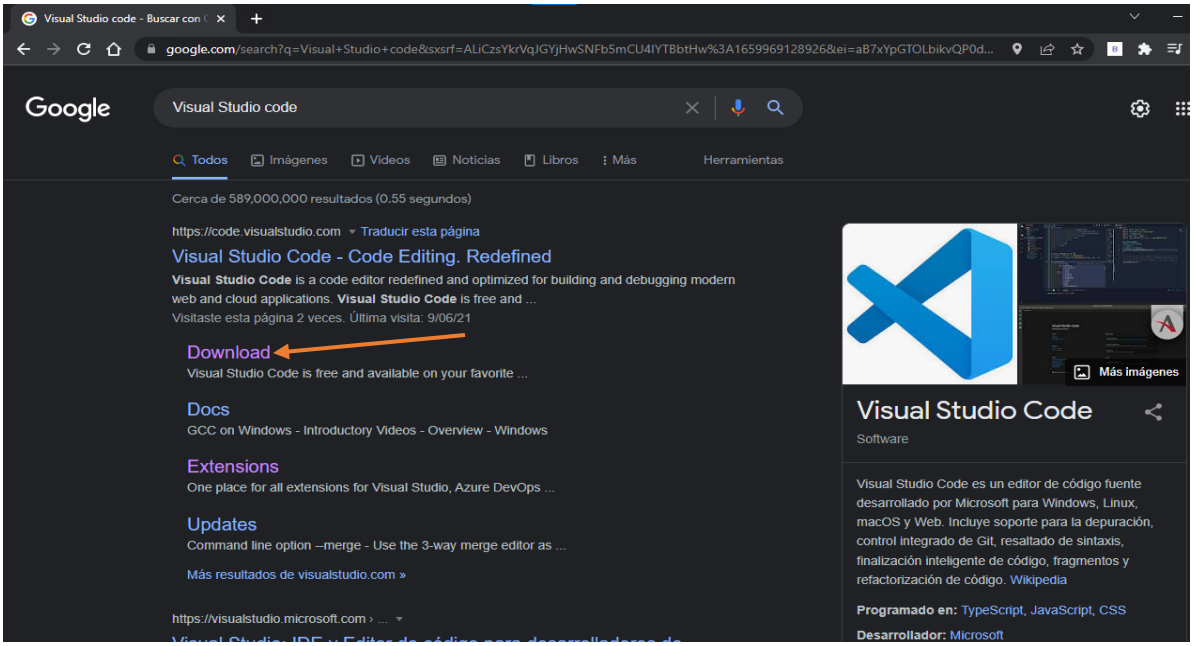
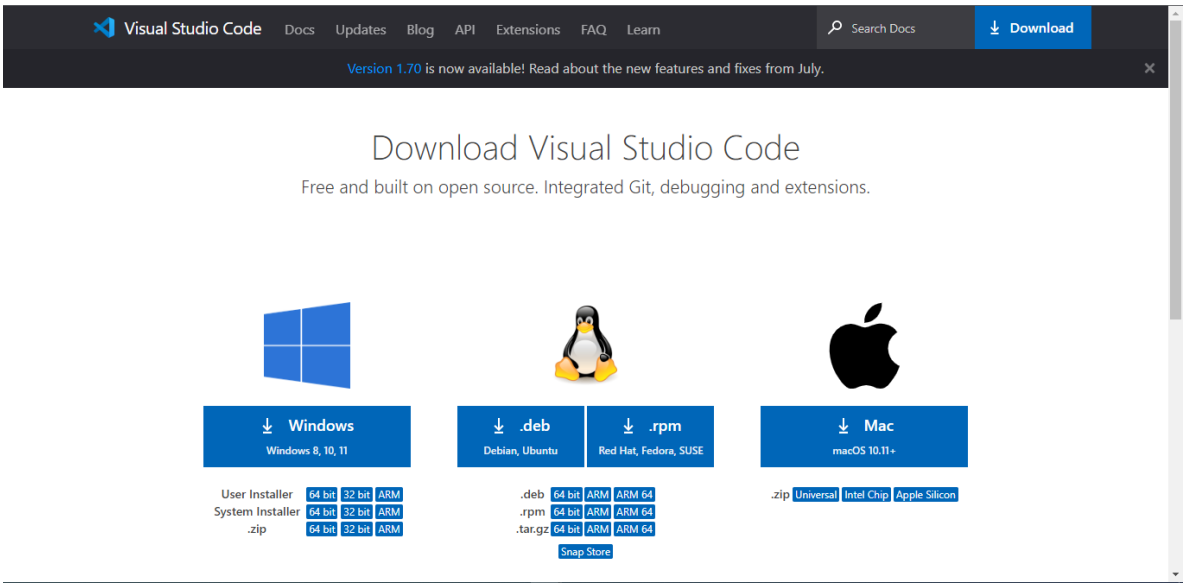
Proceso instalación Visual Studio Code

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

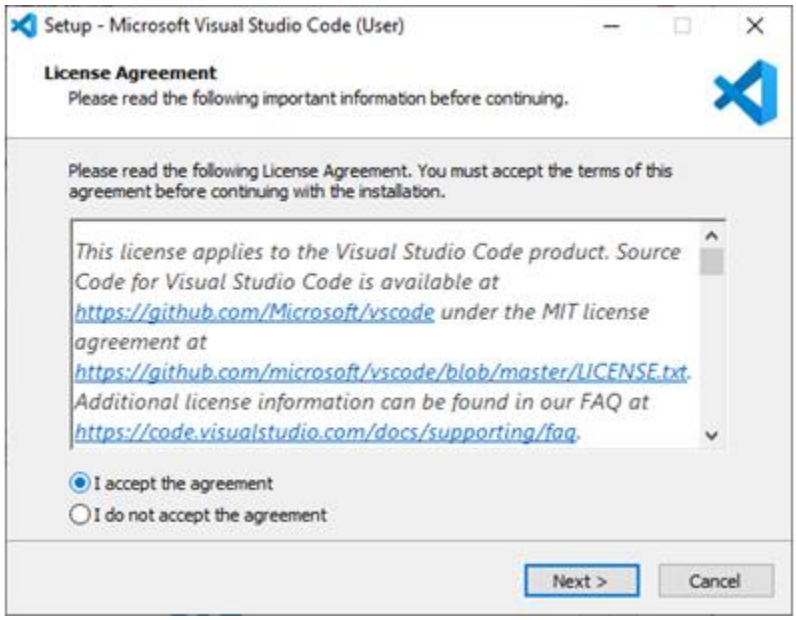


Paso 2: Seleccionamos el sistema operativo que tenemos y lo descargamos.

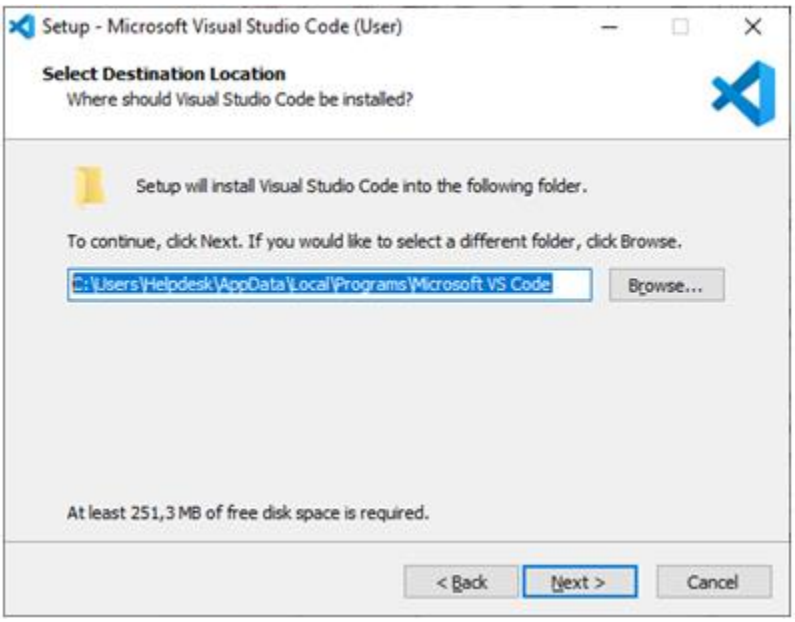


Paso 3: Al darle clic nos descargará un .exe, al cual le daremos clic encima.

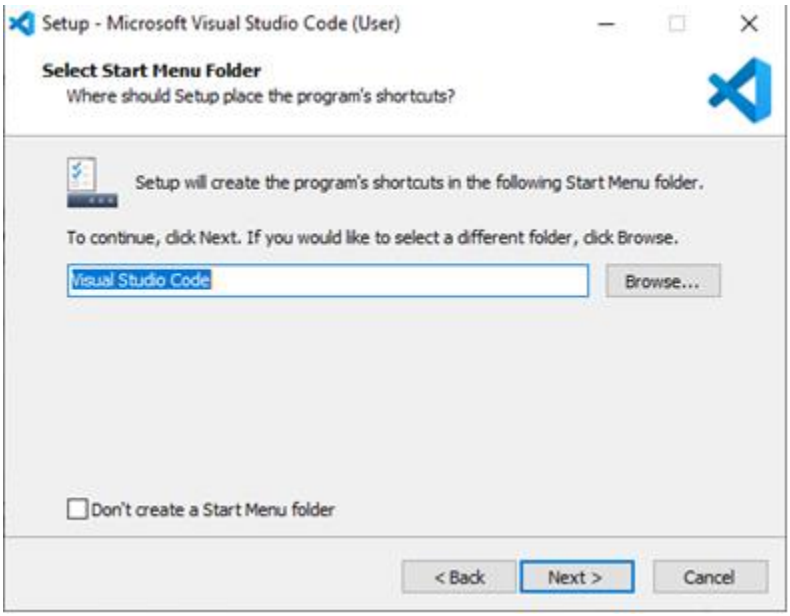
Paso 4: Lee y acepta el acuerdo de licencia. Haz clic en Next para continuar.



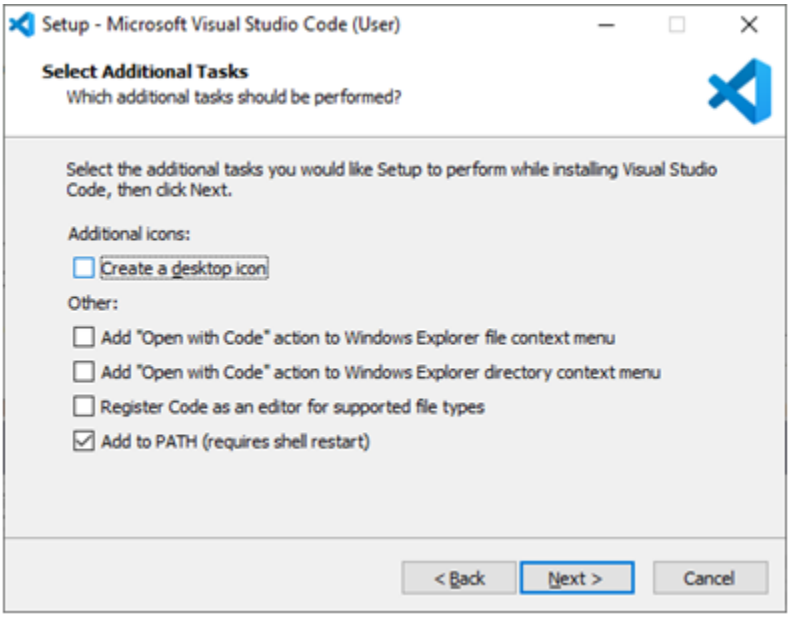
Paso 5: Puedes cambiar la ubicación de la carpeta de instalación o mantener la configuración predeterminada. Haz clic en Next para continuar.



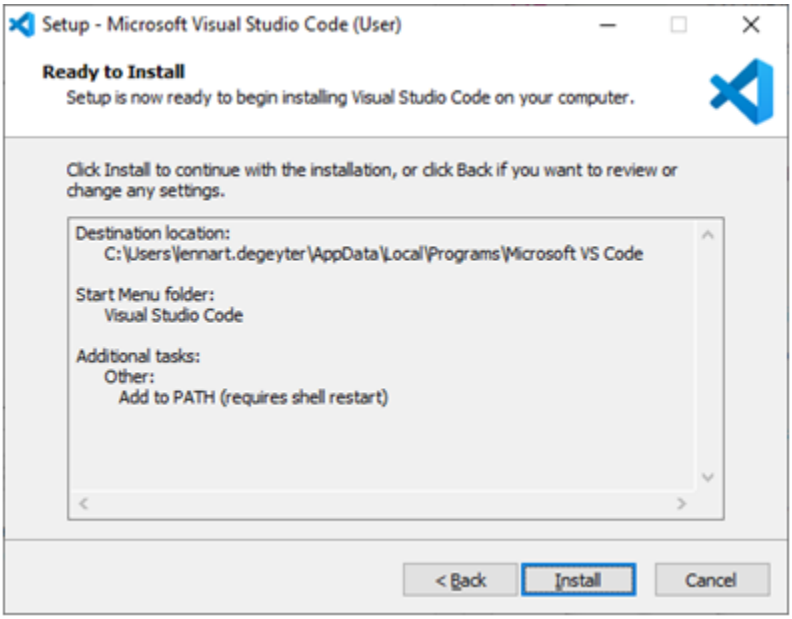
Paso 6: Elige si deseas cambiar el nombre de la carpeta de accesos directos en el menú Inicio o si no deseas instalar accesos directos en absoluto. Haz clic en Next.



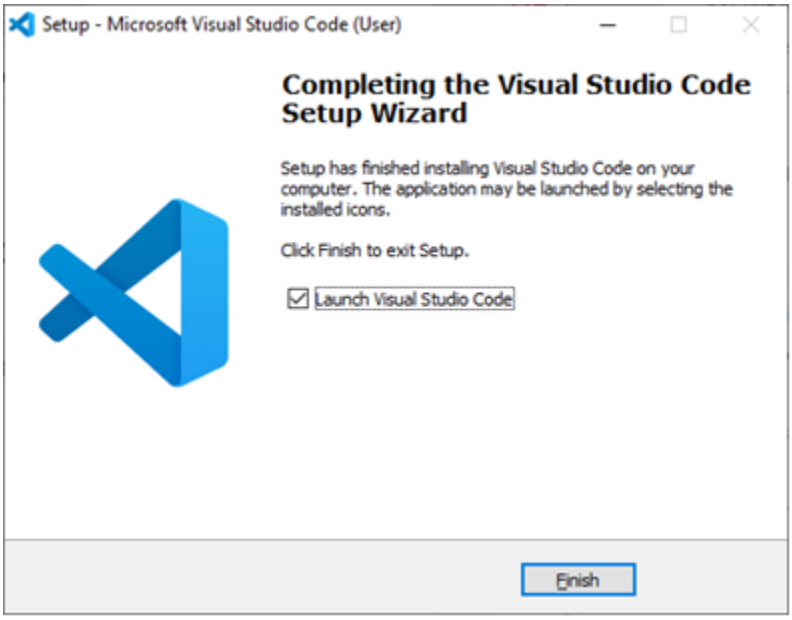
Paso 7: Selecciona las tareas adicionales, por ej. crear un icono en el escritorio o añadir opciones al menú contextual de Windows Explorer. Haz clic en Next.



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

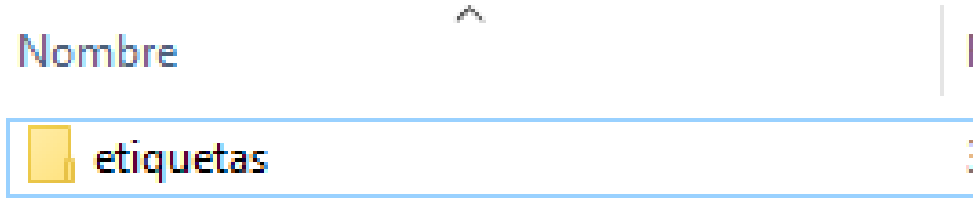


Paso 9: El programa está instalado y listo para usar. Haz clic en Finish para finalizar la instalación y lanzar el programa.

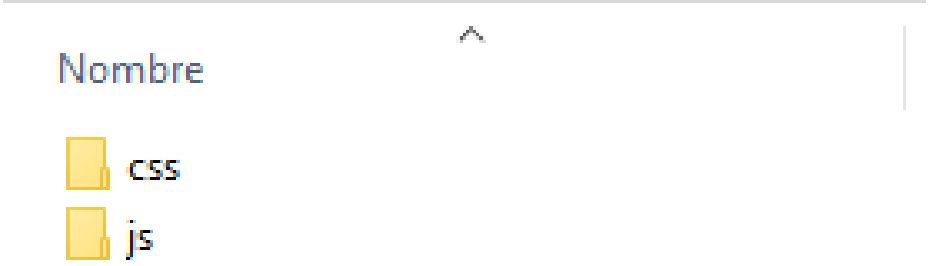


Creación de archivos y carpetas

Paso 1: Creamos una carpeta raíz. En nuestro caso la llamaremos “etiquetas”.

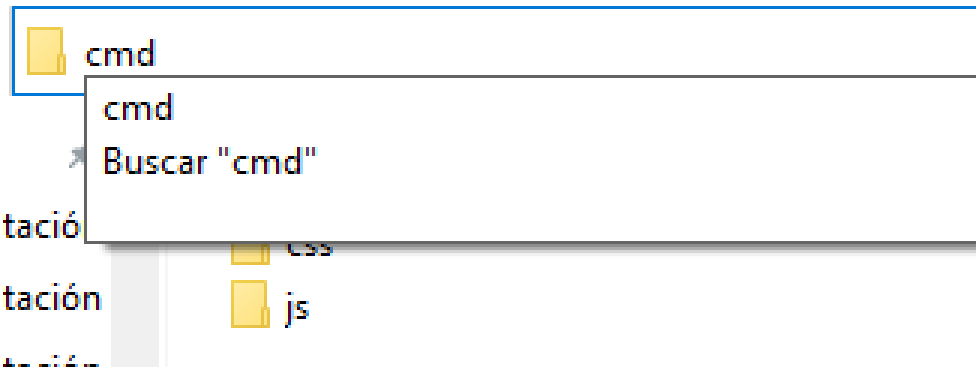


Paso 2: Dentro de esta, crearemos otras dos carpetas llamadas “css”, la cual contendrá todo lo estético de nuestra página, y otra llamada “js” la cual contendrá la lógica de nuestra página.

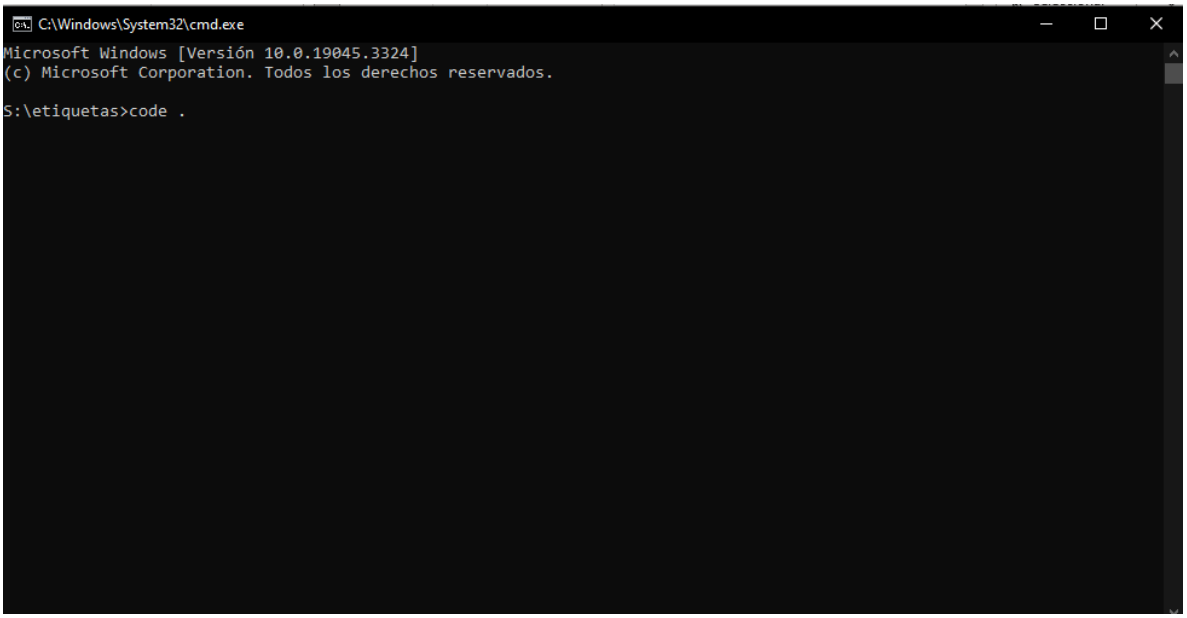


Etiqueta

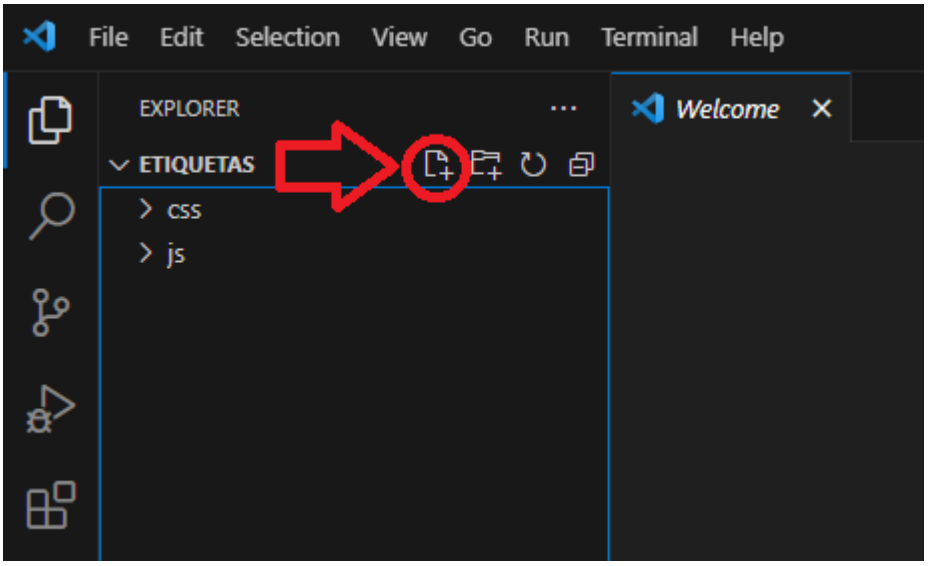
Paso 1: Para abrir el Visual Studio Code, haremos lo siguiente: Dentro de la carpeta raíz, en la barra superior, escribiremos cmd y le damos enter.



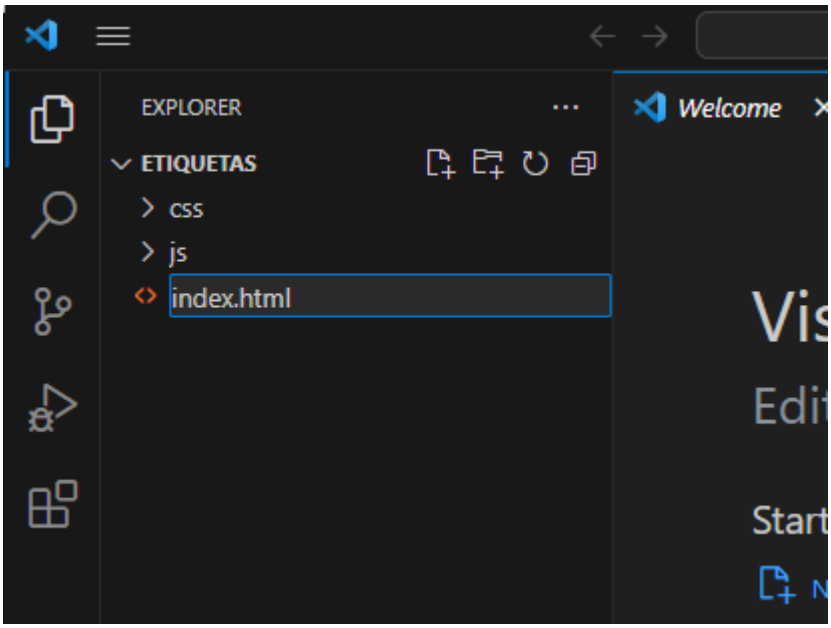
Paso 2: Eso nos abrirá una terminal, solo tendremos que escribir “code .”.



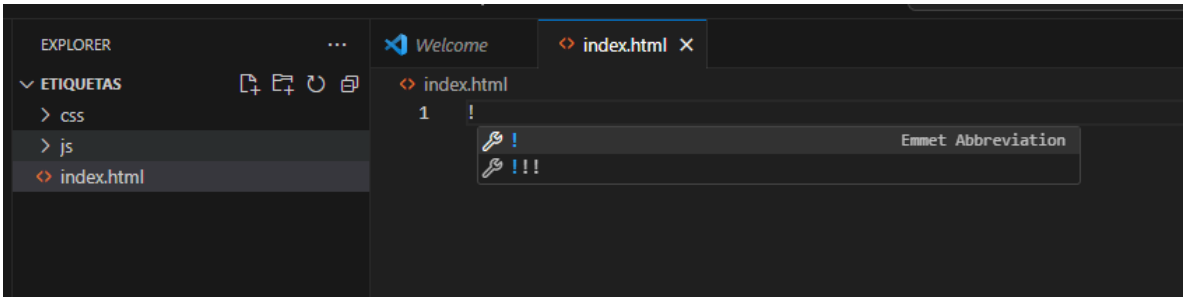
Paso 3: Esto nos abrirá el visual studio code. Ahora, ya que estamos dentro, crearemos un archivo llamado “index.html” dándole en la carpeta raíz dando click donde indica la flecha.



Paso 4: Llamaremos el archivo “index.html”

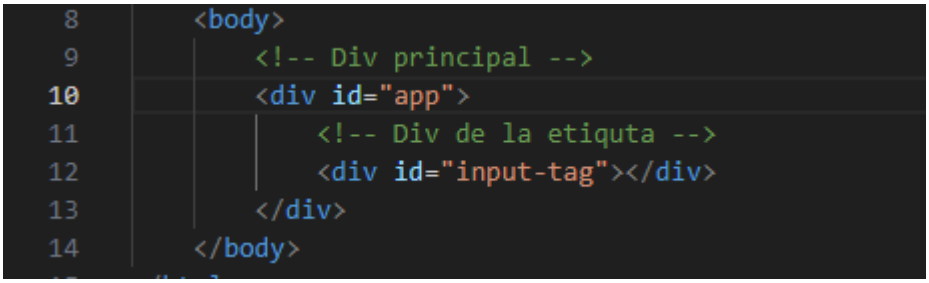


Paso 5: Una vez en el archivo recién creado, colocamos el código “!” y escogemos la primera opción.

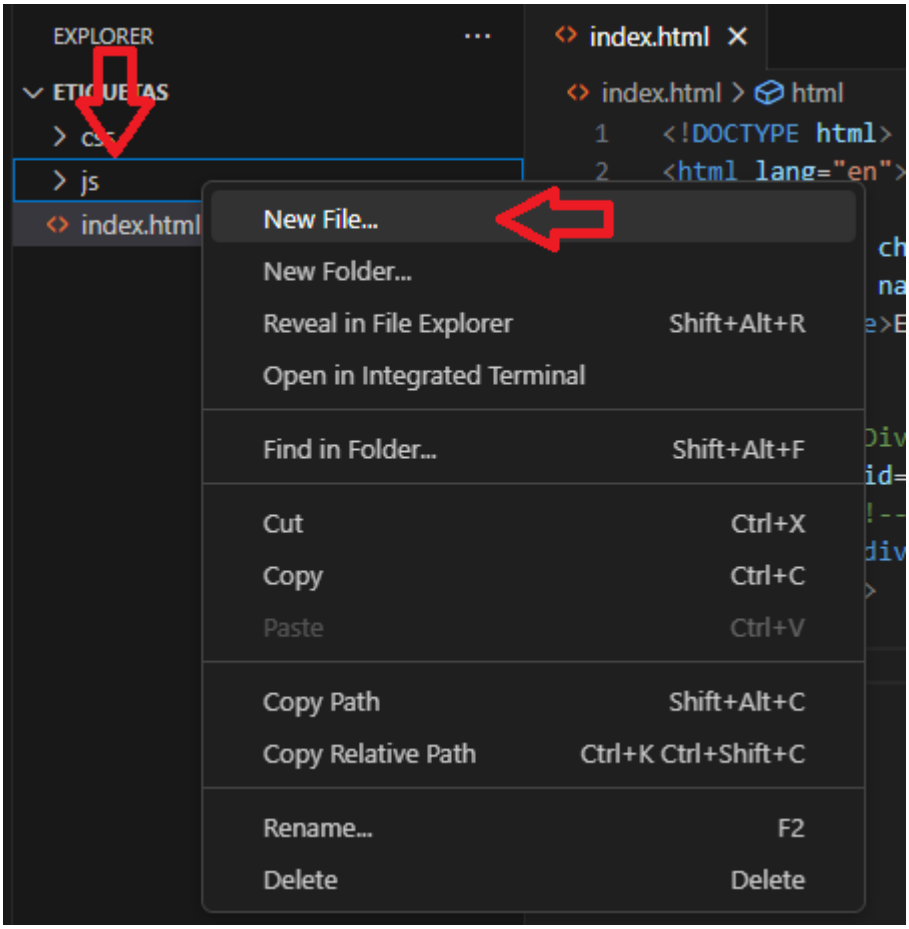


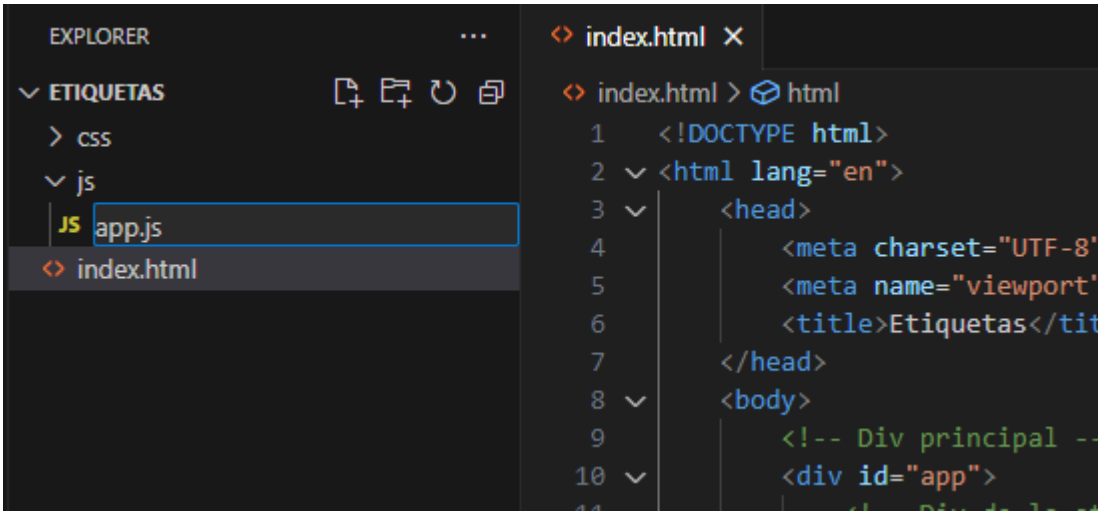
Paso 6: Una vez colocada la estructura, cambiaremos el titulo (línea 6) por “Etiquetas”

Paso 7: Ahora dentro del body crearemos un div principal para nuestra página (Linea 10) le pondremos un id llamado “app”, y además crearemos otro div dentro de ese mismo div (línea 12) y le pondremos el id “input-tag”.



Paso 8: Ahora crearemos un nuevo archivo llamado “app.js” dando click derecho a la carpeta llamada “js” y new file





Paso 9: Una vez creado el archivo vamos a vincular el javascript en nuestro html (Línea 16)

```
10          <div id="app">
11              <!-- Div de la etiquta -->
12              <div id="input-tag"></div>
13          </div>
14
15          <!-- JAVASCRIPT-->
16          <script src="js/app.js"></script>
17      </body>
```

Paso 10: Una vez vinculado el javascript iremos al archivo “app.js”, donde crearemos un array para almacenar las etiquetas creadas por el usuario (línea 2)

Además vamos a llamar el id del html (“input-tag”) que creamos anteriormente (línea 3) además crearemos 2 constantes que se van a encargar de crear un contenedor de etiqueta y de entrada en el html cuando sea necesario (línea 4-5)

index.html | app.js

```
1  // Declaración de variables
2  let tags = []; // Almacenará las etiquetas ingresadas por el usuario
3  const inputTagContainer = document.querySelector("#input-tag"); // Contenedor de entrada de etiquetas
4  const tagsContainer = document.createElement("div"); // Contenedor de etiquetas
5  const inputTag = document.createElement("span"); // Etiqueta de entrada
6
```

Paso 11: Ahora crearemos una función “inputTagContainer” (línea 8) el cual funcionará para los contenedores de las etiquetas, comenzaremos creando un condicional (línea 10), y en caso de que encuentre el id “input-tag” (línea 11) o cualquier elemento secundario de la clase “tag-container” (línea 12), en caso de que se cumpla cualquiera va a enfocar la tarjeta de entrada para editarla (línea 14)

```
6
7   // Manejo del evento click en el contenedor de entrada de etiquetas
8   inputTagContainer.addEventListener("click", (e) => {
9     // Si se hace clic en el contenedor de entrada o en un contenedor de etiquetas
10    if (
11      e.target.id === "input-tag" ||
12      e.target.classList.contains("tag-container")
13    ) {
14      inputTag.focus(); // Enfoca la etiqueta de entrada para editarla
15    }
16  });
17
```

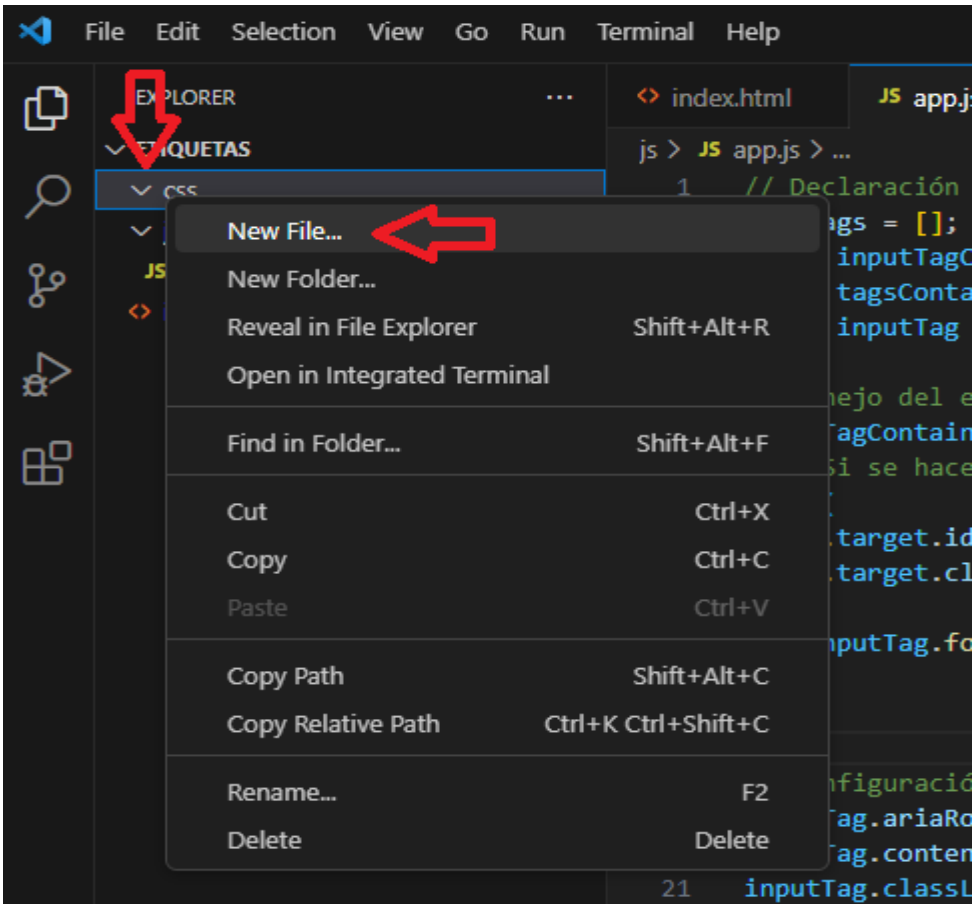
Paso 12: Ahora crearemos un código el cual configura una etiqueta de entrada. Establece su propiedad ariaRoleDescription (línea 19)para mejorar la accesibilidad, permite la edición del contenido con contentEditable(línea 20), agrega la clase CSS "input" (linea21) a la etiqueta y la enfoca automáticamente para que el usuario pueda comenzar a editarla de inmediato (línea 22).

```
17
18  // Configuración de la etiqueta de entrada
19  inputTag.ariaRoleDescription = "textbox"; // Propiedad para accesibilidad
20  inputTag.contentEditable = "true"; // Permite editar el contenido
21  inputTag.classList.add("input"); // Agrega una clase CSS
22  inputTag.focus(); // Enfoca automáticamente la etiqueta de entrada
23
```

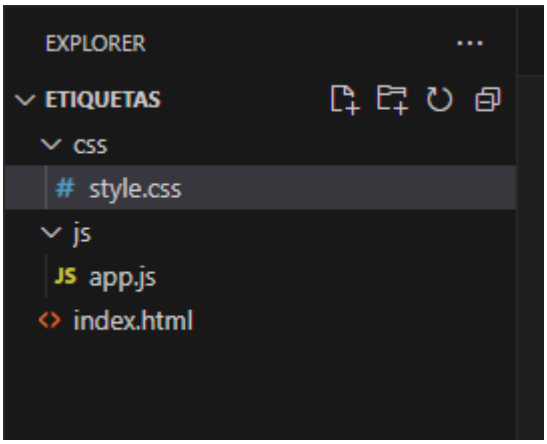
Paso 13: Ahora vamos a añadir clases css a los elementos del HTML (línea 25-26)

```
23
24  // Agrega clases CSS a los contenedores
25  inputTagContainer.classList.add("input-tag-container");
26  tagsContainer.classList.add("tag-container");
27
```

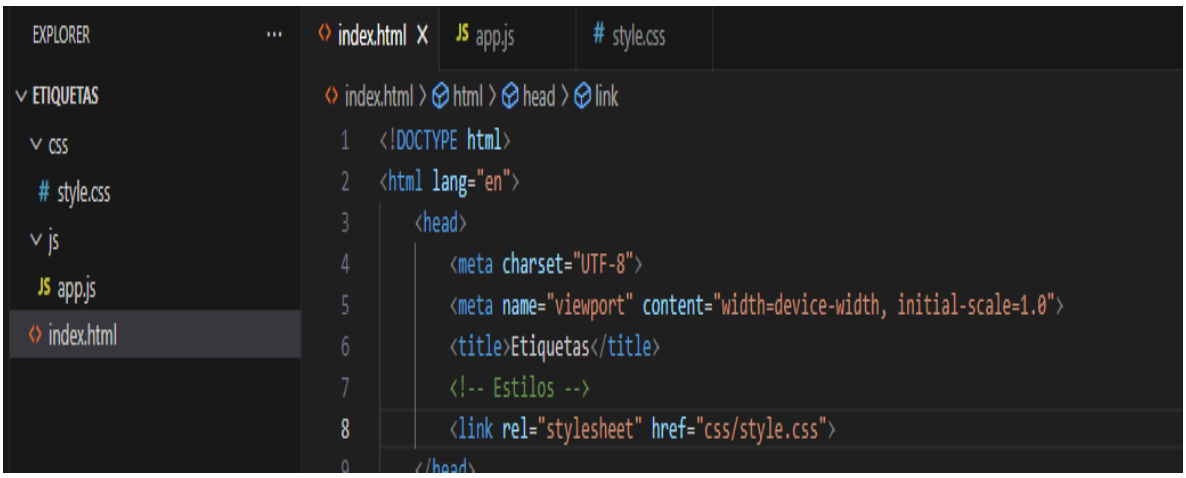
Paso 14: Ahora miraremos el diseño de nuestra página, lo primero que haremos será dar click derecho en la carpeta “css” y darle a new file o nuevo archivo



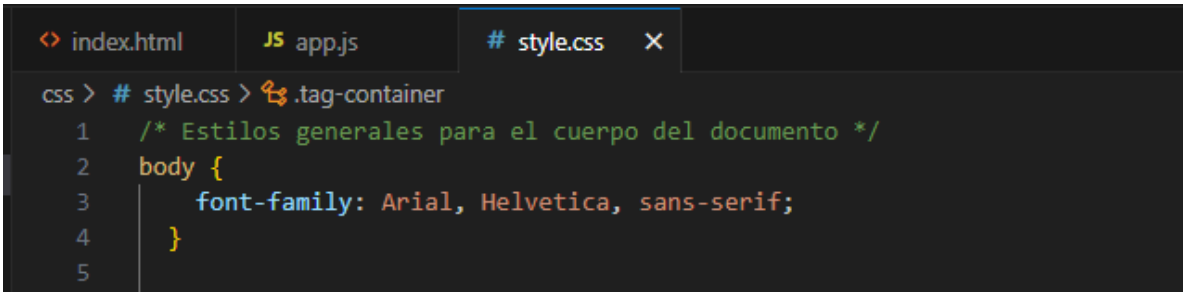
Paso 15: Llamaremos el archivo como “style.css”



Paso 16: Ahora iremos a nuestro archivo “index.html” y vincularemos nuestro css a nuestro html (línea 8)



Paso 17: Iremos a nuestro archivo “style.css” y pondremos unas fuentes (línea 2-4)



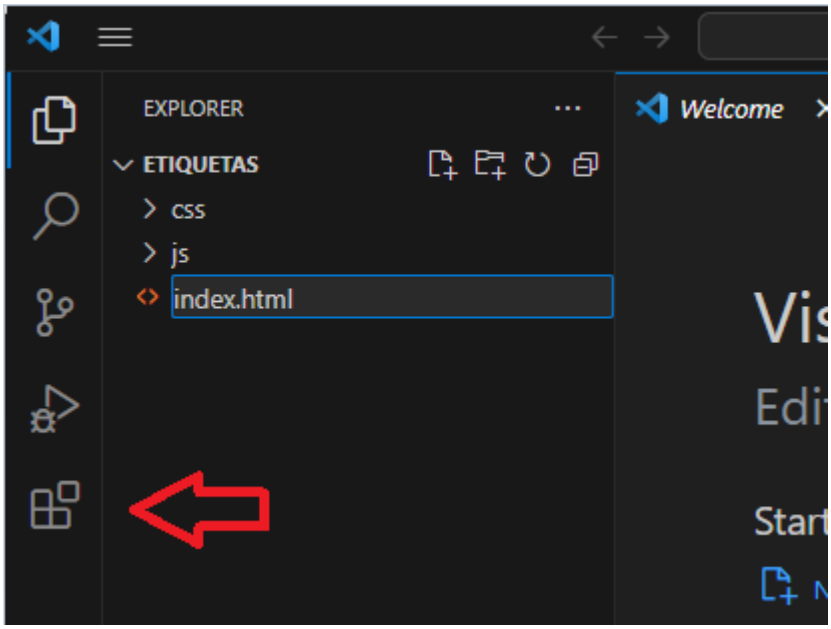
Paso 18: Ahora vamos a crear un diseño para la etiqueta donde escribimos

```
5
6   /* Estilos para la etiqueta de entrada */
7   .input {
8     font-family: inherit; /* Utiliza la fuente del elemento padre */
9     font-size: inherit; /* Utiliza el tamaño de fuente del elemento padre *
10    padding: 10px 6px; /* Espaciado interno vertical y horizontal */
11    min-width: 20px; /* Ancho mínimo del elemento */
12    display: flex; /* Utiliza un modelo de caja flexible */
13    align-items: center; /* Alinea verticalmente el contenido */
14    outline: none; /* Quita el resaltado al enfocar el elemento */
15  }
```

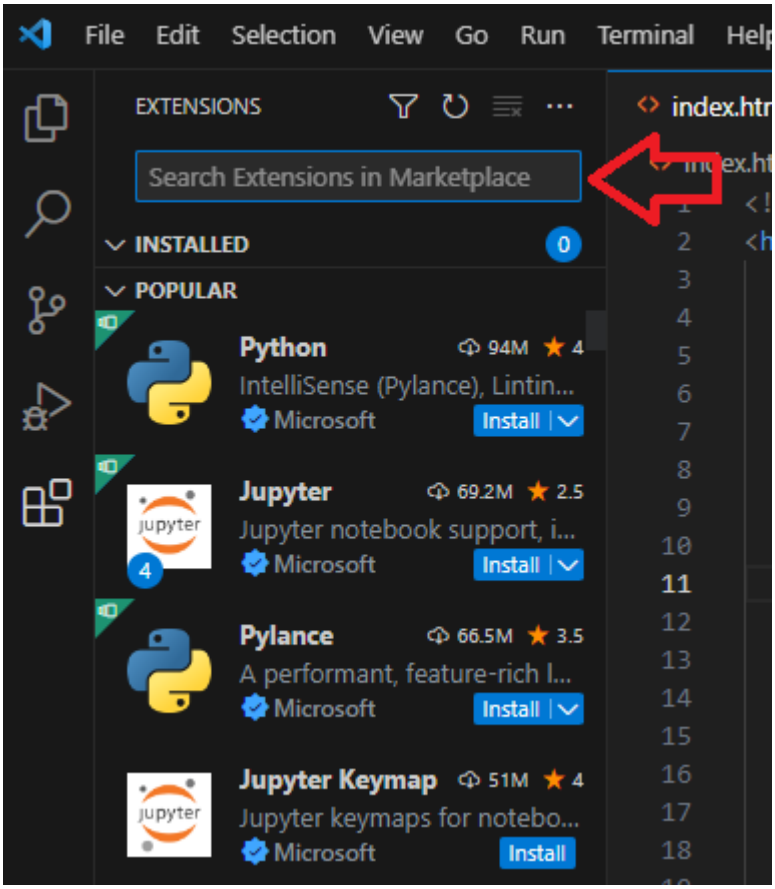
Paso 19: Ahora vamos a modificar el diseño de nuestro contenedor de etiquetas.

```
17  /* Estilos para el contenedor de entrada de etiquetas */
18  .input-tag-container {
19    display: flex; /* Utiliza un modelo de caja flexible */
20    padding: 5px; /* Espaciado interno del contenedor */
21    width: 500px; /* Ancho máximo del contenedor */
22    flex-wrap: wrap; /* Permite que las etiquetas se ajusten a una nueva línea si no hay espacio */
23    border: solid 1px #ccc; /* Borde del contenedor */
24  }
```

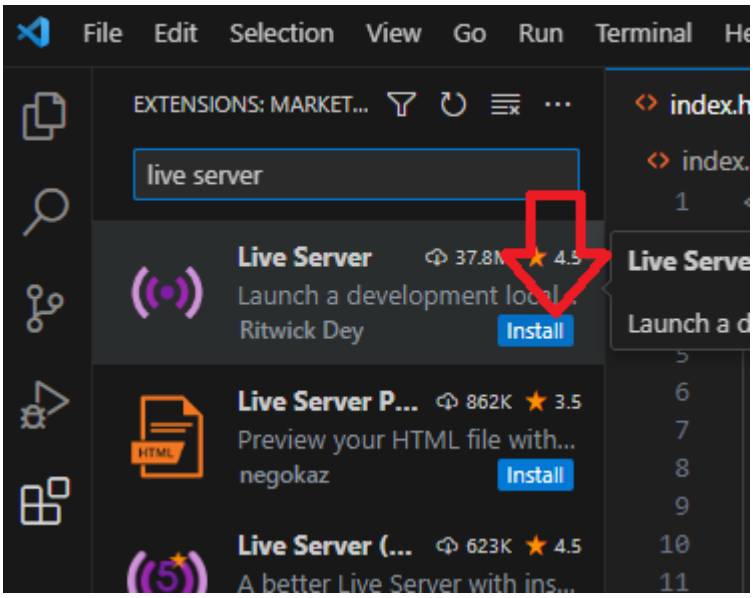
Paso 20: Ahora para visualizar nuestro proyecto daremos click a la zona inferior izquierda como apunta la flecha.



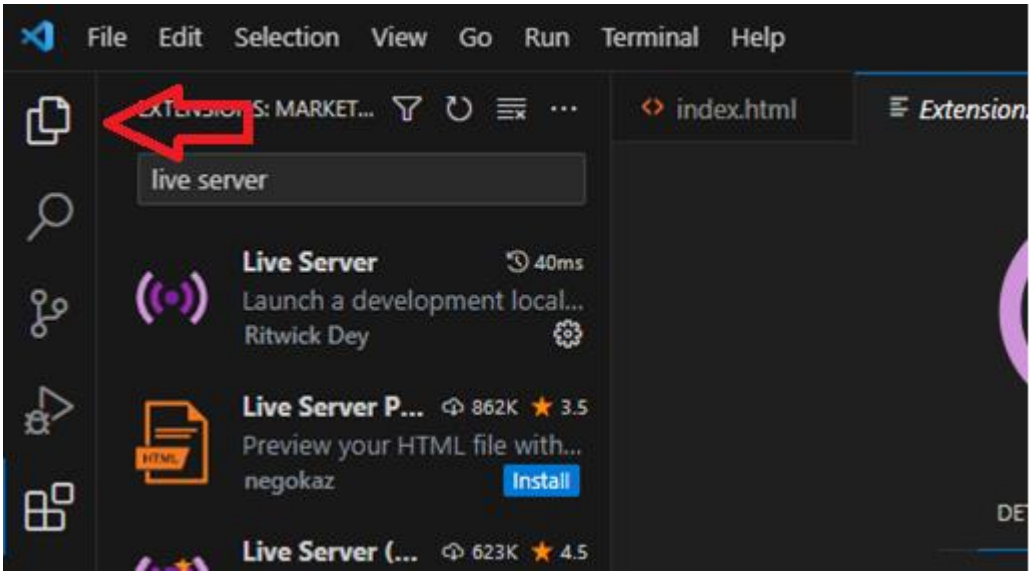
Paso 21: Nos saldrá esta pestaña y buscaremos live server, en la barra de búsqueda.



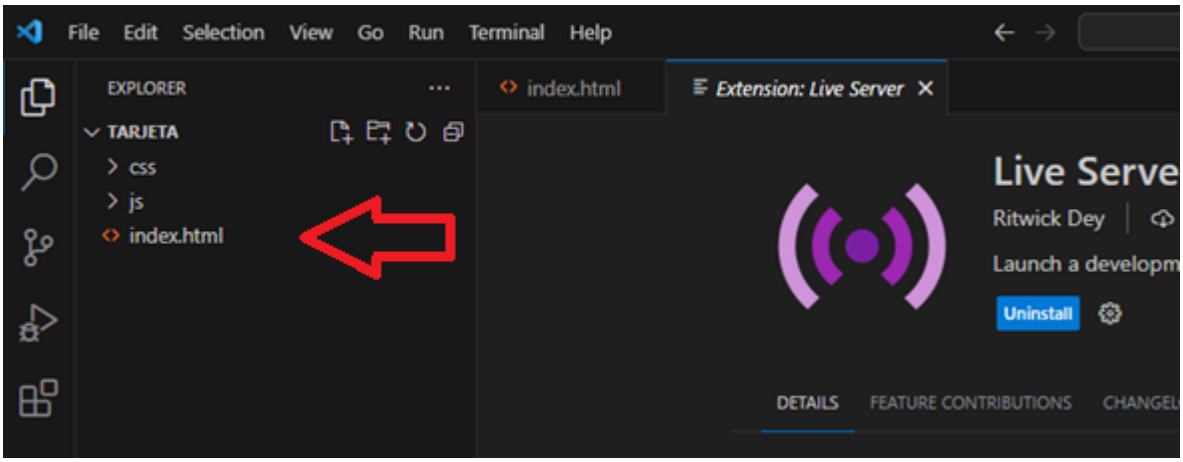
Paso 22: Ahora que encontramos lo que necesitamos, le damos click a “install”



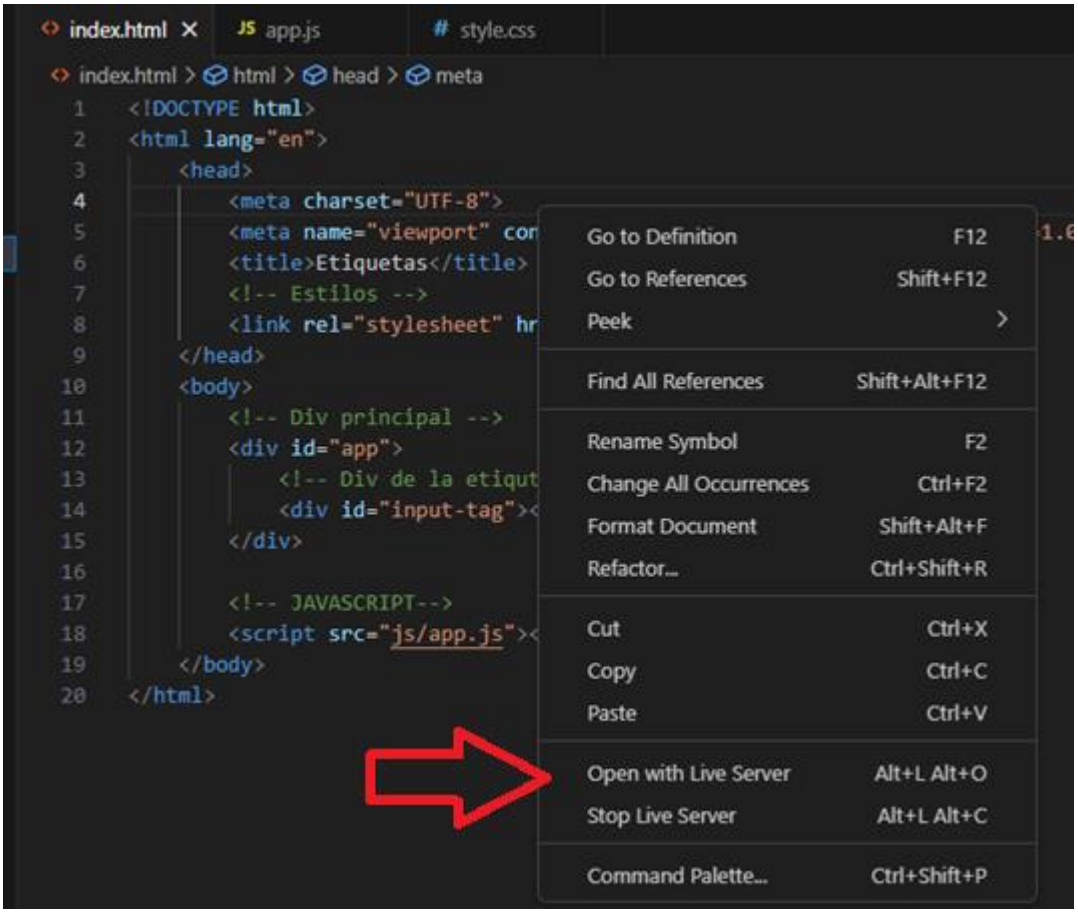
Paso 23: Ahora que tenemos instalado el live server nos debería de aparecer algo así donde podremos volver a nuestro proyecto dando click al explorador.



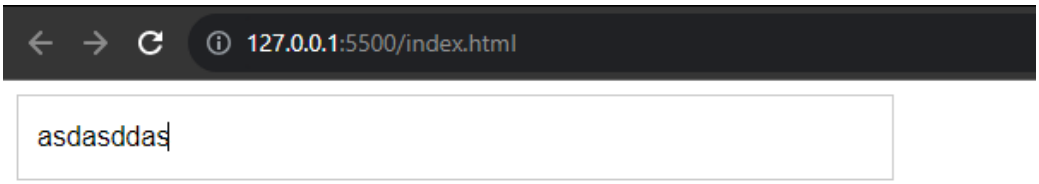
Paso 24: Una vez te sale el explorador volvemos al Index.html



Paso 25: Una vez volvemos al index.html damos click derecho en cualquier parte del código y damos click izquierdo a “open with live server”



Paso 26: Ahora nuestra página estará así



Paso 27: Ahora volveremos a nuestro archivo “app.js” y vamos a añadir un evento (línea 33) y una condicional de que en caso de que se presione la tecla “enter” y el contenido no sea igual a vacio va a quitar el comportamiento por defecto (línea 37) y vamos a agregar la etiqueta en la lista (línea 39) también limpiaremos la lista para que no se esté generando constantemente (línea 40) y vamos a llamar la función “renderTags” que aun no esta creada pero sirve para actualizar la visión y se vea la etiqueta que recién creamos (línea 41).

```
32  // Manejo del evento keydown en la etiqueta de entrada
33  inputTag.addEventListener("keydown", (e) => {
34
35    // Si se presiona la tecla Enter y el contenido de la etiqueta de entrada no está vacío
36    if (e.key === "Enter" && inputTag.textContent !== "") {
37      e.preventDefault(); // Evita el comportamiento por defecto del Enter
38      // Si la etiqueta no existe aún, la agrega a la lista de etiquetas
39        tags.push(inputTag.textContent); // Agrega la etiqueta a la lista
40        inputTag.textContent = ""; // Limpia el contenido de la etiqueta de entrada para la siguiente
41        renderTags(); // Actualiza la visualización de las etiquetas
42      }
43  });
```

Paso 28: Ahora vamos a crear la función “renderTags” línea (línea 46) y comenzaremos limpiando el contenido del contenedor (línea 47) y vamos a recorrer el array “tags” con la función “map” (línea 48) y vamos a crear un elemento de etiqueta que será un div (línea 49) y un botón (línea 50) a esos elementos le agregaremos la clase (tag-item), (línea 51) y al botón le pondremos de contenido una “X” (línea 52) y por añadiremos un evento para que cuando pulse el botón que acabamos de crear se elimine la etiqueta (línea 53 a 54).

```
45  // Función para renderizar las etiquetas en el contenedor de etiquetas
46  function renderTags() {
47    tagsContainer.innerHTML = ""; // Limpia el contenido del contenedor de etiquetas
48    const html = tags.map((tag) => {
49      const tagElement = document.createElement("div"); // Crea un elemento de etiqueta
50      const tagButton = document.createElement("button"); // Crea un botón para eliminar la etiqueta
51      tagElement.classList.add("tag-item"); // Agrega una clase CSS al elemento de etiqueta
52      tagButton.textContent = "X"; // Agrega contenido al botón
53      tagButton.addEventListener("click", (e) => {
54        removeTag(tag); // Maneja el evento click en el botón para eliminar la etiqueta
55      });
```

Paso 29: Ahora vamos a añadir el contenido de la etiqueta al elemento div que hermos creado anteriormente con el texto de la etiqueta (línea 57), y además vamos a añadir la “X” (línea 58)

```
55    });
56
57    tagElement.appendChild(document.createTextNode(tag)); // Agrega el contenido de la etiqueta al elemento
58    tagElement.appendChild(tagButton); // Agrega el botón de eliminación al elemento
59    return tagElement;
60  });
61
```

Paso 30: Ahora dentro de la misma función “renderTags” vamos a hacer que por cada elemento añada un elemento de etiqueta (línea 63-64), también agregaremos la etiqueta “inputTag” a todas las etiquetas generadas, para poder agregar nuevas etiquetas luego (línea 66) y por último colocamos para que el usuario pueda añadir otra etiqueta sin siquiera clicar (línea 67)

```
60    });
61
62    // Agrega cada elemento de etiqueta renderizado al contenedor de etiquetas
63    html.forEach((element) => {
64      tagsContainer.appendChild(element);
65    });
66    tagsContainer.appendChild(inputTag); // Agrega la etiqueta de entrada al final
67    inputTag.focus(); // Enfoca la etiqueta de entrada para la siguiente edición
68  }
```

Paso 31: Ahora mismo nuestro proyecto y veremos que podemos añadir etiquetas sin embargo aún se pueden repetir las etiquetas.

Paso 32: Ahora volvemos a nuestro “app.js” y crearemos una función llamada “existTag” que se encargara de ver si ya existe (linea 70) una etiqueta mirando en caso de que el valor ya este incluido (linea 71)

```
68
69  // Función para verificar si una etiqueta ya existe en la lista
70  function existTag(value) {
71    return tags.includes(value);
72  }
73
```

Paso 33: Ahora vamos a añadir la función “removeTag” para eliminar la etiqueta y actualizamos la visualización (línea 75), filtramos las que no se van a eliminar (línea 76) y renderizamos (línea 77)

```
73
74  // Función para eliminar una etiqueta de la lista y actualizar la visualización
75  function removeTag(value) {
76    tags = tags.filter((tag) => tag !== value); // Filtra las etiquetas que no sean la que se va a eliminar
77    renderTags(); // Actualiza la visualización de las etiquetas
78  }
79
```

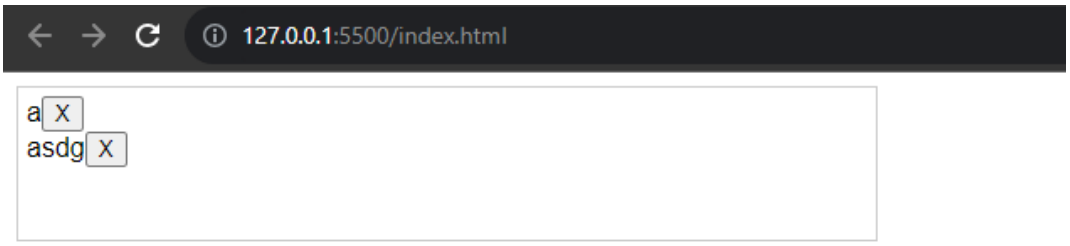
Paso 34: Ahora iremos de nuevo de nuevo al evento en caso de apretar “Enter” (Paso 27) y colocaremos un condicional que en caso de que la etiqueta no exista la coloque (línea 39)

```
32  // Manejo del evento keydown en la etiqueta de entrada
33  inputTag.addEventListener("keydown", (e) => {
34
35      // Si se presiona la tecla Enter y el contenido de la etiqueta de entrada no está vacío
36      if (e.key === "Enter" && inputTag.textContent !== "") {
37        e.preventDefault(); // Evita el comportamiento por defecto del Enter
38        // Si la etiqueta no existe aún, la agrega a la lista de etiquetas
39        if (!existTag(inputTag.textContent)) {
40          tags.push(inputTag.textContent); // Agrega la etiqueta a la lista
41          inputTag.textContent = ""; // Limpia el contenido de la etiqueta de entrada para la siguiente
42          renderTags(); // Actualiza la visualización de las etiquetas
43        }
```

Paso 35: Y ahora en caso de que se aprete la tecla “backspace” (línea 45) y no tenga contenido dentro (línea 46) y que haya mínimo 1 etiqueta (línea 47), va a eliminar la etiqueta (línea 50) y va a llamar la función “renderTags” (línea 51)

```
42          renderTags(); // Actualiza la visualización de las etiquetas
43        }
44      } else if (
45        e.key === "Backspace" &&
46        inputTag.textContent === "" &&
47        tags.length > 0
48      ) {
49        // Si se presiona la tecla Backspace, la etiqueta de entrada está vacía y hay etiquetas en la lista
50        tags.pop(); // Elimina la última etiqueta de la lista
51        renderTags(); // Actualiza la visualización de las etiquetas
52      }
53    });
```

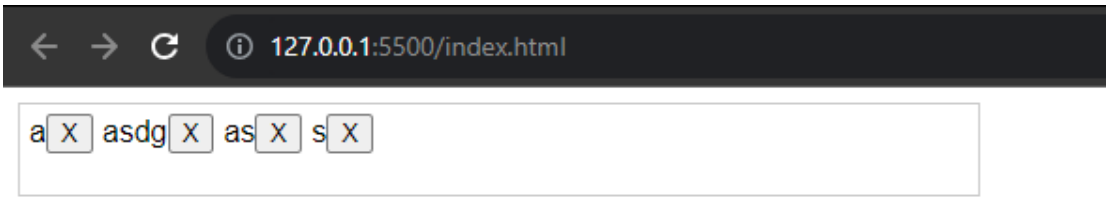
Paso 36: Ahora miramos nuestro proyecto y podemos agregar etiquetas, tampoco podremos colocar 2 etiquetas iguales y podremos eliminar las etiquetas tanto con el botón como con la tecla de retroceso



Paso 37: Ahora iremos a nuestro archivo “style.css” y comenzaremos con el diseño del contenedor.

```
/* Estilos para el contenedor de etiquetas */
.tag-container {
  display: flex; /* Utiliza un modelo de caja flexible */
  gap: 5px; /* Espacio entre las etiquetas */
  flex-wrap: wrap; /* Permite que las etiquetas se ajusten a una nueva línea si no hay espacio */
}
```

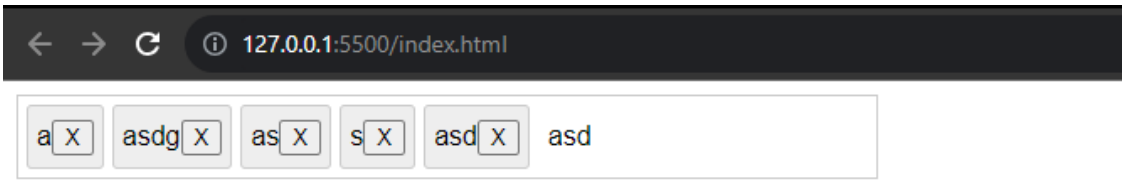
Paso 38: Así va quedando por ahora



Paso 39: Ahora hacemos un poco de diseño para la caja de etiquetas.

```
/* Estilos para cada elemento de etiqueta */
.tag-item {
  border: solid 1px #ccc; /* Borde de cada etiqueta */
  background-color: #eee; /* Color de fondo de las etiquetas */
  display: flex; /* Utiliza un modelo de caja flexible */
  padding: 5px; /* Espaciado interno de cada etiqueta */
  border-radius: 3px; /* Bordes redondeados */
  align-items: center; /* Centra verticalmente el contenido */
}
```

Paso 40: Así va por ahora



Paso 41: Ahora cambiaremos el diseño del botón de eliminar la etiqueta. Y usaremos el ".tag-item button:hover" para cambiar el color cuando pasas el ratón del cursor por encima.

```
42
43  /* Estilos para el botón de eliminar etiqueta */
44  .tag-item button {
45    width: 25px; /* Ancho del botón */
46    height: 25px; /* Altura del botón */
47    border-radius: 50%; /* Crea un círculo con bordes redondeados */
48    padding: 5px; /* Espaciado interno del botón */
49    margin-left: 5px; /* Espacio a la izquierda del botón */
50    border: none; /* Sin borde en el botón */
51    background-color: transparent; /* Fondo transparente */
52    cursor: pointer; /* Cambia el cursor al pasar sobre el botón */
53  }
54
55  /* Estilos al pasar el cursor sobre el botón de eliminar */
56  .tag-item button:hover {
57    background-color: #ccc; /* Cambia el color de fondo al pasar el cursor */
58  }
59
```

Paso 42: Así ha quedado nuestro proyecto.