Samuel Sebastián Cruz González

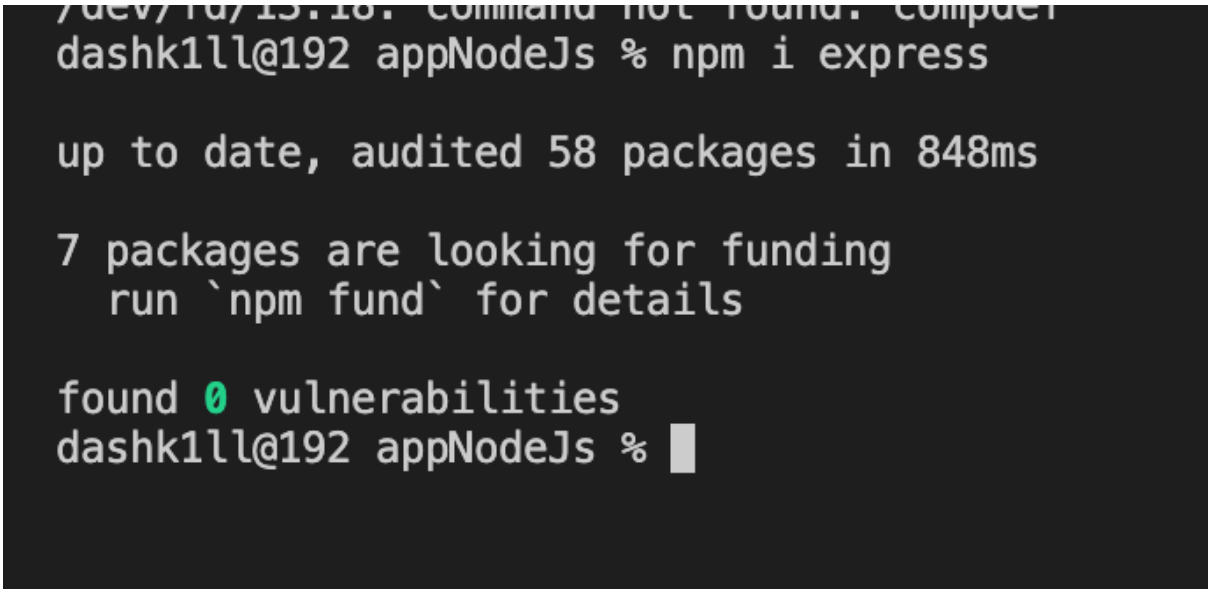
Práctica REST API y Contenedores
NodeJS & Visual Studio Code

```
dashk1ll@192 appNodeJs % npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (appnodejs)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /Users/dashk1ll/Desktop/Practica/appNodeJs/package.json:

{
  "name": "appnodejs",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes) yes
dashk1ll@192 appNodeJs %
```

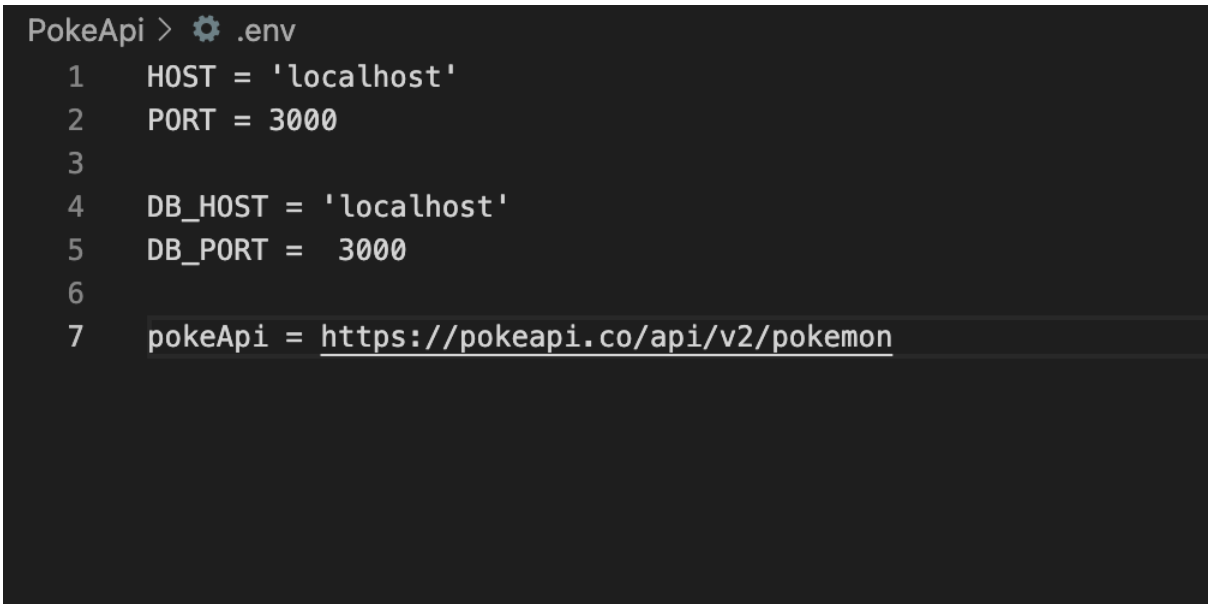
Inicializar el proyecto con el comando *npm init*
Generará un archivo de manera automática en donde se podrán configurar los módulos que se vayan instalando posteriormente.

Samuel Sebastián Cruz González

Instalar módulo *express* para crear nuestro servidor con *NodeJs*



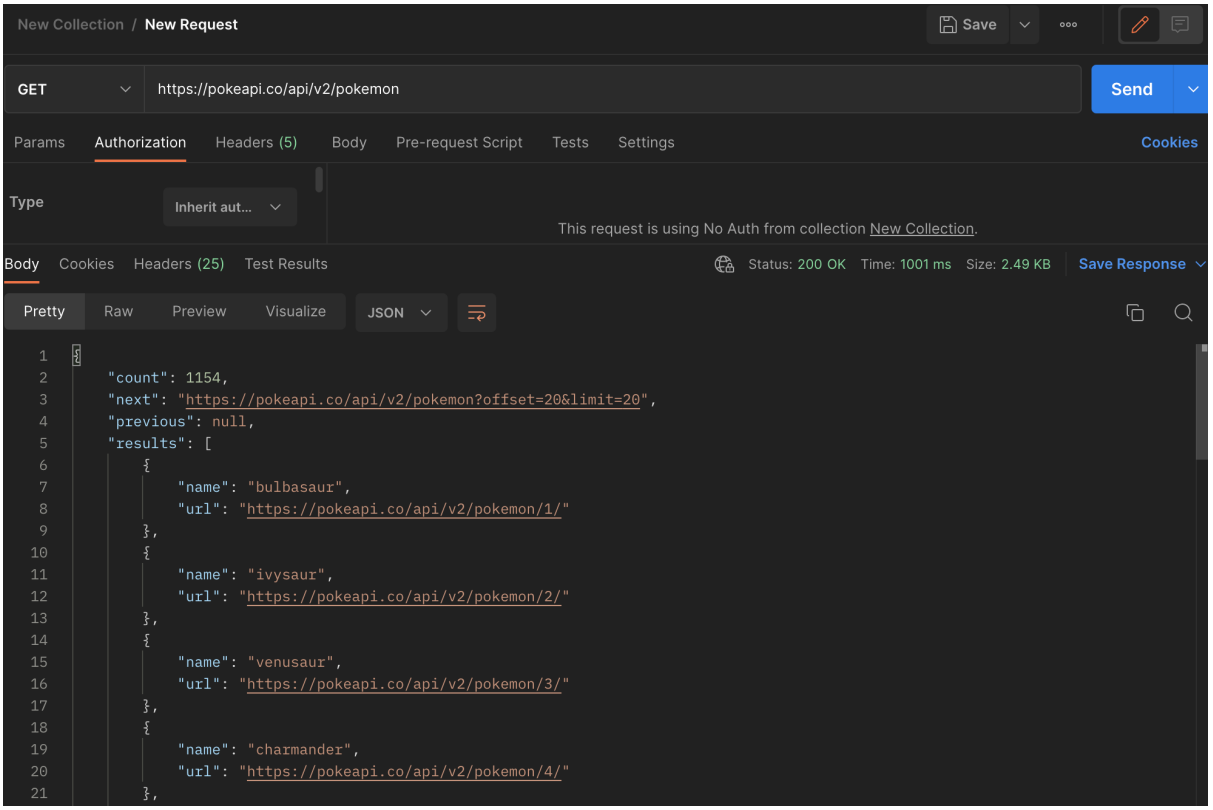
Crear nuestro archivo *.env* para nuestras variables de entorno y proteger nuestras variables que son de relevancia para nuestro proyecto



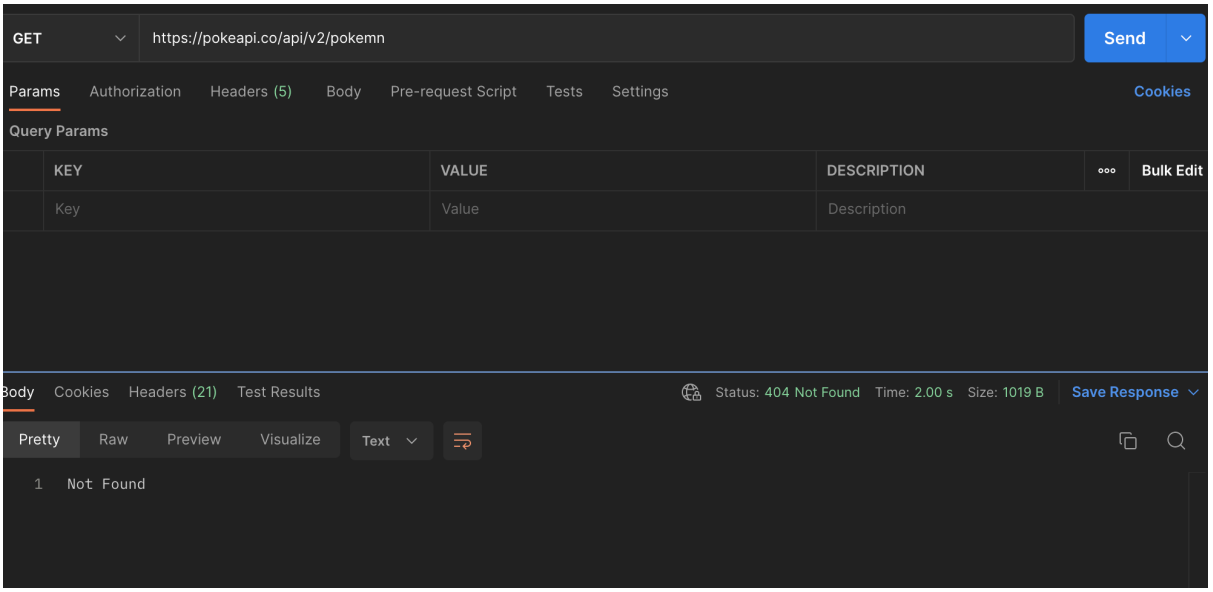
Postman

Postman nos permite conocer de qué manera trabaja nuestra API, generando gets, posts o si es necesario ingresar claves para tener acceso a nuestras APIS, adecuando los

parámetros requeridos o las claves solo si nuestra API lo requiere



Como se puede observar si nos devuelve un código 200 significa que nuestra petición se ha realizado de manera correcta, sin embargo si no es así, nos arrojará otros códigos



En este caso nuestra api está mal direccionada y nos devuelve un valor 404 que es un *no encontrado "Not Found".*

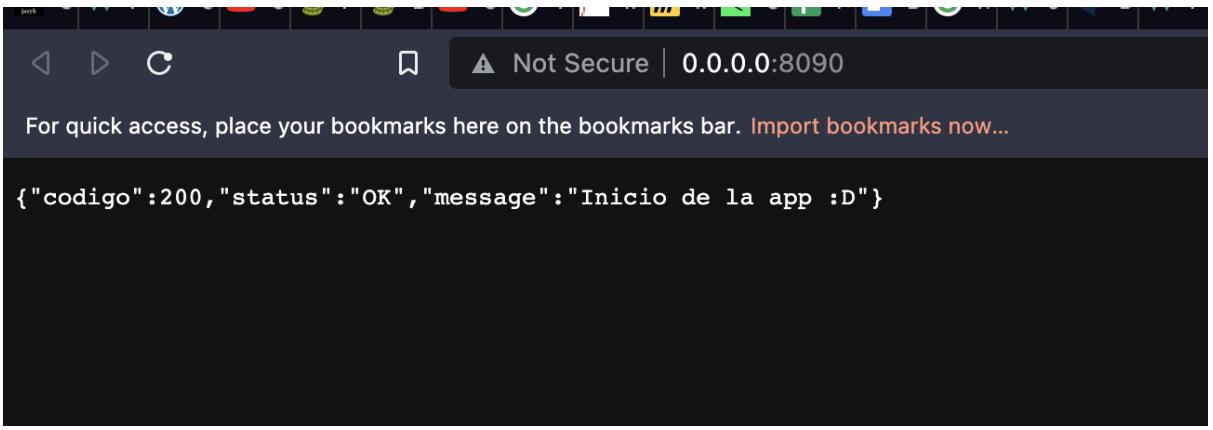
NodeJs

Debemos generar nuestra app que nos permitirá gestionar nuestra aplicación desde los endpoints, importar los módulos que se usarán, crear nuestras funciones para nuestra app y levantar nuestro servidor.

Samuel Sebastián Cruz González

Dentro de nuestra app podemos generar JSON de respuesta de acuerdo a los estados de nuestra conexión.

```
//Importación de modulos
const express = require('express');
const app = express();
require("dotenv").config()
const serviceApi = require('../PokeApi/pokeApi')

app.use(express.json())


//Iniciar nuestro servidor
app.listen(process.env.PORT, () => {
    console.log(`El servidor esta iniciando en http://${process.env.HOST}:${process.env.PORT}`);
});

app.get("/", (req, res) => {
  let respuesta = {
    codigo: 200,
    status : "OK",
    message: "Inicio de la app :D",
  }
  res.json(respuesta);
})
//endpoint inicial
app.get("/pokemon",async function (req, res)  {
  try {
      let poke = await serviceApi.getPoke()
      res.send(poke)
  } catch (error) {
    let finalE = {
      message: "Ocurrió un error",
      error: error.message,

    }
    res.send(finalE)
  }
})
```



Se requiere el módulo node-fetch para generar el JSON, sin embargo me salió este error

```
/Users/dashk1ll/Desktop/Practica/appNodeJs/PokeApi/pokeApi.js:4
const fetch = require('node-fetch')
              ^

Error [ERR_REQUIRE_ESM]: require() of ES Module /Users/dashk1ll/Desktop/Practica/appNodeJs/node_modules/node-fetch/src/index.js from /Users
/dashk1ll/Desktop/Practica/appNodeJs/PokeApi/pokeApi.js not supported.
```

Ya que el instalador agrega la última versión de este módulo y no está del todo optimizada, entonces si aparece el error se debe desinstalar el módulo

`npm uninstall node-fetch`

e instalar la versión 2 que es la versión estable

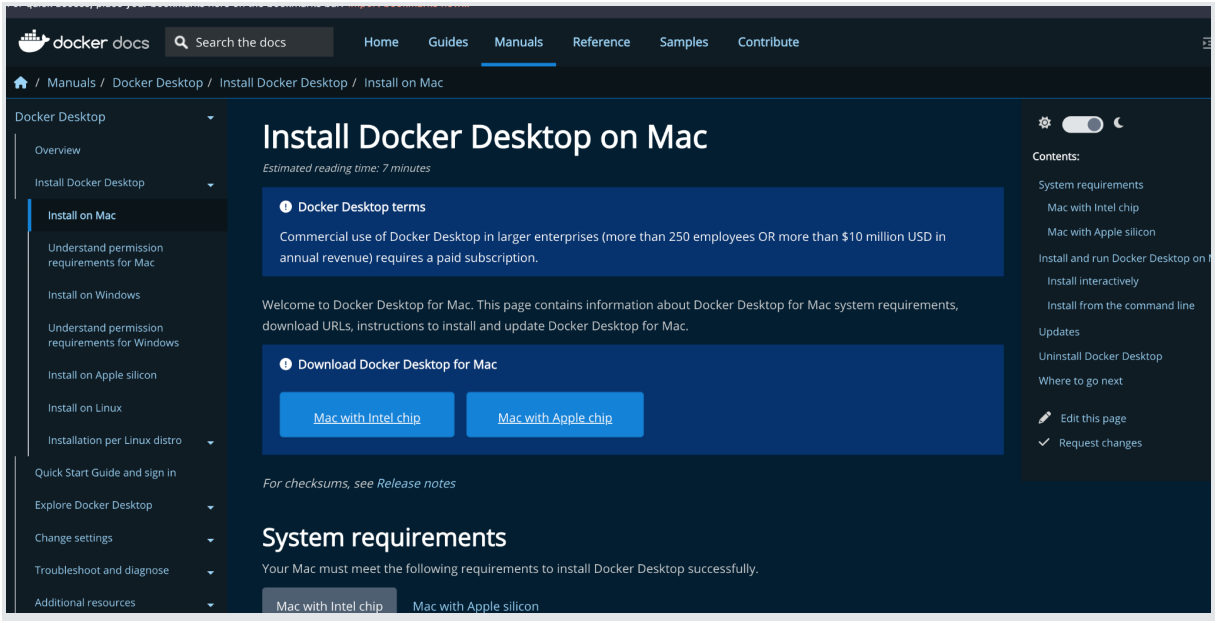
`npm install node-fetch@2`

Nota: Si tienes un error similar al pasado lo mejor es revisar la documentación e instalar las versiones de los módulos que sean más estables o no se encuentren en versiones de desarrollo

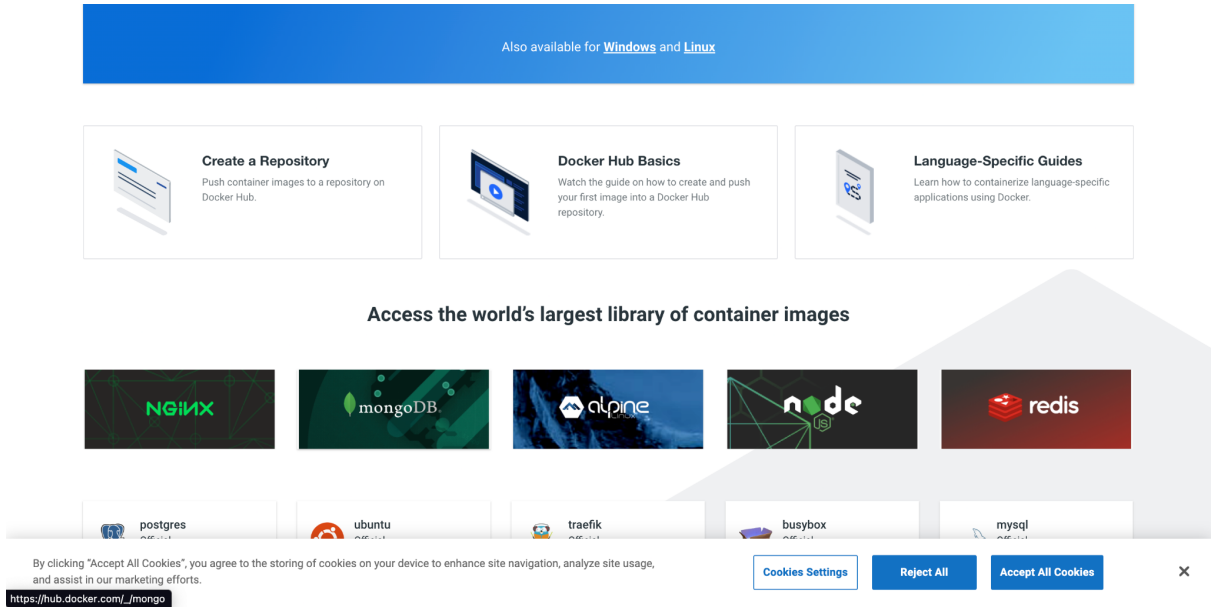
Samuel Sebastián Cruz González

Docker

Instalar docker desde el sitio oficial con el sistema operativo correspondiente (en mi caso MacOs)



Ingresar al sitio oficial https://hub.docker.com/ y descargar la imagen en nuestro caso usaremos un servidor nginx



Y creamos nuestra imagen para eventualmente crear nuestro contenedor

```
docker pull nginx
```

```
dashk1ll@192 appNodeJs % docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
31b3f1ad4ce1: Pull complete
fd42b079d0f8: Pull complete
30585fbbebc6: Pull complete
18f4ffdd25f4: Pull complete
9dc932c8fba2: Pull complete
600c24b8ba39: Pull complete
Digest: sha256:0b970013351304af46f322da1263516b188318682b2ab1091862497591189ff1
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

Creamos nuestro contenedor con el siguiente comando y nos devolverá nuestra imagen

```
dashk1ll@192 appNodeJs % docker run --name server-nginx -d -p 8090 nginx
cb123079b6d61d8961e06855aa1aa5e4bd6cb288ece226e620dd0f39bbabda60
dashk1ll@192 appNodeJs %
```

Comprobamos que se haya creado nuestro contendor con el comando

`docker ps -a`

```
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS                                   NAMES
cb123079b6d6   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp, 0.0.0.0:54570->8090/tcp   server-nginx
dashk1ll@192 appNodeJs %
```

Modificamos nuestras variables de entorno colocando nuestro host y nuestro puerto

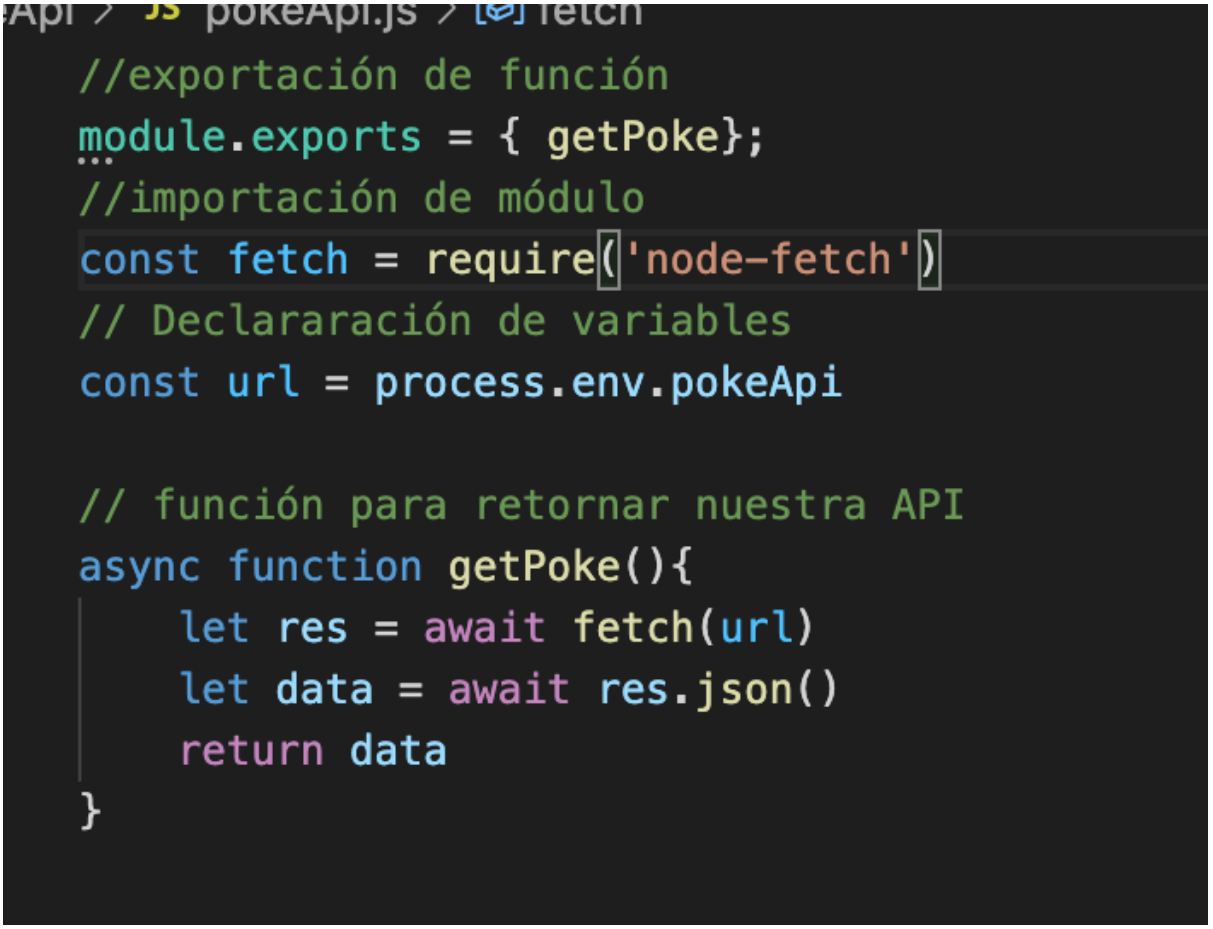
```
HOST = 0.0.0.0
PORT = 8090


pokeApi = https://pokeapi.co/api/v2/pokemon
```

Y comprobamos que nuestro contenedor está funcionando correctamente con nuestra app con la información que nos devuelve en consola

```
dashk1ll@192 PokeApi % node app.js
El servidor esta iniciando en http://0.0.0.0:8090
```

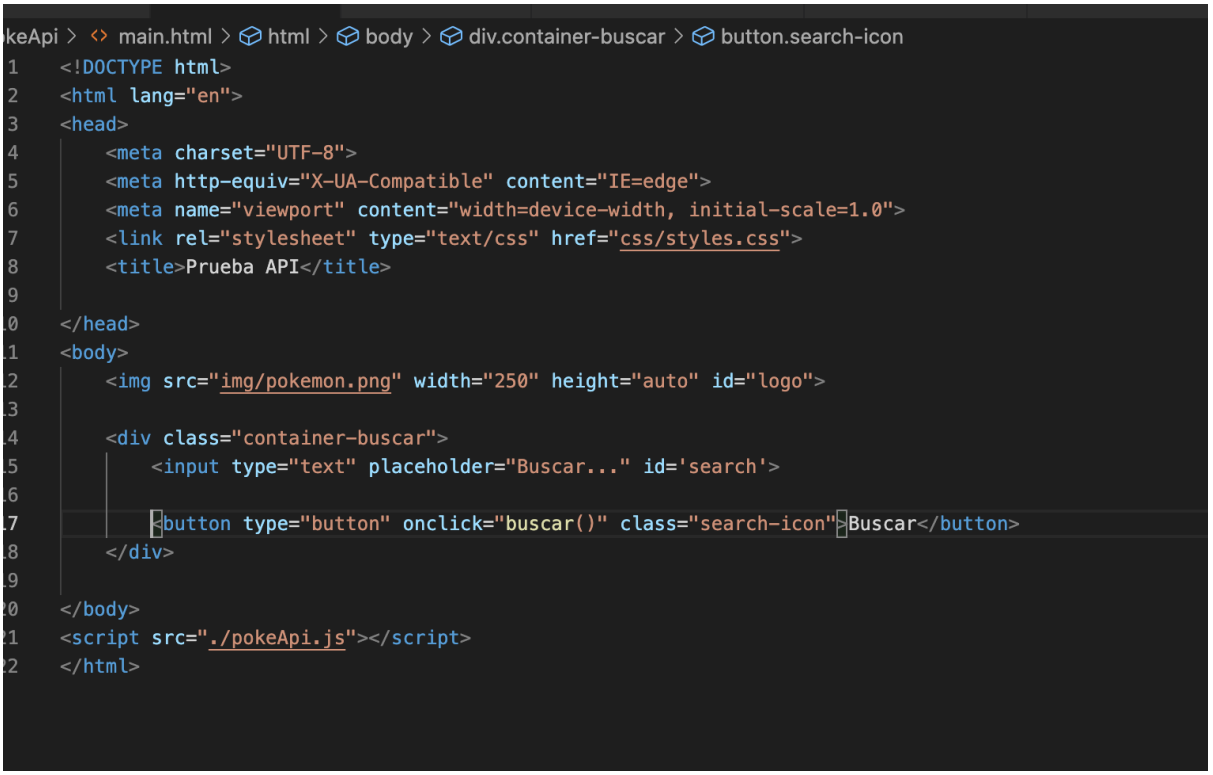
Samuel Sebastián Cruz González

El código siguiente corresponde a la obtención de nuestra API generando un fetch y convirtiéndolo a un formato JSON



Ahora crearemos un archivo HTML y CSS para tener una maquetación y estilo para presentar nuestra API

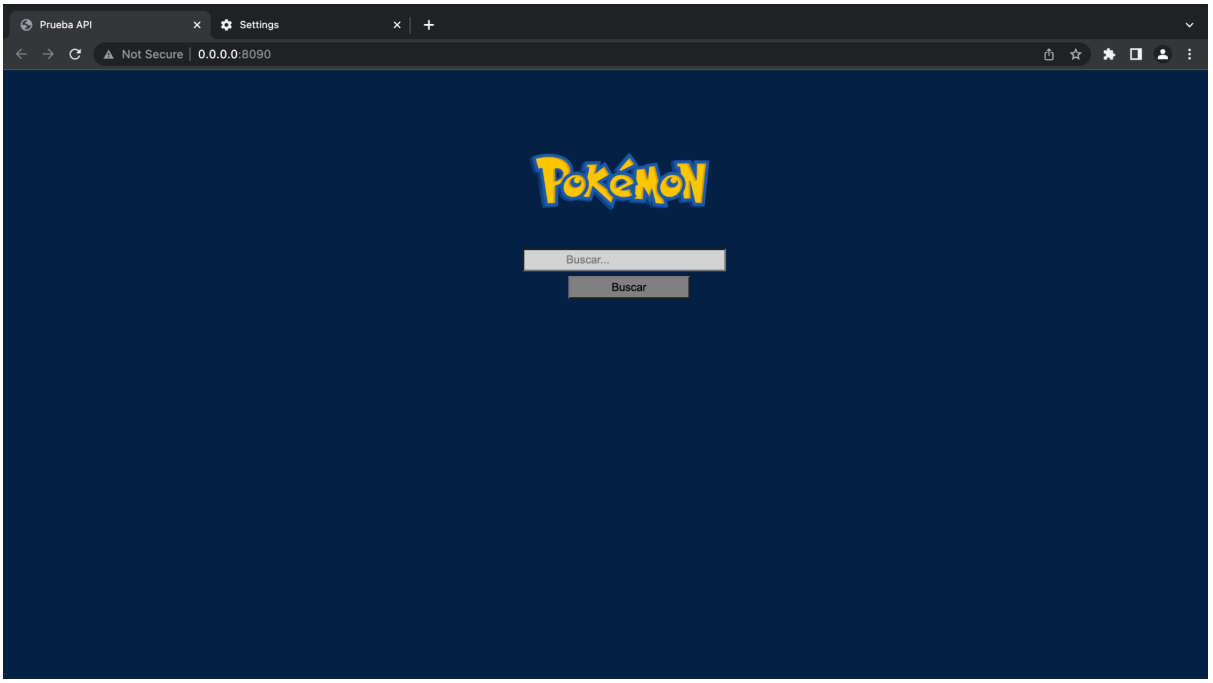
Samuel Sebastián Cruz González



```
keApi > main.html > html > body > div.container-buscar > button.search-icon
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="css/styles.css">
    <title>Prueba API</title>

</head>
<body>
    <img src="img/pokemon.png" width="250" height="auto" id="logo">

    <div class="container-buscar">
        <input type="text" placeholder="Buscar..." id='search'>

        <button type="button" onclick="buscar()" class="search-icon">Buscar</button>
    </div>

</body>
<script src="./pokeApi.js"></script>
</html>
```

```css
*{
    background-color: #032044;
}
#logo{

    position: absolute;
    top: 10%;
    right: 40%;

}

input[type="text"]{
    background-color: lightgray;
    color: black;
    padding: 4px 50px;
    position: absolute;
    top: 26%;
    right: 40%;
}

.search-icon{
    background-color: grey;
    color: black;
    padding: 4px 50px;
    position: absolute;
    top: 30%;
    right: 43%;
}
```

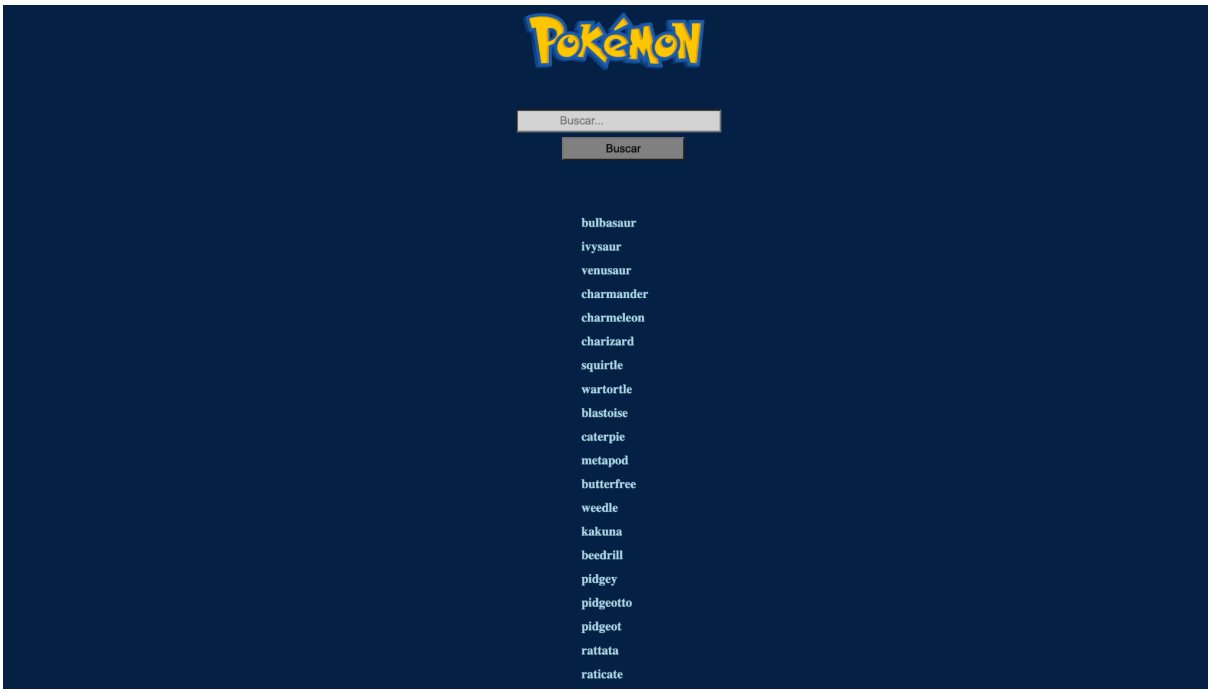
Samuel Sebastián Cruz González



Usamos DOM para generar nuestro fetch a nuestro JSON

```
// Declaración de variables
const url = "https://pokeapi.co/api/v2/pokemon/"
const principal = document.getElementById("principal")
console.log(url)



document.addEventListener('DOMContentLoaded', buscar(), false)

function buscar() {
    let input = document.getElementById('search').value
    getPoke(input)

}

async function getPoke(pokemon){
    let resultPoke = url + pokemon
    fetch(resultPoke)
        .then(response => response.json())
        .then(json => {
            for (let i = 0; i < resultPoke.length; i++) {
                const myDiv = document.createElement('div')
                myDiv.className = 'div-mostrar'
                const para = document.createElement('h1')
                para.textContent = `${json.results[i].name}`
                console.log(json.results[i].name)
                myDiv.appendChild(para)
                principal.appendChild(myDiv)
            }
        }).catch(err => {
            console.error("Error: ", err)
        })
}
```

Y nos devolverá los valores en nuestra página con los pokemones



GitHub

Iniciamos nuestro repositorio con el comando

```
git init
```

Añadimos todos nuestros archivos con el comando

```
git add .
```

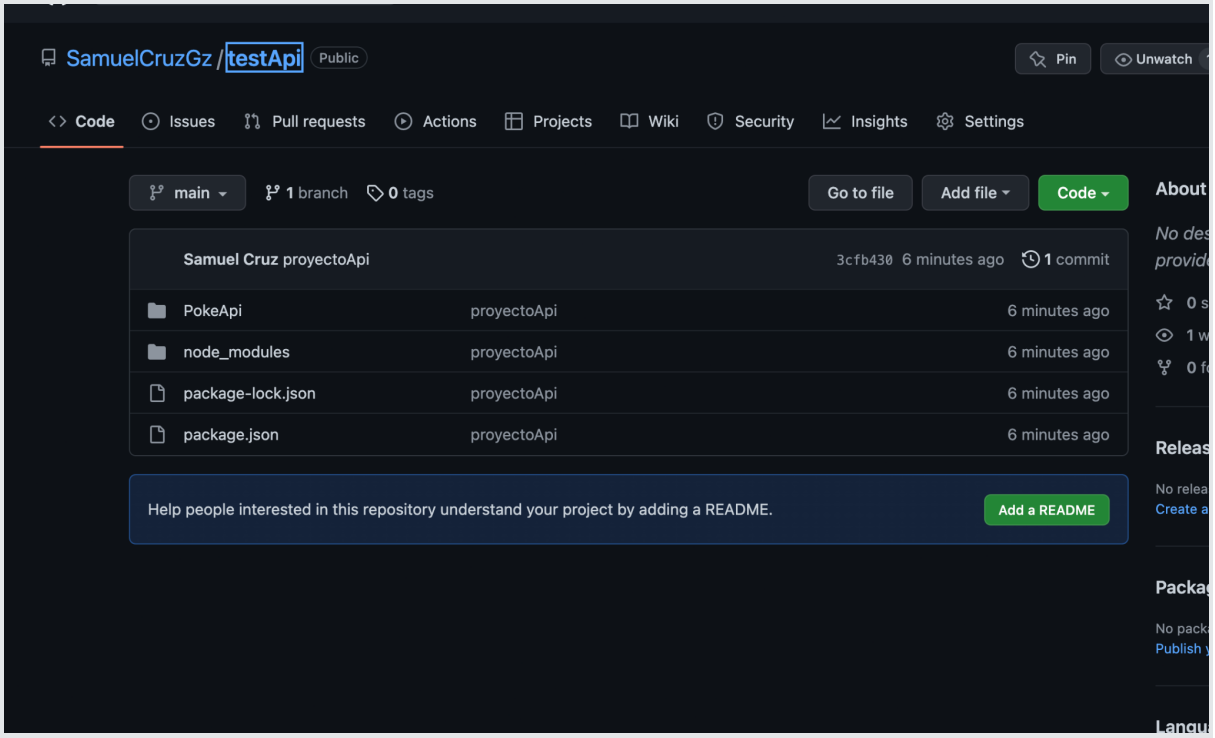
Realizamos nuestro primer comentario sobre nuestro repositorio con el comando

```
git commit -m “comentario”
```

Finalmente enlazamos nuestro repositorio con nuestra cuenta y generamos el push

```
git remote add origin https://github.com/NOMBRE_USUARIO/NOMBRE_PROYECTO.git

git push -u origin master
```

Samuel Sebastián Cruz González



Swagger

Usamos la herramienta swagger para documentar el funcionamiento de nuestra API, con los métodos que este puede tener

```yaml
          example: sdAasf3tgAFs214gD12
      responses:
        200:
          description: Exito en la consulta
          content:
            application/json:
              schema:
                type: array
                items:
                  properties:
                    name:
                      type: string
                      example: ditto
                    url:
                      type: string
                      example: https://pokeapi.co/api/v2
                        /pokemon/1/
        400:
          description: Token de acceso expirado/invalido
          content:
            application/json:
              schema:
                type: string
                example:  {"message" : "Acess token expired
                  /invalid"}
    post:
      description: Obtener pokemons y access token
      parameters:
        - in: query
          name: username
          description: Codigo de acceso para el usuario
```

```
                /invalid"}
    post:
      description: Obtener pokemons y access token
      parameters:
        - in: query
          name: username
          description: Codigo de acceso para el usuario
          required: true
          schema:
            type: string
            example: vxl9xtOaBQqlzn4cGZ
        - in: query
          name: password
          description: Código de acceso para la contraseña
          required: true
          schema:
            type: string
            example: CJoyZNah
        - in: query
          name: pokemon
          description: Nombre del pokemon
          required: true
          schema:
            type: string
            example: ditto
      responses:
        201:
          description: Pokemon correcto
        400:
          description: 'Pokemon invalido'
```

```yaml
openapi: 3.0.0
info:
  version: "1.0.0"
  title: Simple Inventory API
  description: Documentación de Pokemon
  contact:
    name: Samuel Cruz
    url: https://rucmahe-eval-prod.apigee.net/v1/poke-samuel
    email: you@your-company.com
paths:
  /pokemon:
    get:
      description: Permite obtener los datos de los
        pokemones
      parameters:
        - in: query
          name: access_token
          description: el token que brinda acceso a la API
          required: true
          schema:
            type: string
            example: sdAasf3tgAFs214gD12
      responses:
        200:
          description: Exito en la consulta
          content:
            application/json:
              schema:
                type: array
                items:
                  properties:
```

Y los estados que puede tener durante el proceso de ejecución

200 *No links*

Exito en la consulta

Media type

application/json

Controls `Accept` header.

Example Value | Schema

```
[
  {
    "name": "ditto",
    "url": "https://pokeapi.co/api/v2/pokemon/1/"
  }
]
```

400 *No links*

Token de acceso expirado/invalido

Media type

application/json

Example Value | Schema

```
{
  "message": "Acess token expired/invalid"
}
```