

**Proyecto HelloZUM**

**Servicio REST realizado con NodeJS, Express JS**

**y Docker.**

**Realizado por Antony Dixon Albites Tapia**

**Contenido**

[**1-Tecnologías Por Utilizar:** 3](#_Toc5049820)

[**2-Creando el Servicio Rest:** 3](#_Toc5049821)

[2.1 Creando un repositorio local 4](#_Toc5049822)

[2.2 Iniciando con NodeJS y ExpressJS : 4](#_Toc5049823)

[**3- “Dockerizando” Nuestro servicio.** 7](#_Toc5049824)

[**4-Subiendo nuestro proyecto a GITHUB** 10](#_Toc5049825)

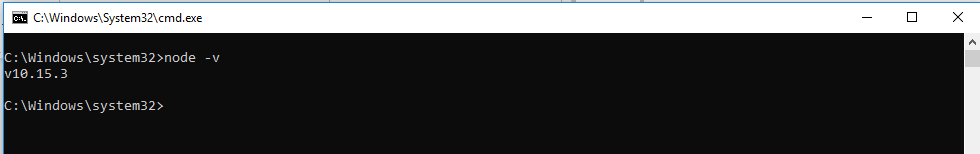
# **1-Tecnologías Por Utilizar:**

* **NodeJS y ExpressJS:** Para el desarrollo del servicio rest donde solo expondremos una ruta “/rest” para retornar un Json: {helloZum: 'Bienvenido a HelloZUM'}.
* **Docker:** Para poder encapsular todo en un contenedor, primero crearemos una Imagen con un Dockerfile y después la ejecutaremos con docker-compose.
* **GitHutb:** Una vez terminado con ayuda de git subiremos lo realizado a un repositorio en github.

# **2-Creando el Servicio Rest:**

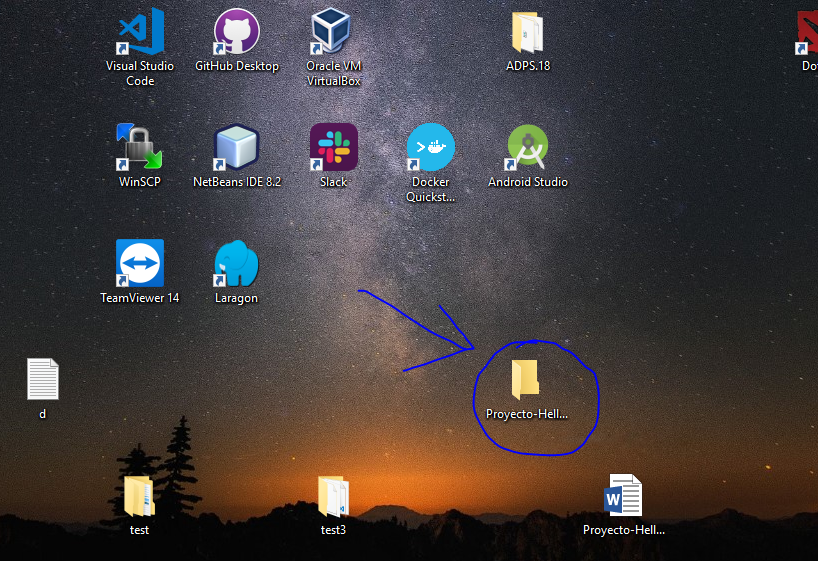
En esta sección crearemos el servicio con ayuda de NodeJS y ExpressJS, muy importante tener instalado NODE.JS antes de realizar los siguientes pasos, puedes instalar node.js de este enlace: <https://nodejs.org/es/>.

Descargar la versión recomendada y verificar si todo se instalo correctamente, abrir un CMD y escribir lo siguiente:



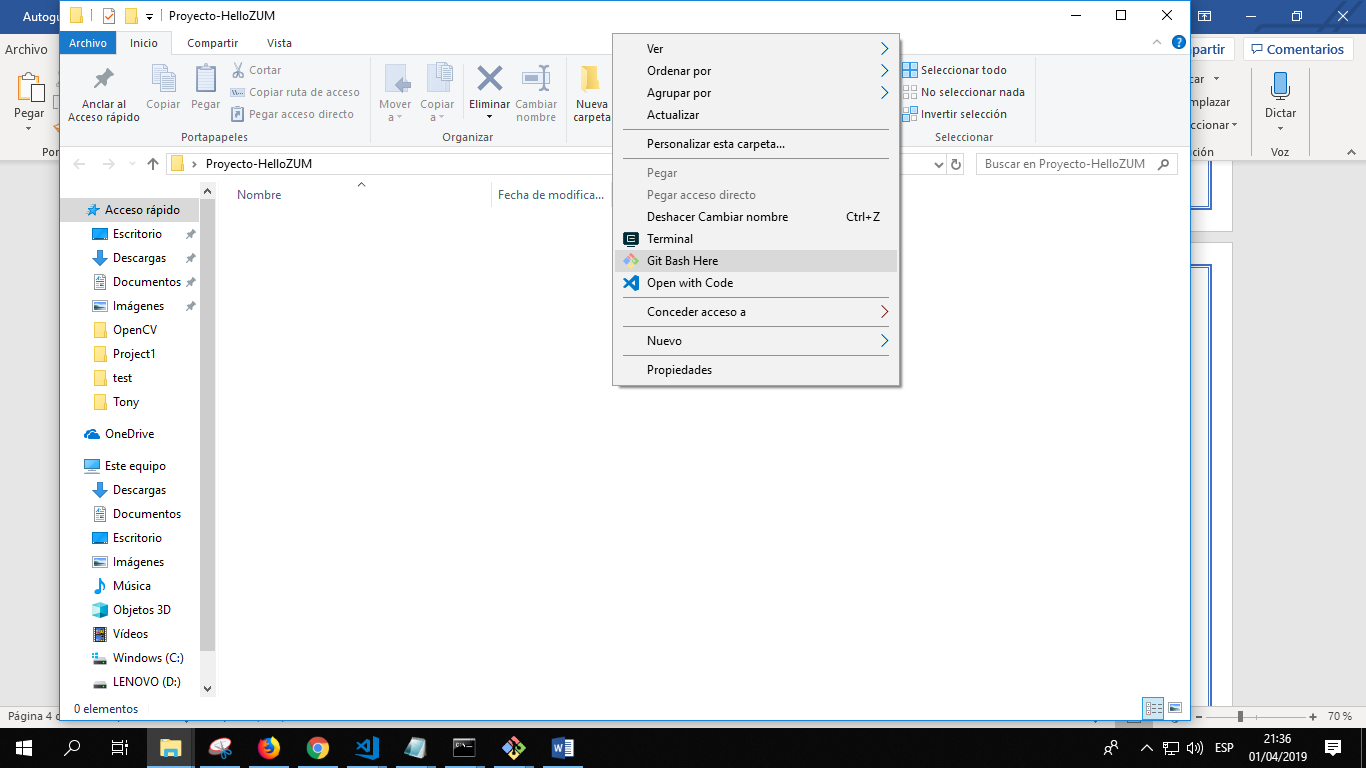
Si se muestra como la imagen, todo está correcto.

Crearemos una carpeta en el directorio de preferencia, yo lo haré en el escritorio.

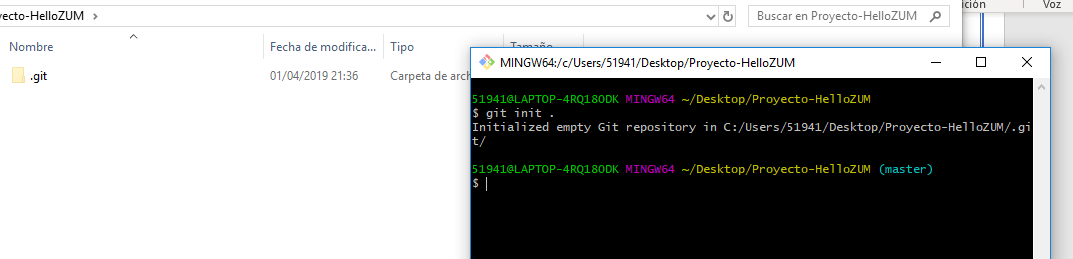


## 2.1 Creando un repositorio local

Dentro antes de iniciar con NODE.JS, crearemos un repositorio local con GIT, recuerda tenerlo instalado previamente, puedes instalarlo en el siguiente enlace: <https://git-scm.com/>. Instalar y después realizar lo siguiente dentro de la carpeta creada.



Iniciaremos un bash y dentro escribiremos lo siguiente.



Iniciaremos un repositorio local, por ahora es todo a realizar, puedes dejar el bash abierto, después con ayuda de él podremos subirlo a github.

## 2.2 Iniciando con NodeJS y ExpressJS :

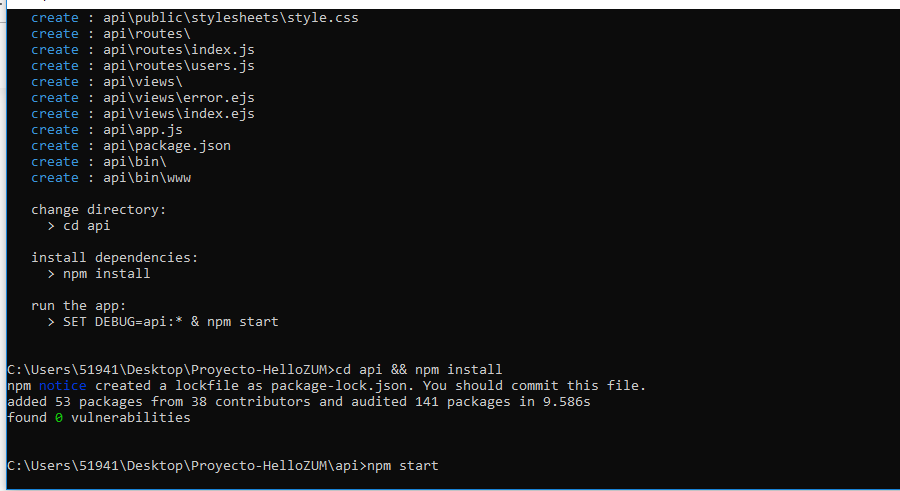
Abrimos un CMD en la carpeta actual y escribiremos los siguientes comandos en este orden:

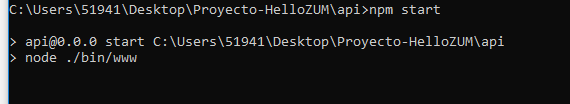
npm install express-generator -g

express --view=ejs api

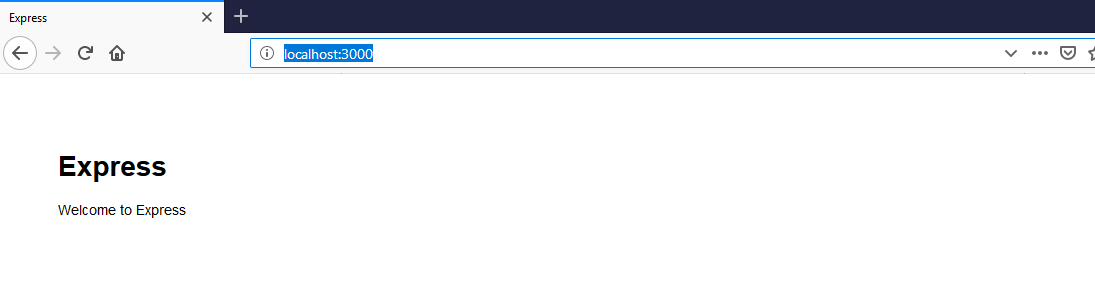
cd api && npm install

npm start

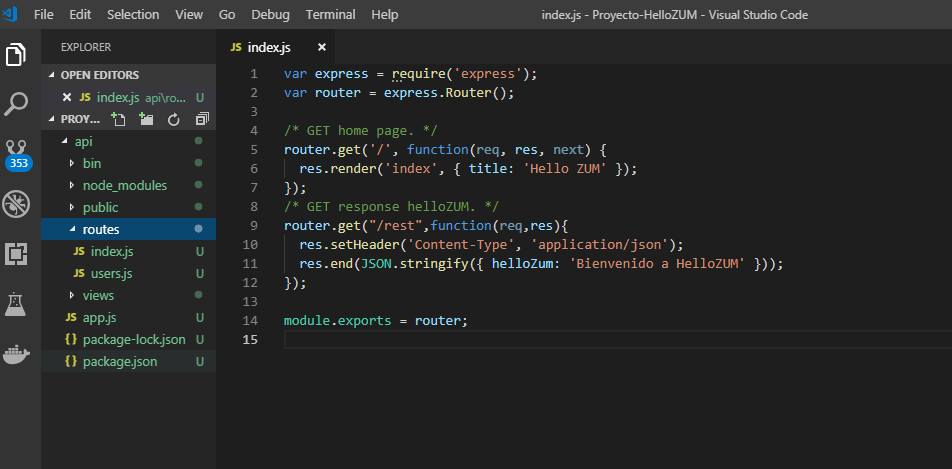




Ya se habrá creado nuestra aplicación Express, podemos acceder a ella mediante esta dirección : http://localhost:3000/

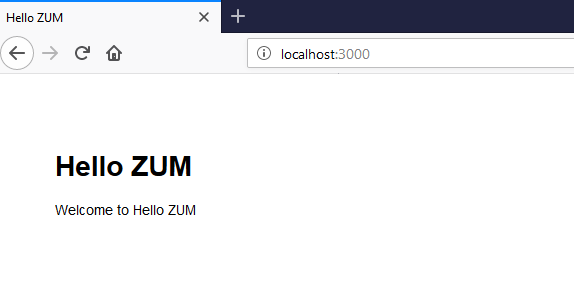


Ahora entraremos a la carpeta “API” y modificaremos unos archivos, puedes usar el editor de código de preferencia, en mi caso será Visual Studio Code.

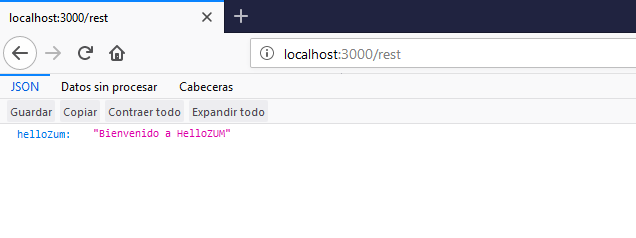


Ubicaremos el archivo que se visualiza en la imagen y lo modificaremos como se encuentra en la captura.

Después detendremos el comando “npm start” con CTRL+C y lo volveremos a ejecutar, accedemos a la ruta y se apreciará lo siguiente.



Entramos a la ruta ‘localhost:3000/rest’ y veremos lo siguiente.

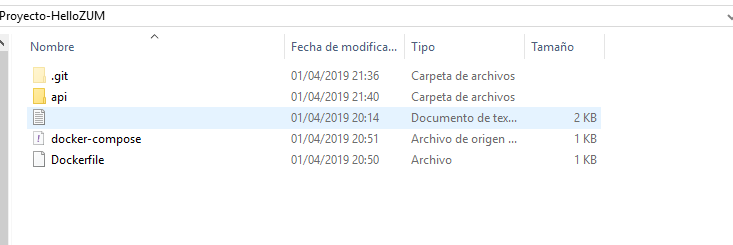


Por ahora ya tenemos terminado nuestro servicio, es hora de “dockerizarlo” ¡.

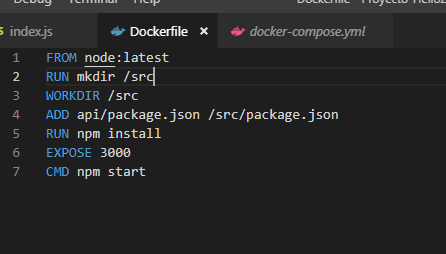
# **3- “Dockerizando” Nuestro servicio.**

Como ya sabemos Docker trabaja con contenedores, y hace mucho más fácil el desarrollo en un equipo e incluso facilita el envío a producción. En esta sección veremos cómo configurar el dockerfile y el Docker-compose.yml para esta mini App.

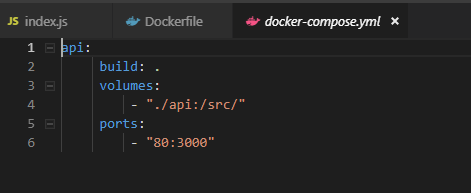
Primero crearemos estos 2 archivos que se muestran en la imagen.



1. Docker-compose.yml
2. Dockerfile



Este es el contenido del Dockerfile. Donde básicamente es crear una imagen de node:latest, donde “latest” es el tag, que significa que traerá la última versión de Dockerhub. Creará un directorio llamado “src” donde ahí estará nuestra aplicación, donde justamente con el comando “add” copiaremos de nuestro archivo local “package.json” al del contenedor, después instalaremos nuestras dependencias con el archivo archivo “package.json” , expondrá el puerto 3000 y por ultima ejecutará NODE.JS.



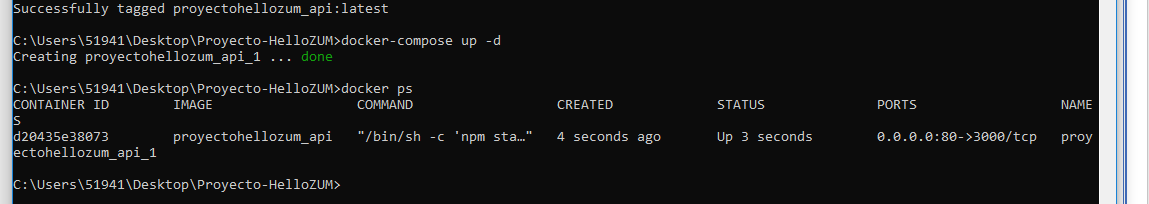
El archivo docker-compose básicamente construye nuestro Dockerfile, “volumes”, permite copiar los archivos de nuestra máquina al Docker para que de esta forma podamos trabajar desde fuera sin la necesidad de meter los archivos al Docker cada que los actualizamos y por último con “ports” es como accederemos a este, mediante el puerto 80 ingresaremos al puerto 3000 de nuestro contenedor.

Y listo es hora de ejecutar el docker-compose, ejecutaremos los siguientes comandos en este orden:

docker-compose build

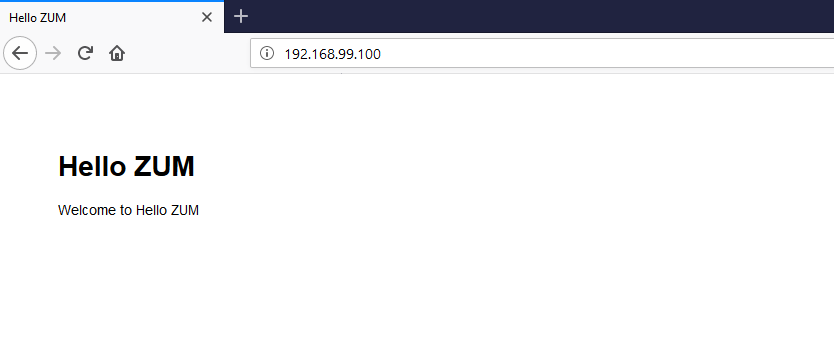
docker-compose up -d

docker ps

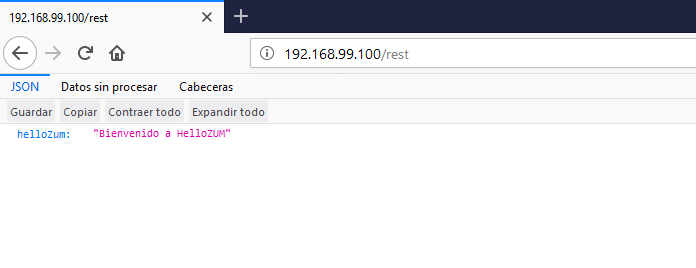


Tendremos una sección como esta, donde nos dice que el contendor está ejecutado y está exponiendo un puerto 3000, nosotros accederemos por el puerto 80.



Dependiendo de la configuración en algunos casos se accede con “localhost:80” o en mi caso con accederé con la ip que muestra el comando “docker-machine ip”.

Ahora accederemos a la ruta “/rest”:



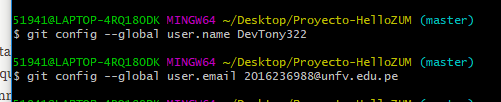
Y listo ya está todo funcionado ¡

# **4-Subiendo nuestro proyecto a GITHUB**

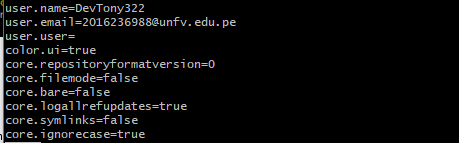
Ahora con nuestro bash de git abierto procederemos a realizar los siguiente.

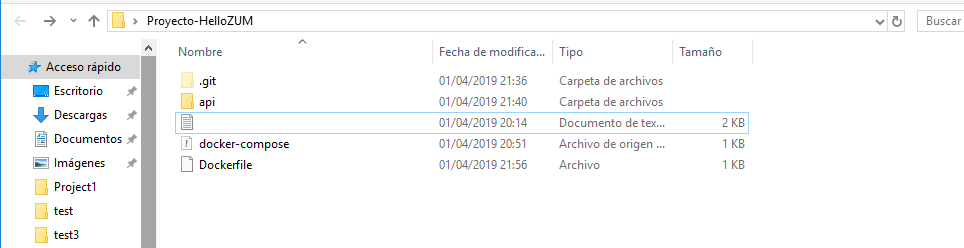
Primero recuerda haber creado una cuenta en GITHUB, la puedes crear desde su página oficial: <https://github.com/>.

Una vez creado tendremos que registrar nuestros datos en nuestro GIT.



Con el comando “ git config –l “, verificamos que todo este correcto.



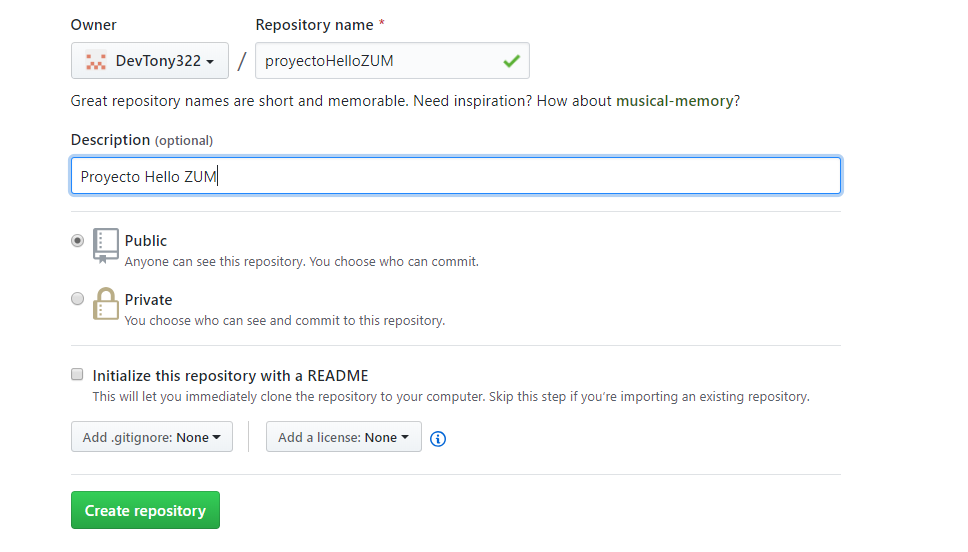
Antes de realizar nuestro commit, agregaremos un archivo “. gitignore”, para evitar subir archivos innecesarios a nuestro repositorio en github.

Ahora haremos nuestro commit

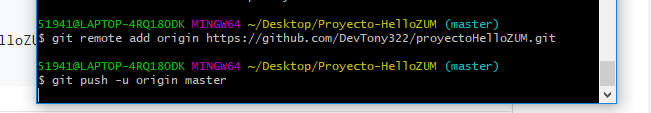




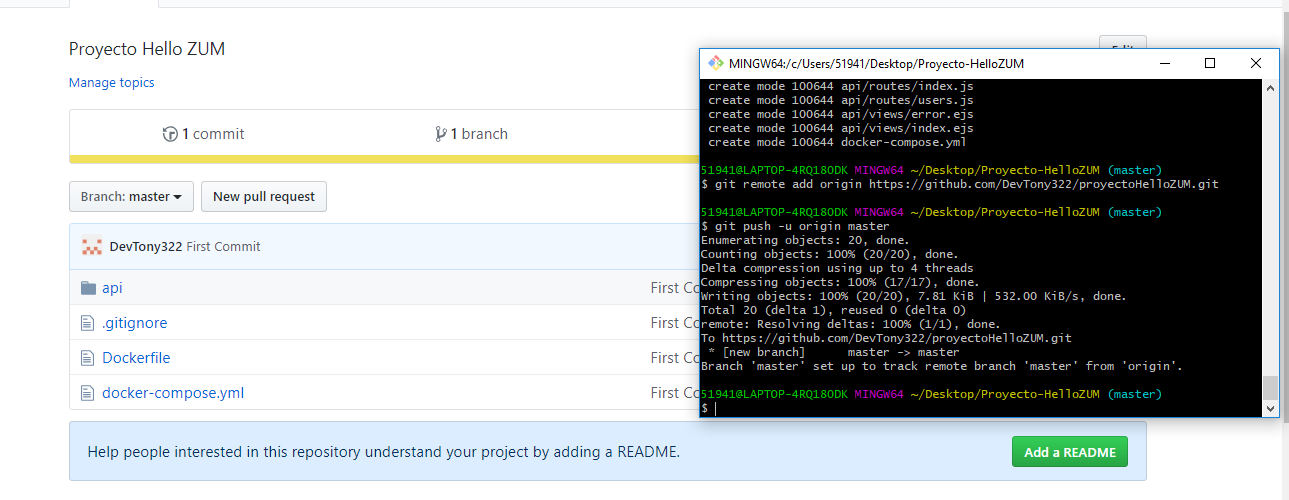
Ahora, tendremos que crear un repositorio en github.



Creamos el repositorio y por último escribiremos los siguientes comandos:



Donde agregamos nuestro repositorio en github y subimos todos nuestros cambios a la rama máster.



Y vemos que todo se subió correctamente.

:D