos el script correspondiente para ejecutar **Parcel** en el archivo **"package.json"**. En "scripts", borramos la línea 7, que dice "test", y en su lugar agregamos:

```
1 "parcel-dev": "parcel index.html",  
2 "parcel-watch": "parcel watch index.html",  
3 "parcel-production": "parcel build index.html"
```

Debería quedar de la siguiente forma:



Ordenamos, guardamos, y ahora crearemos un archivo `index.html`, donde colocaremos la estructura base de este tipo de archivos, y agregaremos las etiquetas `script`. Cambiaremos el `title`, por: "Deploy con Parcel".



```
EXPLORER
PROYECTO-PARCEL
  node_modules
  index.html
  package-lock.json
  package.json

index.html X
index.html > html > head
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Deploy con Parcel</title>
8 </head>
9 <body>
10
11   <script src=""></script>
12 </body>
13 </html>
```

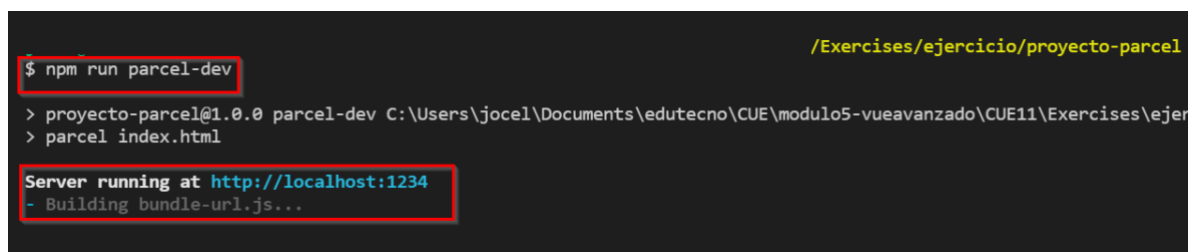
EMPAQUETANDO CON PARCEL

Finalmente, estando en la terminal, y siempre cerciorándonos de que nos encontramos ubicados en la raíz del proyecto, podremos ejecutar `Parcel` en modo desarrollo, a través de: `npm run parcel-dev`.



```
/Exercises/ejercicio/proyecto-parcel
$ npm run parcel-dev
```

Éste empaquetará el proyecto, y lo disponibilizará a través de un servidor web, al cual se puede acceder a través de la dirección: `localhost:1234`.

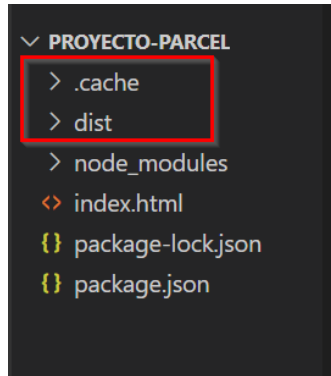


```
/Exercises/ejercicio/proyecto-parcel
$ npm run parcel-dev
> proyecto-parcel@1.0.0 parcel-dev C:\Users\jocel\Documents\edutecno\CUE\modulo5-vueavanzado\CUE11\Exercises\ejercicio
> parcel index.html

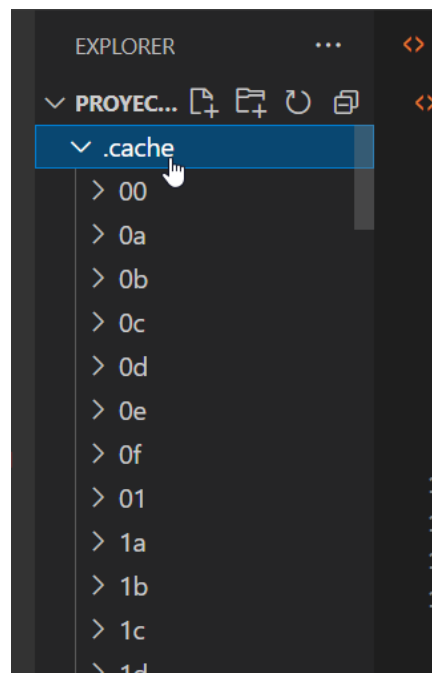
Server running at http://localhost:1234
- Building bundle-url.js...
```



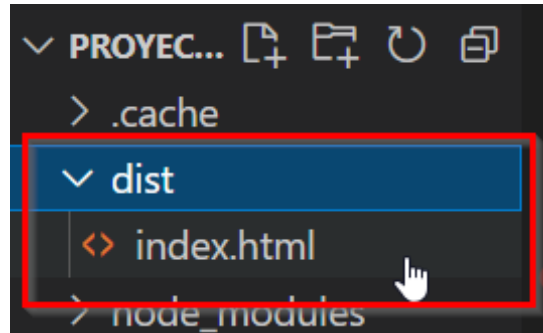
Las carpetas que se crearon son: **“cache”** y **“dist”**.



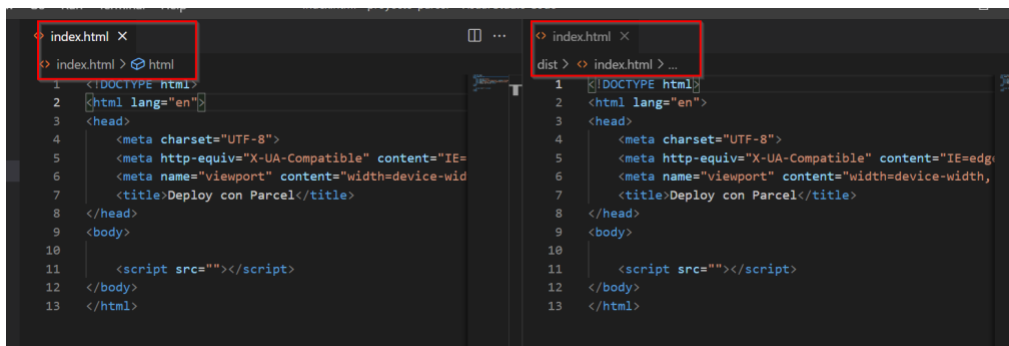
“Cache” es una carpeta que utilizada por **Parcel**, para darse cuenta de qué cosas tienen que recargar, y qué no, y así hacer el proceso lo más rápido posible. Se debe omitir al momento de emplear un sistema de control de versiones como **Git**.



“Dist” contiene un “index.html”.



En este momento, se verá igual al “index” que tenemos, pues no hemos hecho modificaciones grandes, pero si hubiéramos colocado más cosas, Parcel lo procesaría y convertiría, haciendo algunos cambios en el “index” de la carpeta “dist”.

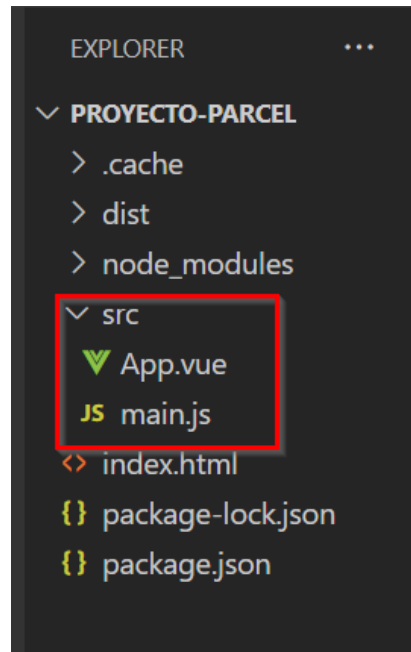


Lo siguiente que haremos será agregar **Vue** a nuestro proyecto, para esto escribiremos el comando: “npm add vue”, y presionamos “enter”.





A continuación, crearemos la carpeta `src`, con dos archivos dentro, que son: `main.js` y `App.vue`.



En `main.js`, agregamos el siguiente código:

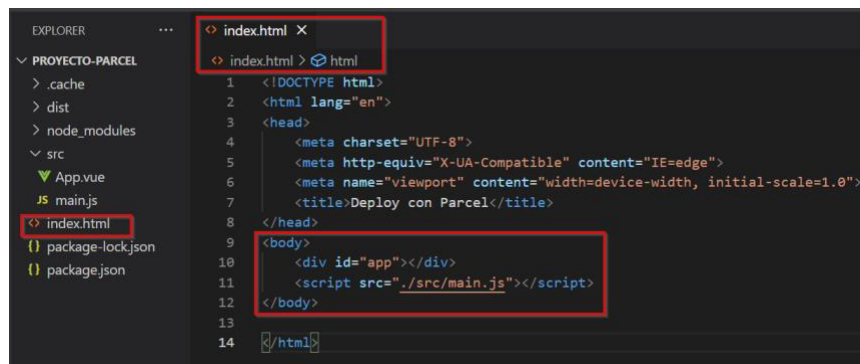
```
1 import Vue from 'vue';
2 import App from './App.vue';
3
4 new Vue({
5   el: "#app",
6   render: h => h(App),
7 })
```

Y guardamos.

En `App.vue`, agregamos:

```
1 <template>
2   <div id="app">
3     {{message}}
4   </div>
5 </template>
6
7 <script>
8   export default {
9     data() {
10      return {
11        message: 'Hola mundo desde un SFC en vue';
12      }
13    }
14  </script>
15  <style scoped>
16    #app {
17      font-size: 50px;
18      font-family: "Calibri", sans-serif;
19      color: rgb(146, 201, 46);
20      background: rgb(25, 46, 167);
21    }
22  </style>
```

Y guardamos. Por último, llamaremos al archivo `main.js` en el `index.html`, entre las etiquetas `script`, y crearemos un `<div id="app"></div>`.



Guardamos todo, y levantamos nuestro proyecto con: `"npm run parcel-dev"`.

```
$ npm run parcel-dev

> proyecto-parcel@1.0.0 parcel-dev C:\Users\jocel
> parcel index.html

Server running at http://localhost:1234
🌟 Built in 179ms.
```

Nos dirigimos al navegador, y finalmente podemos ver nuestro proyecto de **Single File Component** empaquetado con **Parcel**.



EXERCISE 3: DEPLOY DE UNA APLICACIÓN CON FIREBASE

INTRODUCCIÓN

En este ejercicio, aprenderemos a desplegar nuestra aplicación o sitio web en **Firebase**. Como vimos anteriormente, éste dispone de herramientas que nos ayudan a que los proyectos sean más fáciles de construir. Una de ellas, es el servicio de hosting, que permite que nuestro proyecto quede alojado en un servidor de Google, y nos da la posibilidad de que otros usuarios puedan utilizar la aplicación.

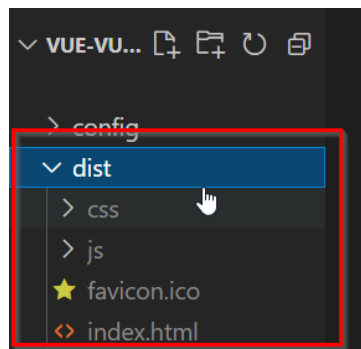
PASO A PRODUCCIÓN

Con todo lo que hemos estructurado hasta este momento, hacer un **deploy** de nuestra aplicación será sencillo. En primer lugar, escogeremos el proyecto que se subirá, en este caso, será el de Postres. Lo abrimos con VSC, y en la consola escribiremos el comando `"npm run build"` para pasarlo a producción.



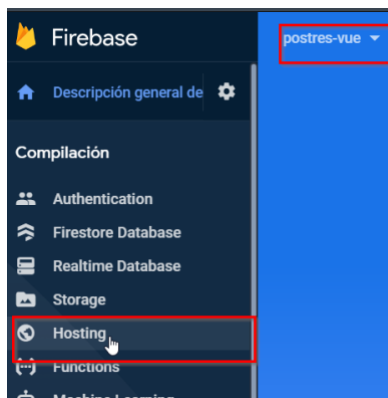
```
$ npm run build
> vue-vuetify@0.1.0 build
> vue-cli-service build
```

Para confirmar que todo salió correctamente, revisamos que se haya creado la carpeta **"dist"**, donde estará nuestro proyecto compilado.



HOSTING CON FIREBASE

Lo segundo que haremos, será dirigirnos a **Firebase**, y abrimos el proyecto que creamos: **"postres-vue"**. Luego, vamos a "hosting".





Y seleccionamos comenzar.



A continuación, vamos a instalar las herramientas de **Firebase**, para eso copiamos el comando que nos aparece, y lo insertamos en la consola de VSC, presionamos "enter".

```
$ npm install -g firebase-tools
```

Una vez instalado, volvemos a la página de **Firebase**, y presionamos siguiente.



× Configurar Firebase Hosting

1 Instala Firebase CLI

Necesitas Firebase CLI (una herramienta de línea de comandos) para alojar tu sitio con Firebase Hosting.

Ejecuta el siguiente comando de [npm](#) para instalar la CLI o actualizar a su versión más reciente.

```
$ npm install -g firebase-tools
```

¿No funciona? Consulta la [referencia de Firebase CLI](#) o cambia tus [permisos de npm](#)

☐ Mostrarme también los pasos para agregar el SDK de Firebase JavaScript a mi app web
El SDK incluye Cloud Firestore, Authentication, Performance Monitoring y mucho más.
Puedes agregarlo ahora o más adelante.

Siguiente

2 Inicializa el proyecto

3 Implementa en Firebase Hosting

Nos aparecerá la opción para autenticarnos.

2 Inicializa el proyecto

Abre una ventana de la terminal y navega a un directorio raíz para tu app web (si no tienes deberás crearlo).

Acceder a Google

```
$ firebase login
```

Inicia el proyecto

Ejecuta el siguiente comando en el directorio raíz de tu app:

```
$ firebase init
```

Siguiente



Copiamos y pegamos en la terminal `“firebase login”`, presionamos “enter”.

```
Exercises/ejercicio/VueVuetify/vue-vuetify (main)
$ firebase login
```

En la siguiente opción, dejaremos la que viene por defecto: `“Yes”`.

```
$ firebase login
i Firebase optionally collects CLI usage and error reporting information to help improve our products. Data collection policy (https://policies.google.com/privacy) and is not used to identify you.

? Allow Firebase to collect CLI usage and error reporting information? Yes
i To change your data collection preference at any time, run `firebase logout` and log in again.
```

Se abrirá una nueva ventana, ésta es para darle los permisos necesarios a **Firestore** con la cuenta con la que creamos nuestra aplicación, y presionaremos permitir.



Esto permitirá a **Firebase CLI** hacer lo siguiente:

- Ver, editar, configurar y eliminar tus datos de Google Cloud Platform ⓘ
- 🔥 Ver y administrar la configuración y los datos de Firebase ⓘ
- Ver tus proyectos de Cloud Platform ⓘ

Confirma que confías en Firebase CLI

Puede que estés compartiendo información sensible con este sitio o esta aplicación. Puedes ver o retirar el acceso en cualquier momento en tu [cuenta de Google](#).

Descubre cómo te ayuda Google a [compartir datos de forma segura](#).

Consulta la [Política de Privacidad](#) y los [Términos del Servicio](#) de Firebase CLI.

[Cancelar](#)

Permitir

Nos aparecerá una ventana de confirmación.

Woohoo!

Firebase CLI Login Successful

You are logged in to the Firebase Command-Line interface. You can immediately close this window and continue using the CLI.

Con el **login** finalizado, seguiremos con el comando **"firebase init"**, lo copiamos, pegamos y presionamos "enter".

```
Edutecno@DESKTOP-LQFETQU MINGW64 ~/Documents/Front-end/MSCUE/CUE7/vue-vuetify (master)
$ firebase init

#####
##
#####
##
##

You're about to initialize a Firebase project in this directory:
```

Lo siguiente que nos aparecerá será un menú en el que debemos escoger. Primero, presionamos "enter", y ahora debemos seleccionar el hosting mediante las flechas arriba/abajo. Nos movemos en el menú, y con la tecla de espacio elegimos la alternativa que queremos, y nuevamente "enter".

```
You're about to initialize a Firebase project in this directory:

C:\Users\Edutecno\Documents\Front-end\MSCUE\CUE7\vue-vuetify

? Are you ready to proceed: Yes
? Which Firebase features do you want to set up for this directory? Press Space to select features, the
ct, <a> to toggle all, <i> to invert selection)
( ) Realtime Database: Configure a security rules file for Realtime Database and (optionally) provisio
( ) Firestore: Configure security rules and indexes files for Firestore
( ) Functions: Configure a Cloud Functions directory and its files
>( ) Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys
( ) Hosting: Set up GitHub Action deploys
( ) Storage: Configure a security rules file for Cloud Storage
( ) Emulators: Set up local emulators for Firebase products
(Move up and down to reveal more choices)
```

Ahora, nos dará a elegir el proyecto donde queremos alojar nuestra aplicación. En este caso, buscaremos la aplicación con nombre **"postres-vue"**, y presionamos "enter".



```
Project Setup

First, let's associate this project directory with a Firebase project.
You can create multiple project aliases by running firebase use --add,
but for now we'll just set up a default project.

? Please select an option: Use an existing project
? Select a default Firebase project for this directory:
  nasa-api-1a27c (Nasa-api)
  nueva-api-nasa (nueva api nasa)
  otto-klaus-e57ec (Otto-Klaus)
> postres-vue-8040f (postres-vue)
  practicaensayo (practicaEnsayo)
  pree-72485 (pree)
  prueba-ensayo-2b352 (prueba-ensayo)
(Move up and down to reveal more choices)
```

También debemos especificar cuál será nuestro directorio público. Éste es un paso importante, ya que debemos colocar la carpeta **"dist"**, que fue creada al momento de pasar nuestro proyecto a producción, y presionamos "enter".

```
=== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? dist
```

En la siguiente opción colocaremos **"Yes"**, y por último, pondremos la opción **"No"**, presionamos "enter y pondremos **"No"**.

```
? Configure as a single-page app (rewrite all urls to /index.html)? Yes
? Set up automatic builds and deploys with GitHub? No
? File dist/index.html already exists. Overwrite? No
i Skipping write of dist/index.html
```



Ahora, se crearán otros archivos relacionados con **Firebase**. Volvemos a compilar con el comando: **"npm run build"**.

```
Edutecno@DESKTOP-LQFETQU MINGW64 ~/Documents/Front-end/M5CUE/CUE7/vue-vuetify (master)
$ npm run build
> vue-vuetify@0.1.0 build
> vue-cli-service build
- Building for production...
```

Una vez que todo se haya compilado, lo que debemos hacer es el **deploy**. Para ello, escribiremos el comando **"firebase deploy"**, presionamos "enter", y esperamos a que termine.

```
Edutecno@DESKTOP-LQFETQU MINGW64 ~/Documents/Front-end/M5CUE/CUE7/vue-vuetify (master)
$ firebase deploy
=== Deploying to 'postres-vue-8040f' ...

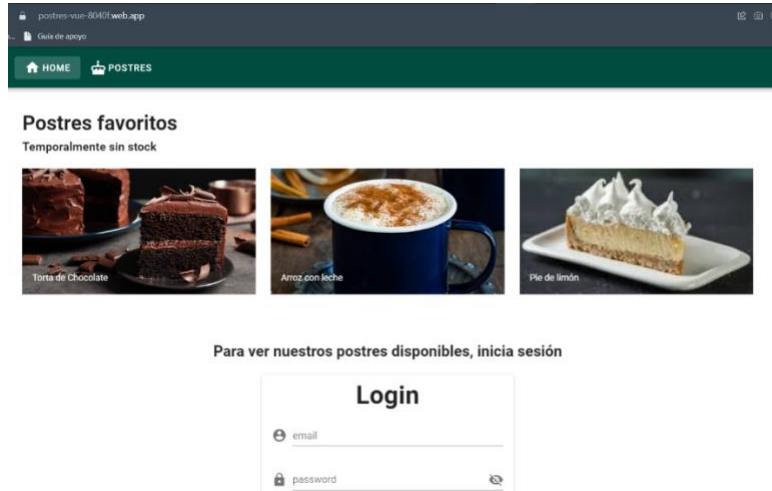
i deploying hosting
```

Para cerciorarnos de que el **deploy** se realizó correctamente, debemos ingresar a la siguiente **URL**, la copiamos, vamos al navegador, la pegamos y presionamos "enter".

```
+ Deploy complete!

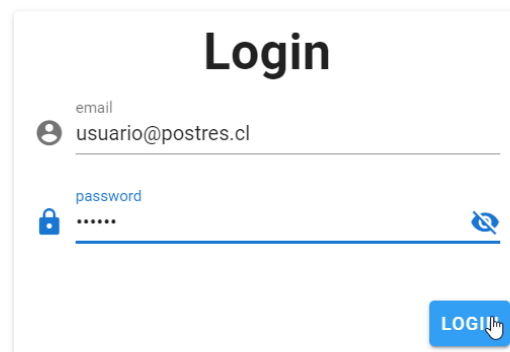
Project Console: https://console.firebase.google.com/project/postres-vue-8040f/overview
Hosting URL: https://postres-vue-8040f.web.app
```

Finalmente, podemos ver nuestra aplicación 100% funcional.



Ingresamos con el usuario que creamos anteriormente, junto con la password.

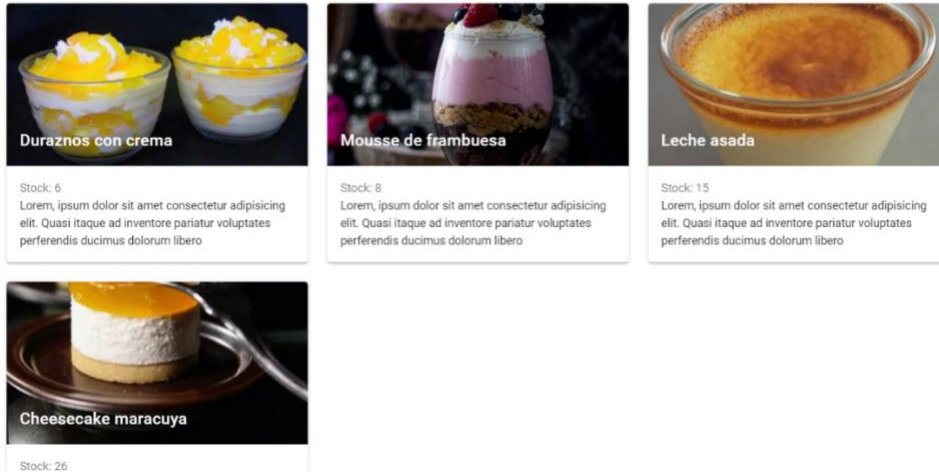
Para ver nuestros postres disponibles, inicia sesión



Y veremos que también funciona correctamente.

Bienvenido usuario@postres.cl

Listado de postres con stock



De esta manera, es como nosotros podemos hacer **deploy** de nuestra aplicación con **VUEJS** y **Firebase**.

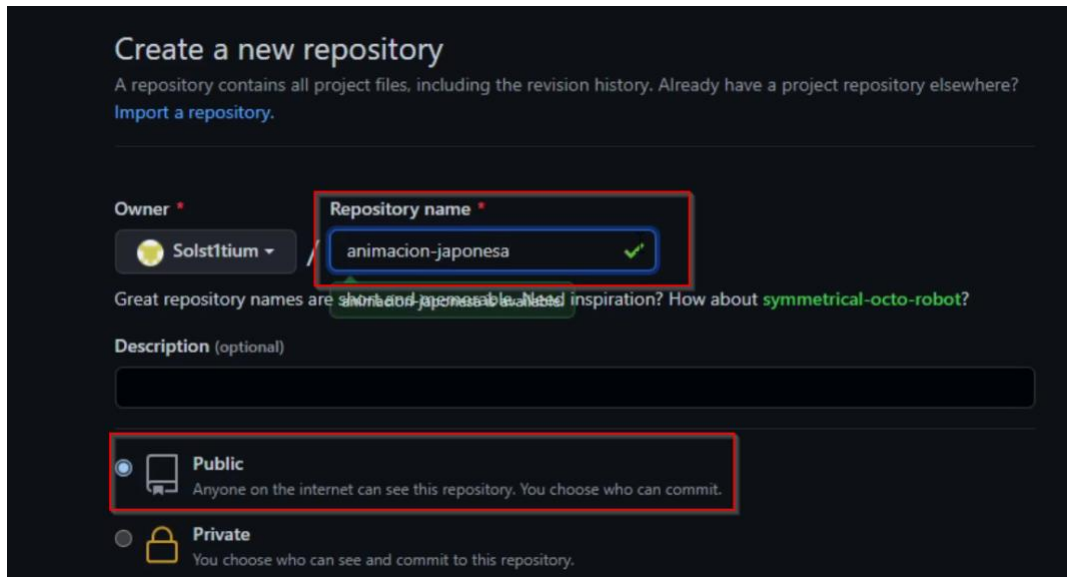
EXERCISE 4: DEPLOY DE UNA APLICACIÓN PARA GITHUB PAGES

INTRODUCCIÓN

En este ejercicio aprenderemos cómo publicar una aplicación creada con **Vue.JS**, en **GitHub Pages**. Esto nos permitirá ahorrarnos una cuenta de hosting y la configuración de un servidor, siendo un lugar ideal para subir documentaciones o aplicaciones de demostración. Antes de comenzar, recordemos que debemos tener instalado **Git** en nuestro computador.

CREANDO EL REPOSITORIO

Lo primero que haremos será escoger algún proyecto que hayamos creado. Una vez esté listo, iremos a nuestra cuenta de **Github**, iniciamos sesión, y crearemos un repositorio vacío para dicho proyecto, que no incluya ningún archivo. En este caso, le pondremos de nombre “animación-japonesa”, y será público.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *
Solstium

Repository name *
animacion-japonesa

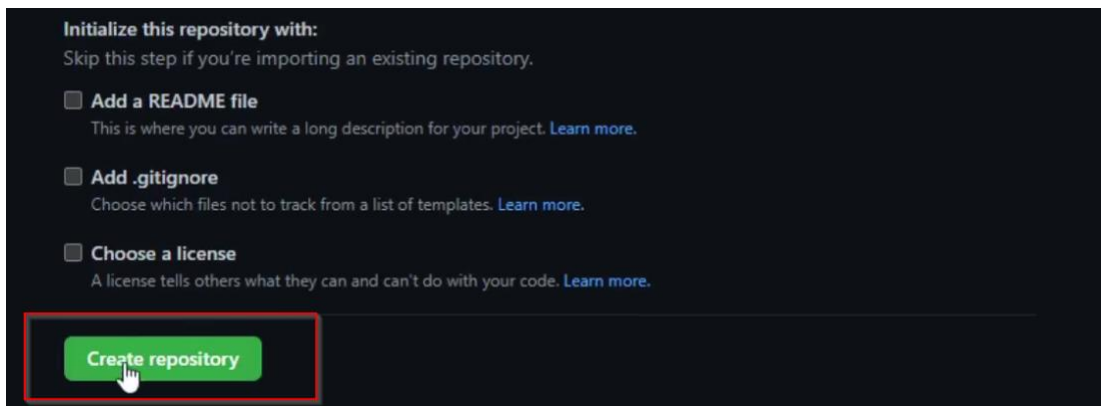
Great repository names are short and memorable. Need inspiration? How about [symmetrical-octo-robot?](#)

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Con esto definido, le damos clic al botón **“Create repository”**.



Initialize this repository with:
Skip this step if you're importing an existing repository.

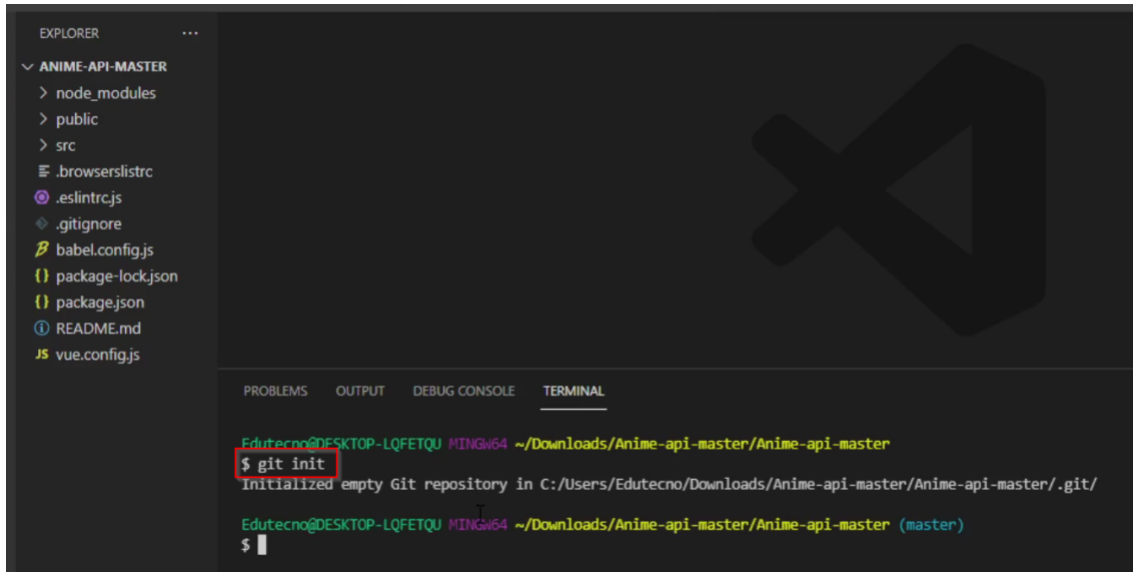
☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

Una vez creado el repositorio, iremos a VSC con nuestro proyecto abierto en el editor, desplegamos la consola, y escribiremos: **“git init”**, para así confirmar que se haya inicializado **Git**.

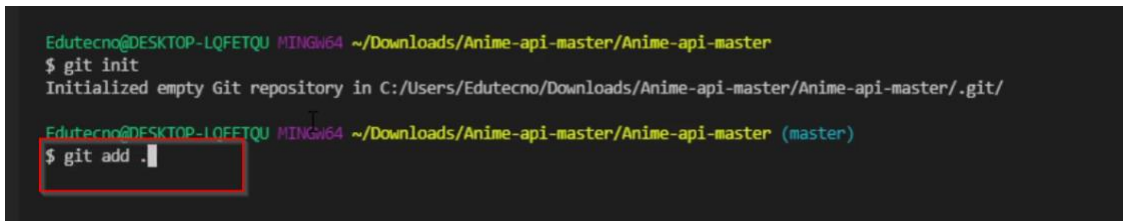


```
EXPLORER
  ANIME-API-MASTER
    node_modules
    public
    src
    .browserslistrc
    .eslintrc.js
    .gitignore
    babel.config.js
    package-lock.json
    package.json
    README.md
    vue.config.js

EduTecno@DESKTOP-LQFETQU MINGW64 ~/Downloads/Anime-api-master/Anime-api-master
$ git init
Initialized empty Git repository in C:/Users/EduTecno/Downloads/Anime-api-master/Anime-api-master/.git/

EduTecno@DESKTOP-LQFETQU MINGW64 ~/Downloads/Anime-api-master/Anime-api-master (master)
$
```

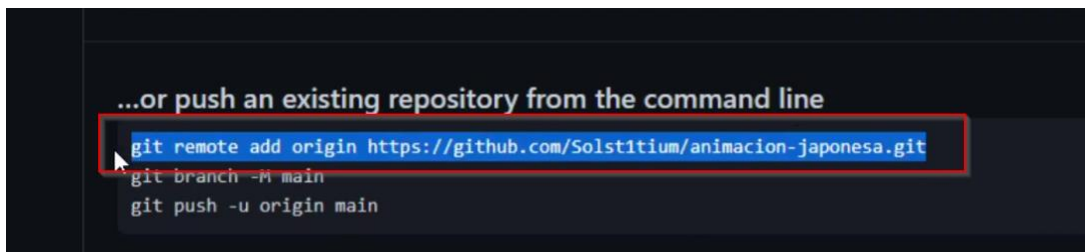
A continuación, escribiremos el comando: `"git add ."`



```
EduTecno@DESKTOP-LQFETQU MINGW64 ~/Downloads/Anime-api-master/Anime-api-master
$ git init
Initialized empty Git repository in C:/Users/EduTecno/Downloads/Anime-api-master/Anime-api-master/.git/

EduTecno@DESKTOP-LQFETQU MINGW64 ~/Downloads/Anime-api-master/Anime-api-master (master)
$ git add .
```

Ahora, agregaremos el repositorio que hemos creado en **GitHub** como origen, y copiaremos el comando: `"git remote add origin urlRepositorio"`.



```
...or push an existing repository from the command line

git remote add origin https://github.com/Solstitium/animacion-japonesa.git
git branch -M main
git push -u origin main
```


Lo pegamos en nuestra terminal, y presionamos “enter”. Luego, haremos un **commit** de los archivos de nuestro proyecto, escribiendo: **“git commit -m ‘mi primer commit’”**.

```
EduTecno@DESKTOP-LOFETQU MINGW64 ~/Downloads/Anime-api-master/Anime-api-master (master)
$ git remote add origin https://github.com/Solstitium/animacion-japonesa.git

EduTecno@DESKTOP-LOFETQU MINGW64 ~/Downloads/Anime-api-master/Anime-api-master (master)
$ git commit -m "mi primer commit"
```

Después, copiamos: **“git Branch -M main”**.

```
...or push an existing repository from the command line

git remote add origin https://github.com/Solstitium/animacion-japonesa.git
git branch -M main
git push -u origin main
```

Lo pegamos en nuestra terminal, y presionamos “enter”.

```
EduTecno@DESKTOP-LOFETQU MINGW64 ~/Downloads/Anime-api-master/Anime-api-master (master)
$ git branch -M main
```

Y, por último, haremos un **push** desde el repositorio local, hasta la rama **main** del repositorio de GitHub, escribiendo: **“git push -u origin main”**.

```
create mode 100644 vue.config.js

Edutecno@DESKTOP-LQFETQU MINGW64 ~/Downloads/Anime-api-master/Anime-api-master (master)
$ git branch -M main

Edutecno@DESKTOP-LQFETQU MINGW64 ~/Downloads/Anime-api-master/Anime-api-master (main)
$ git push -u origin main
Enumerating objects: 34, done.
Counting objects: 100% (34/34), done.
Delta compression using up to 4 threads
Compressing objects: 100% (30/30), done.
Writing objects: 100% (34/34), 673.40 KiB | 10.36 MiB/s, done.
Total 34 (delta 1), reused 0 (delta 0), pack-reused 0
```

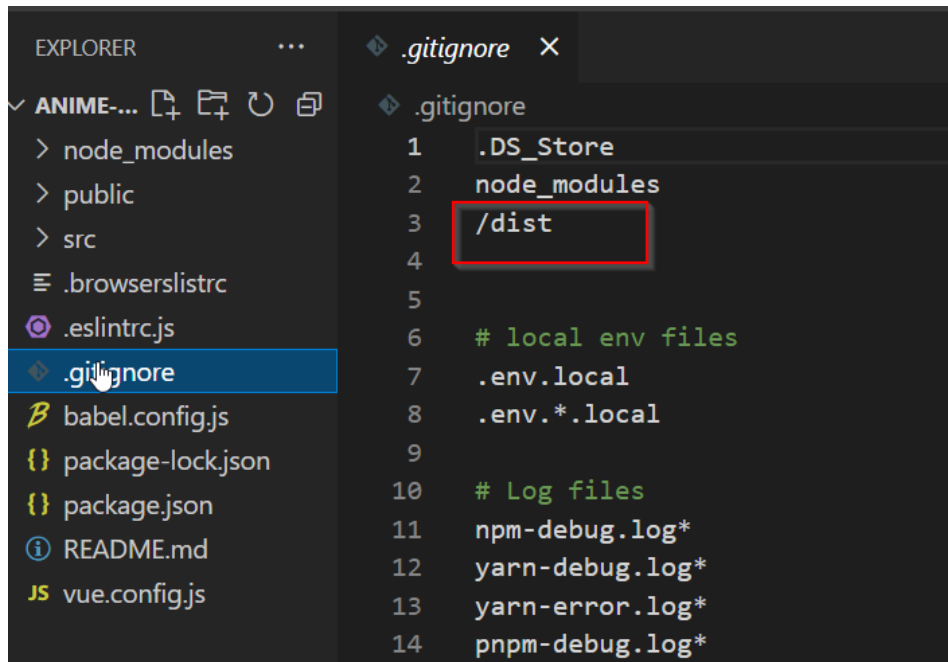
Con todo esto, tendremos el código del proyecto en **GitHub**.

DEPLOY EN GITHUB PAGES

Ahora, haremos el **deploy** en **github pages**. En primer lugar, debemos recordar que cuando compilamos una aplicación **Vue**, la carpeta por defecto es **/dist**, por lo que tendremos que enviarla al repositorio. Para eso, necesitamos crear una nueva rama, en la que agregaremos ciertos cambios en la configuración de **Vue**; la llamaremos **gh-pages**, escribiendo en la terminal: **"git checkout -b gh-pages"**.

```
Edutecno@DESKTOP-LQFETQU MINGW64 ~/Downloads/Anime-api-master/Anime-api-master (main)
$ git checkout -b gh-pages
Switched to a new branch 'gh-pages'
```

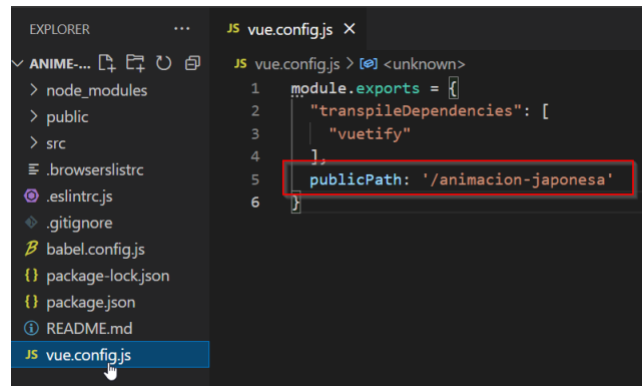
Seguidamente, ya en la nueva rama, editaremos el archivo **".gitignore"**, y eliminaremos la línea donde llama a la carpeta **/dist**, de modo que sea posible enviar este directorio a **GitHub**, y guardamos.



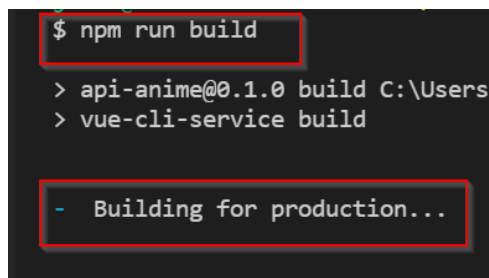
Lo siguiente que haremos será crear el archivo **vue.config.js**, en la carpeta raíz de nuestro proyecto, de modo que podamos establecer las opciones de compilación de **Vue**. Primero, debemos colocar el directorio público en el que estarán los archivos de nuestro proyecto. Una vez esté el proyecto en **GitHub Pages**, la carpeta tendrá el mismo nombre que el repositorio asociado, por lo que debemos agregar lo siguiente:

```
1 module.exports = {
2   publicPath: '/nombre-repositorio'
3 }
```

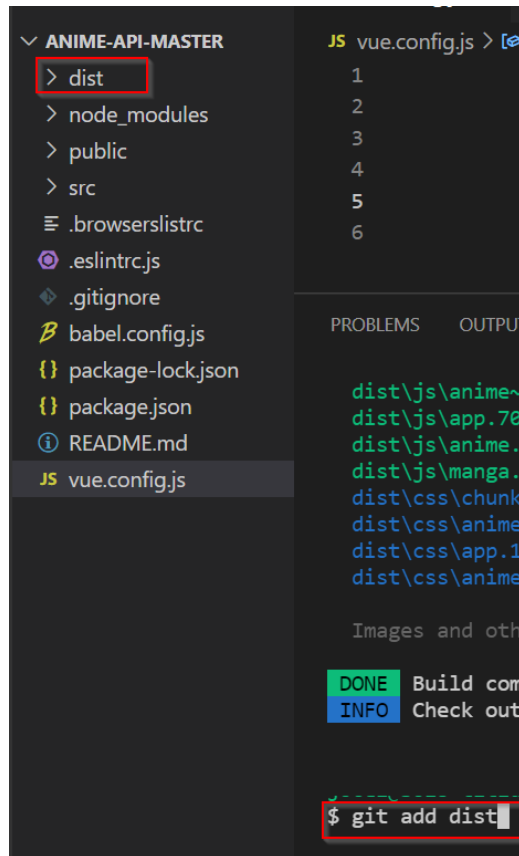
Guardamos todos los cambios.



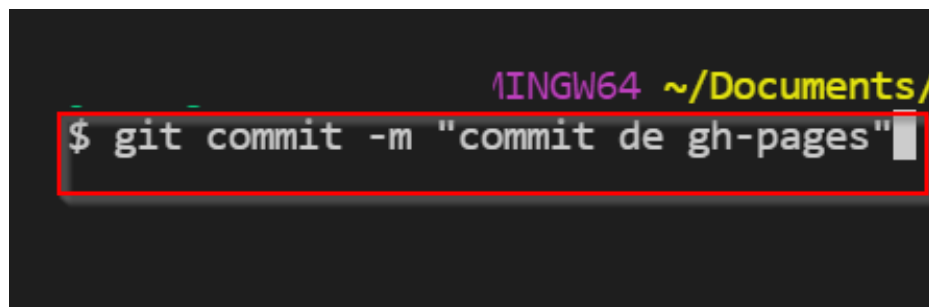
Ahora ya podemos hacer un **build** del proyecto para generar la carpeta **/dist**. Para ello, ejecutaremos el comando: **“npm run build”**.



Lo siguiente será agregar la carpeta **/dist** al repositorio, y ejecutaremos el comando: **“git add dist”**.



Haremos un **commit** con: `git commit -m "commit de gh-pages"`.



Y finalmente, haremos un **push** de la rama **gh-pages** a **GitHub**, agregaremos el prefijo **dist**, y escribiremos el comando: `git subtree push -prefix dist origin gh-pages`. Aquí indica que antes de **prefix** es doble guion, lo colocamos y ejecutamos.

```
$ git subtree push --prefix dist origin gh-pages
```

Esperamos que se suba nuestra página, y con esto hemos terminado. **Github** publicará la página automáticamente, y podremos acceder a ella en la **URL**: <https://usuario.github.io/nombre-repositorio> (<https://solst1tium.github.io/animacion-japonesa/>)

