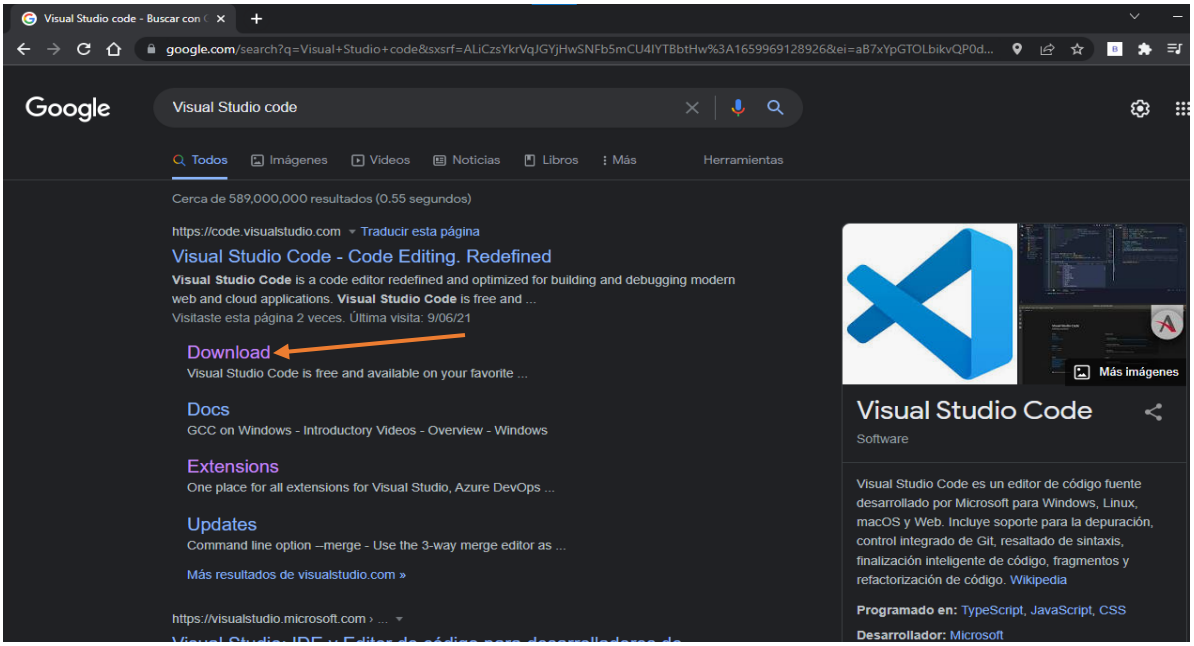
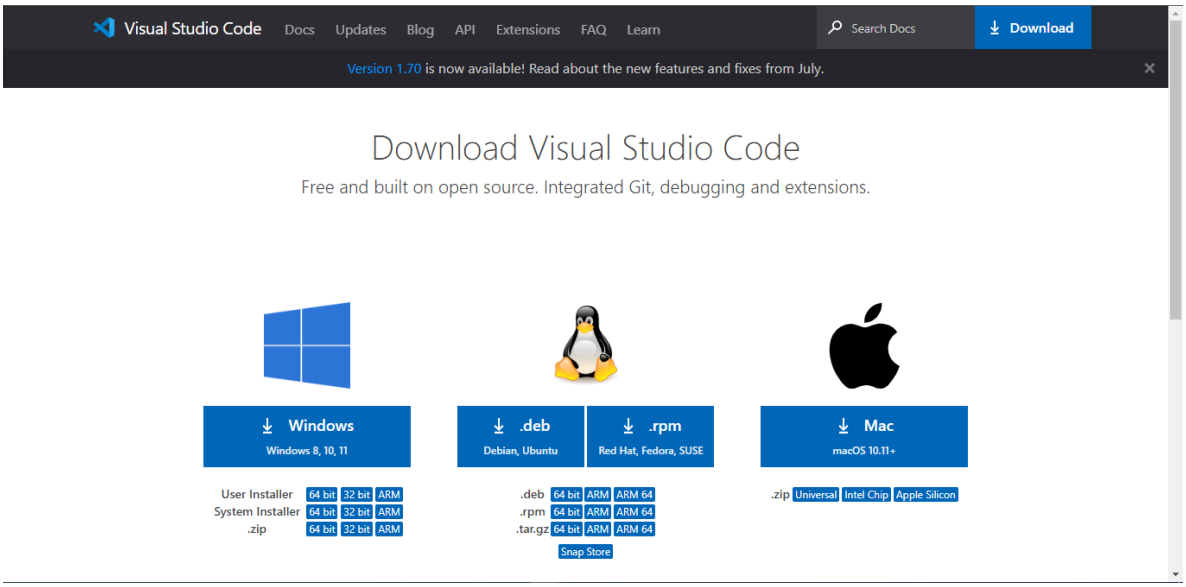
Proceso instalación Visual Studio Code

Paso 1: Escribimos en Google Visual Studio Code y seleccionamos donde dice “Download”.

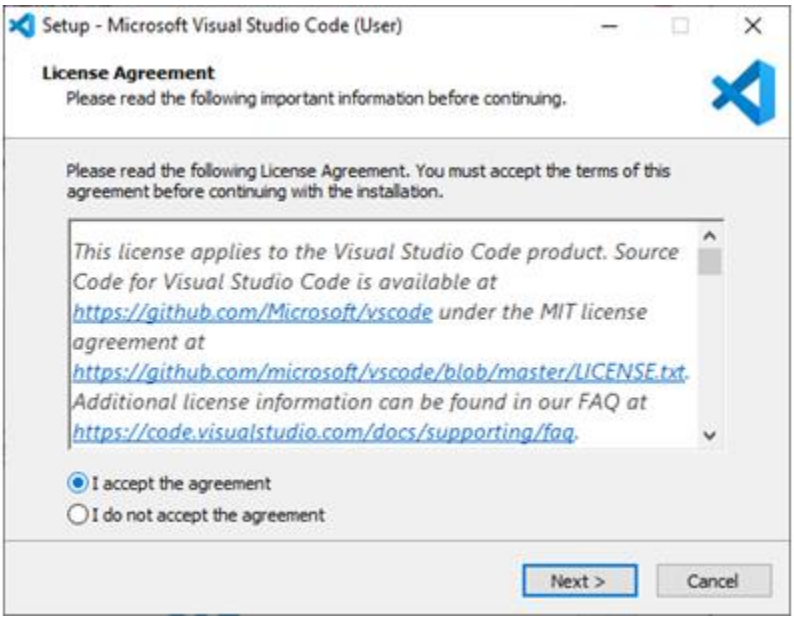


Paso 2: Seleccionamos el sistema operativo que tenemos y lo descargamos.

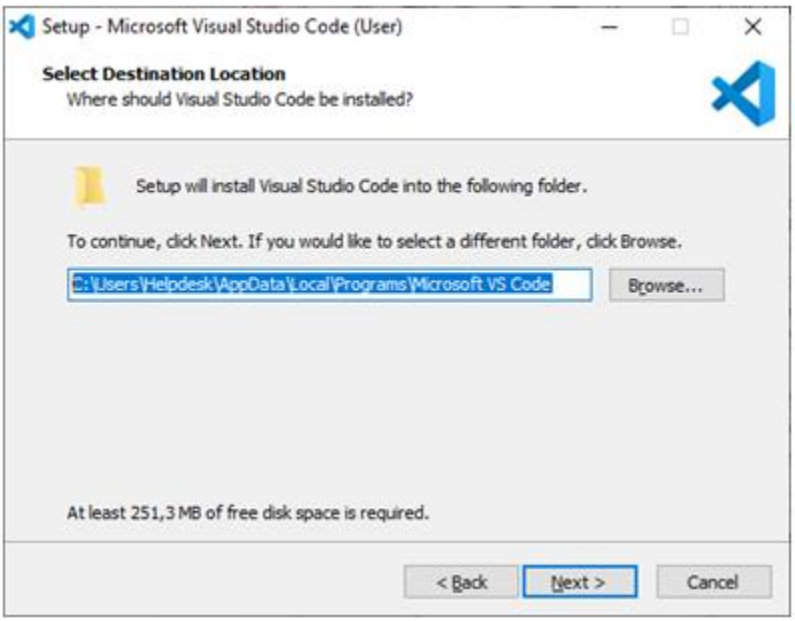


Paso 3: Al darle clic nos descargará un .exe, al cual le daremos clic encima.

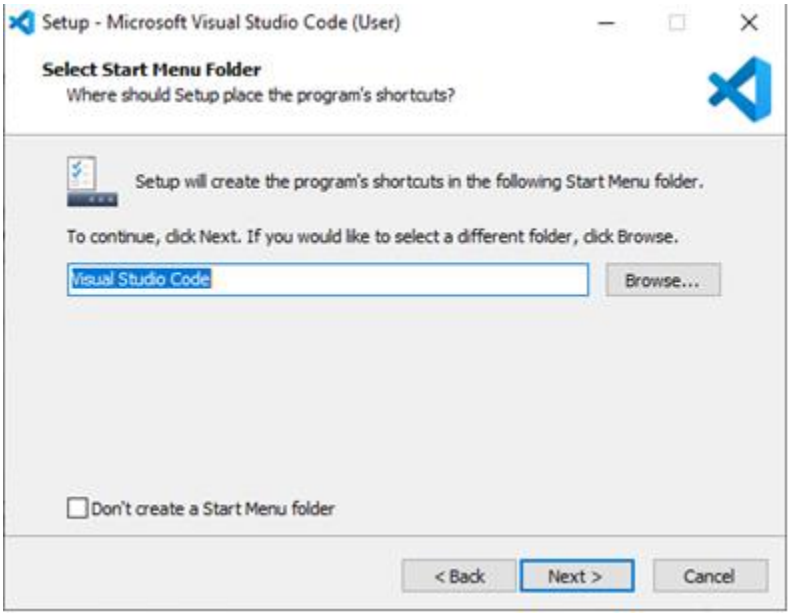
Paso 4: Lee y acepta el acuerdo de licencia. Haz clic en Next para continuar.



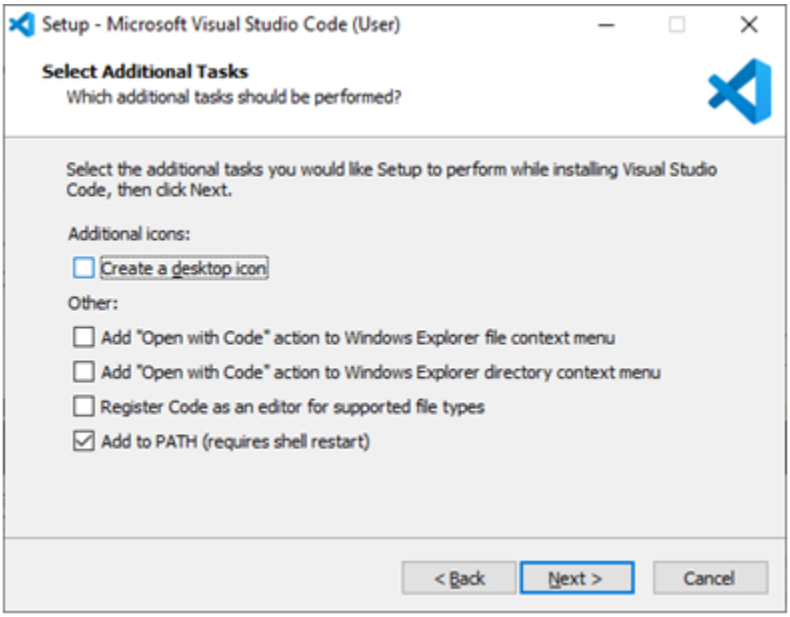
Paso 5: Puedes cambiar la ubicación de la carpeta de instalación o mantener la configuración predeterminada. Haz clic en Next para continuar.



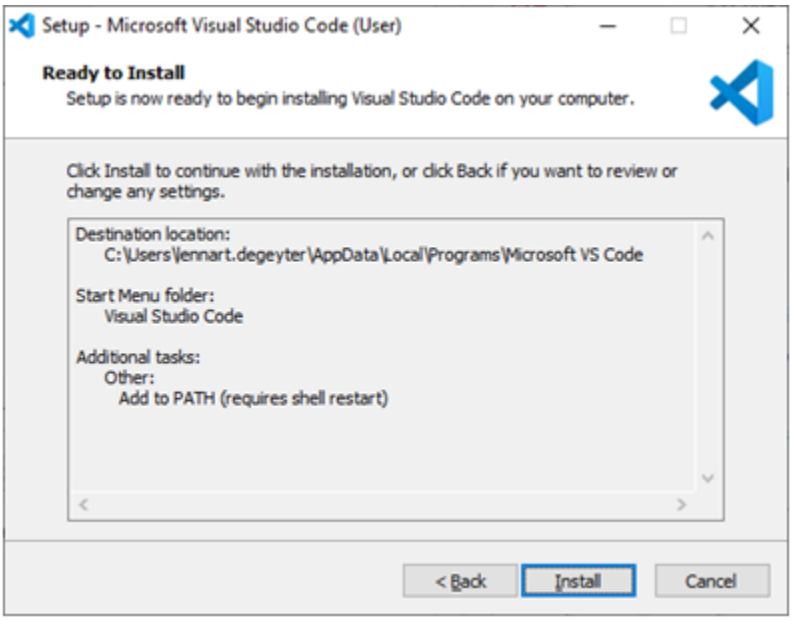
Paso 6: Elige si deseas cambiar el nombre de la carpeta de accesos directos en el menú Inicio o si no deseas instalar accesos directos en absoluto. Haz clic en Next.



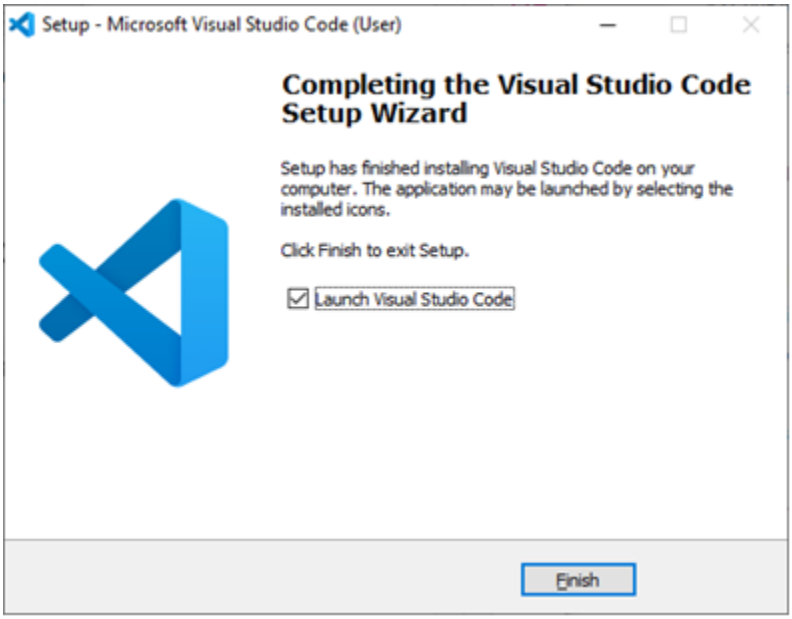
Paso 7: Selecciona las tareas adicionales, por ej. crear un icono en el escritorio o añadir opciones al menú contextual de Windows Explorer. Haz clic en Next.



Paso 8: Haz clic en Install para iniciar la instalación.

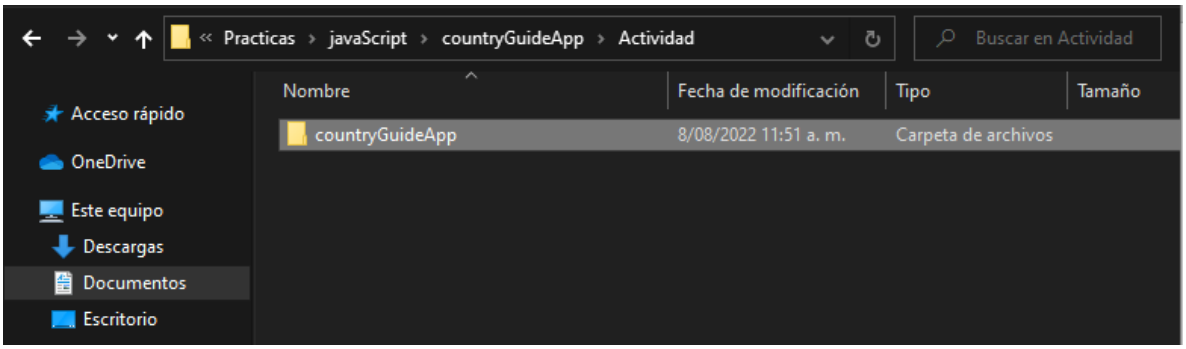


Paso 9: El programa está instalado y listo para usar. Haz clic en Finish para finalizar la instalación y lanzar el programa.

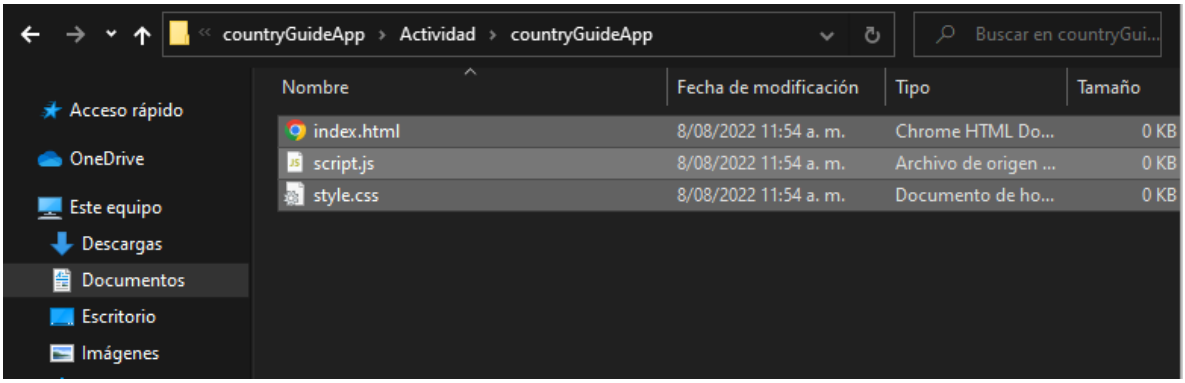


Creación de archivos

Creamos una carpeta raíz. En nuestro caso la llamaremos “countryGuideApp”.



Dentro de esta, crearemos tres archivos. El primero llamado index.html, otro llamado style.css y el último llamado script.js (Es importante tener exactamente esa extensión en el archivo).



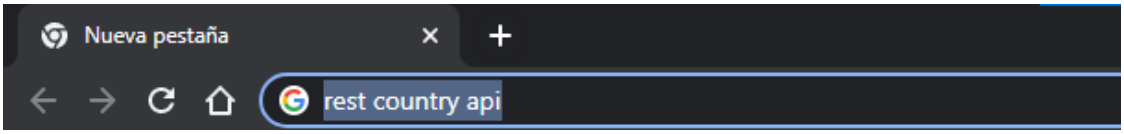
Country Guide App

Empezamos creando el html raíz dentro del archivo index.html, y de una vez linkeamos el css y el js, también importamos la fuente de letra que vamos a utilizar.

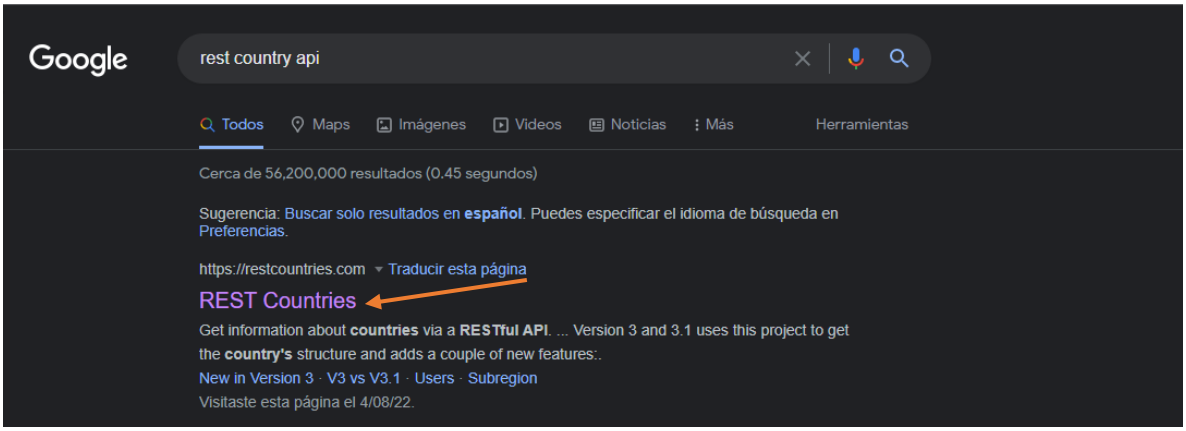
```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Country Guide App</title>
        <!-- Importamos la fuente que vamos a utilizar -->
        <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500&display=swap"rel="stylesheet">
        <!-- Linkeamos el css -->
        <link rel="stylesheet" href="style.css">
    </head>
    <body>

        <!-- Linkeamos el js -->
        <script src="script.js"></script>
    </body>
</html>
```

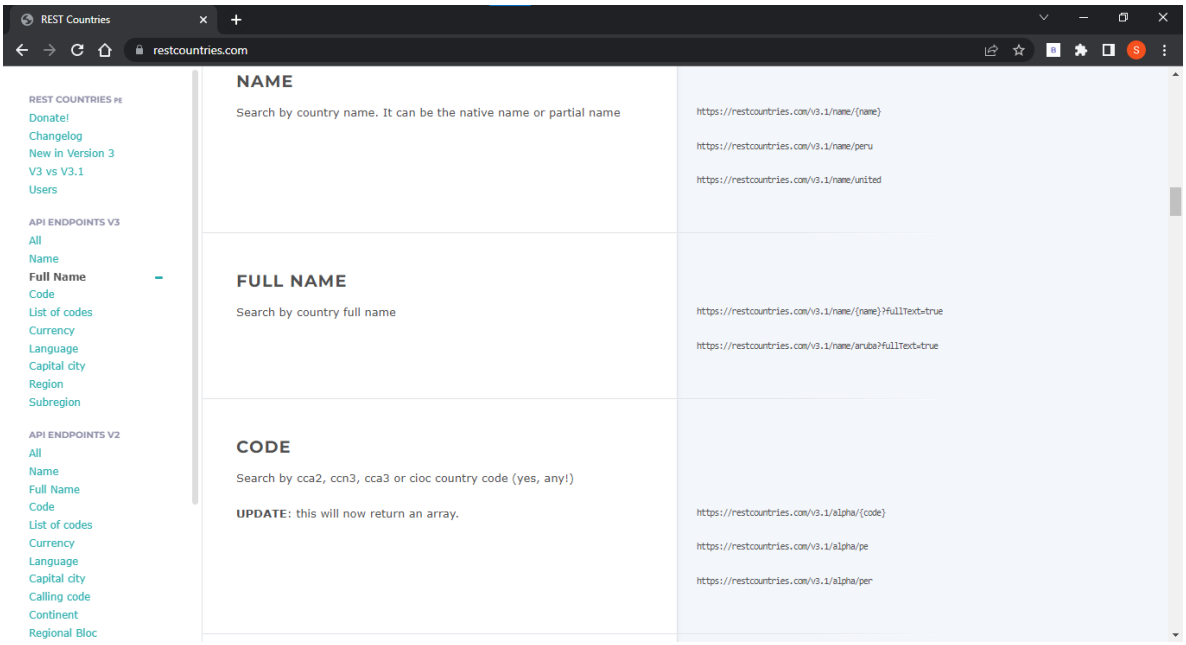
Luego buscamos la api que vamos a consumir, esta se llama “rest country api”.



Lo buscamos en Google, y le damos clic a la primera opción que nos sale.



Luego ahí dentro, bajamos hasta donde dice full name.



Luego copiamos el primer link de la derecha, ese será el url necesario para obtener los datos de los países.

Dentro del archivo html, crearemos dos contenedores, uno con la sección para buscar y el otro con el resultado que nos arroja.

```
13    <body>
14        <div class="container">
15            <div class="search-wrapper">
16                <input type="text" id="country-inp" placeholder="Enter a country name here...">
17                <button id="search-btn">Search</button>
18            </div>
19            <div id="result">
20
21            </div>
22        </div>
```

Ahora, vamos a estilizar esos contenedores.

```css
/* Estilos para toda la pagina como tal */
* {
    padding: 0;
    margin: 0;
    box-sizing: border-box;
    font-family: "Poppins", sans-serif;
}

/* Estilos para el color de la pagina */
body {
    background-color: #3d64e6;
}

/* Estilos para el contenedor */
.container {
    background-color: #ffffff;
    width: 80vw;
    max-width: 37.5em;
    padding: 3em 2.5em;
    position: absolute;
    transform: translate(-50%, -50%);
    top: 50%;
    left: 50%;
    border-radius: 0.62em;
    box-shadow: 0 1.5em 1.8em rgba(8, 21, 65, 0.25);
}

/* Estilos para el contenedor para buscar */
.search-wrapper {
    display: grid;
    grid-template-columns: 9fr 3fr;
    gap: 1.25em;
}

/* Estilos para el boton en el contenedor para buscar */
.search-wrapper button {
    font-size: 1em;
    background-color: #3d64e6;
    color: #ffffff;
    padding: 0.8em 0;
    border: none;
    border-radius: 1.5em;
}
```

```css
/* Estilos para el input en el contenedor para buscar */
.search-wrapper input {
    font-size: 1em;
    padding: 0 0.62em;
    border: none;
    border-bottom: 2px solid #3d64e6;
    outline: none;
    color: #222a43;
}

/* Estilos para el contenedor de resultado */
#result {
    margin-top: 1.25em;
}

/* Estilos para la bandera */
.container .flag-img {
    display: block;
    width: 45%;
    min-width: 7.5em;
    margin: 1.8em auto 1.2em auto;
}

/* Estilos para los textos, exactamente los h2 */
.container h2 {
    font-weight: 600;
    color: #222a43;
    text-align: center;
    text-transform: uppercase;
    letter-spacing: 2px;
    margin-bottom: 1.8em;
}

/* Estilos para la informacion */
.data-wrapper {
    margin-bottom: 1em;
    letter-spacing: 0.3px;
}

/* Estilos para los textos, exactamente los h4 */
.container h4 {
    display: inline;
    font-weight: 500;
    color: #222a43;
}
```

```css
/* Estilos para los span que esten dentro de un container */
.container span {
    color: #5d6274;
}

/* Estilos para los textos, exactamente los h3 */
.container h3 {
    text-align: center;
    font-size: 1.2em;
    font-weight: 400;
    color: #ff465a;
}
```

Luego de tener estilos, la página se verá bonita, pero aún falta la funcionalidad. Vamos al js para realizarla. Primero llamamos por id el input y el botón de search.

```
// Se definen dos variables donde se guardarán los elementos llamados por id
let searchBtn = document.getElementById('search-btn');
let countryInp = document.getElementById('country-inp');

```

Ahora escuchamos el evento que se realiza en el botón, que deberá ser “click”. Dentro de esto obtenemos el nombre que el usuario digitó y después completamos la url del api para poder que nos traiga toda la información de ese país.

```
// Se escucha el evento del boton
searchBtn.addEventListener('click', () => {
    // Se obtiene el nombre del pais que digita el usuario
    let countryName = countryInp.value;
    // Utilizamos la API y le enviamos el nombre del pais para completar la URL
    let finalURL = `https://restcountries.com/v3.1/name/${countryName}?fullText=true
    `;

```

Luego, para saber en qué posición y donde están ubicados los datos, imprimimos por consola buscando lo que necesitamos (Cuando ya hayamos terminado todo, se podrán comentar todos los console.log()).

```
    // Generamos una respuesta utilizando la URL donde se almacenan todos los datos
    fetch(finalURL)
    .then((response) => response.json())
    .then((data) => {
        // Prueba para saber donde estan ubicados los datos y los mostramos en consola
        console.log(data[0]);
        console.log(data[0].capital[0]);
        console.log(data[0].flags.svg);
        console.log(data[0].name.common);
        console.log(data[0].continents[0]);
        console.log(Object.keys(data[0].currencies)[0]);
        console.log(data[0].currencies[Object.keys(data[0].currencies)].name);
        console.log(Object.values(data[0].languages).toString().split(",").join(","));
```

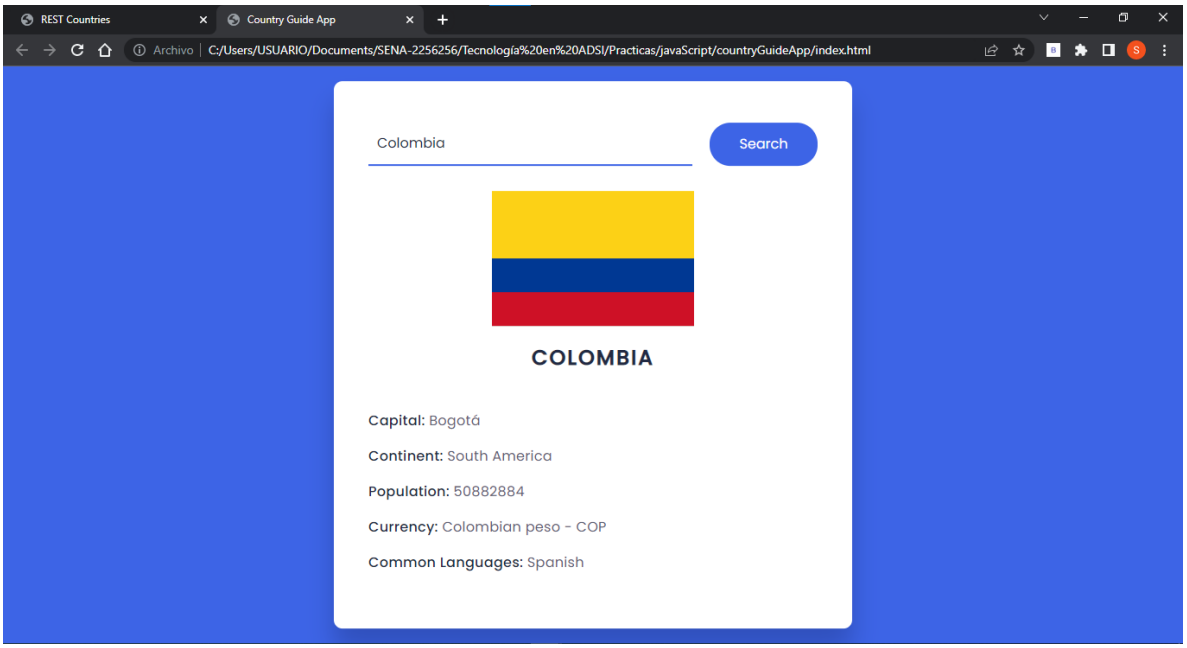
Ahora, nos debería mostrar los datos por consola. Como ya sabemos donde están los datos que vamos a utilizar, ahora los mostraremos dentro del aplicativo.

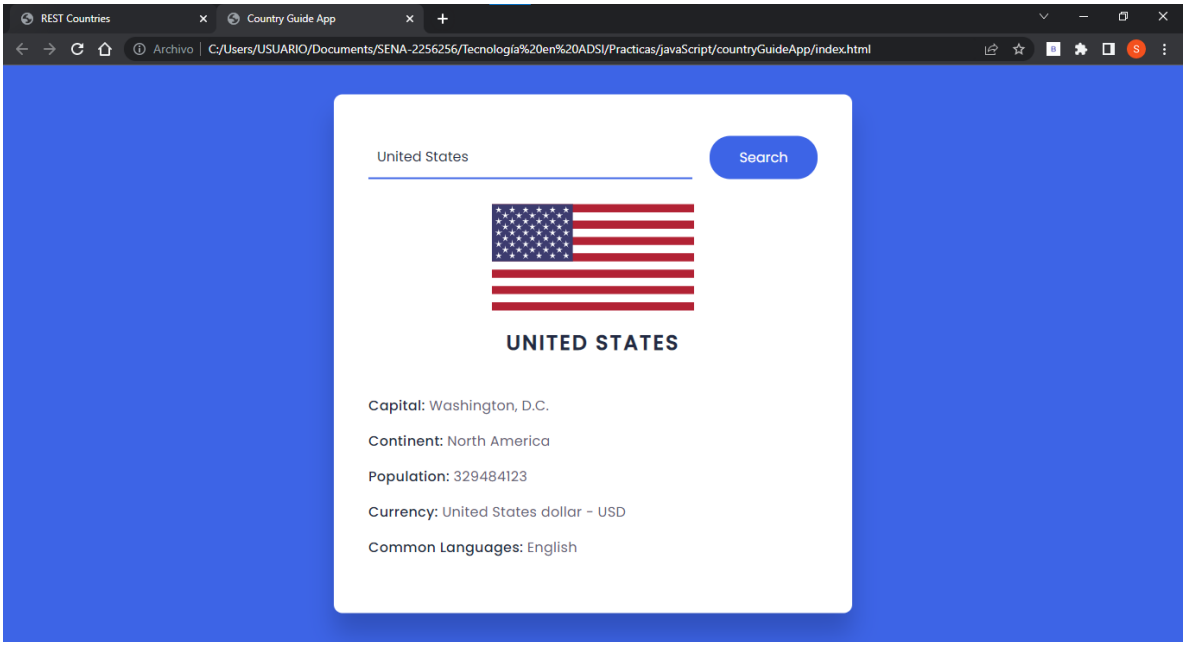
```
        // Insertamos codigo html donde se mostraran los datos basicos del pais
        // Tales como bandera, nombre, continente donde esta el pais, cantidad de poblacion,
        // la moneda y los lenguajes comunes
        result.innerHTML = `
            <img src="${data[0].flags.svg}" class="flag-img">
            <h2>${data[0].name.common}</h2>
            <div class="wrapper">
                <div class="data-wrapper">
                    <h4>Capital:</h4>
                    <span>${data[0].capital[0]}</span>
                </div>
            </div>
            <div class="wrapper">
                <div class="data-wrapper">
                    <h4>Continent:</h4>
                    <span>${data[0].continents[0]}</span>
                </div>
            </div>
            <div class="wrapper">
                <div class="data-wrapper">
                    <h4>Population:</h4>
                    <span>${data[0].population}</span>
                </div>
            </div>
            <div class="wrapper">
                <div class="data-wrapper">
                    <h4>Currency:</h4>
                    <span>${data[0].currencies[Object.keys(data[0].currencies)].name} - ${Object.keys(data[0].currencies)}</span>
                </div>
            </div>
            <div class="wrapper">
                <div class="data-wrapper">
                    <h4>Common Languages:</h4>
                    <span>${Object.values(data[0].languages).toString().split(",").join(",")}</span>
                </div>
            </div>
        `;
    // Evaluamos si no se ingresa nada y enviamos un mensaje y tambien si lo escrito no coincide
    })
```

Y ya por último evaluamos si no se ingresa nada en el input o si lo ingresado no es correspondiente.

```
    // Evaluamos si no se ingresa nada y enviamos un mensaje y tambien si lo escrito no coincide
    }).catch(()=>{
        if(countryName.length == 0){
            result.innerHTML = `
                <h3>The input field cannot be empty</h3>
            `;
        } else {
            result.innerHTML = `
                <h3>Please enter a valid country name</h3>
            `;
        }
    });
});
```

Ahora que terminamos de codificar, el resultado debería ser parecido a lo siguiente:


REST Countries
Country Guide App
Archivo | C:/Users/USUARIO/Documents/SENA-2256256/Tecnología%20en%20ADSI/Practicas/javaScript/countryGuideApp/index.html
Colombia
Search
COLOMBIA
Capital: Bogotá
Continent: South America
Population: 50882884
Currency: Colombian peso - COP
Common Languages: Spanish


REST Countries
Country Guide App
Archivo | C:/Users/USUARIO/Documents/SENA-2256256/Tecnología%20en%20ADSI/Practicas/javaScript/countryGuideApp/index.html
United States
Search
UNITED STATES
Capital: Washington, D.C.
Continent: North America
Population: 329484123
Currency: United States dollar - USD
Common Languages: English