Git

Entorno visual en visual studio code

INDICE

**Tabla de contenido**

[.Git 3](#_Toc102556121)

[¿Qué es git? 3](#_Toc102556122)

[Flujo de trabajo 3](#_Toc102556123)

[1. **Directorio de trabajo**: 3](#_Toc102556124)

[2. **Staging Area** 3](#_Toc102556125)

[3. **Head:** 3](#_Toc102556126)

[**Directorio de Trabajo:** 4](#_Toc102556127)

[**Area de Trabajo:** 4](#_Toc102556128)

[**Repositorio (Commit):** 4](#_Toc102556129)

[Operaciones Locales 4](#_Toc102556130)

[Repositorios 6](#_Toc102556131)

[Obteniendo un repositorio Git 6](#_Toc102556132)

[Inicializando un repositorio en un directorio existente 6](#_Toc102556133)

[Revisando el Estado de tus Archivos 8](#_Toc102556134)

[Agregar los contenidos de archivos al staging área 10](#_Toc102556135)

[Agregar archivo por archivo al staging área 11](#_Toc102556136)

[**Agregar múltiples archivos al staging área** 15](#_Toc102556137)

[Preparar archivos modificados 17](#_Toc102556138)

[Estado Abreviado 22](#_Toc102556139)

[Ignorar Archivos 23](#_Toc102556140)

[Ingnorar Multiples Archivos 28](#_Toc102556141)

[Negación de un Archivo Ignorado 29](#_Toc102556142)

# **.Git**

# **¿Qué es git?**

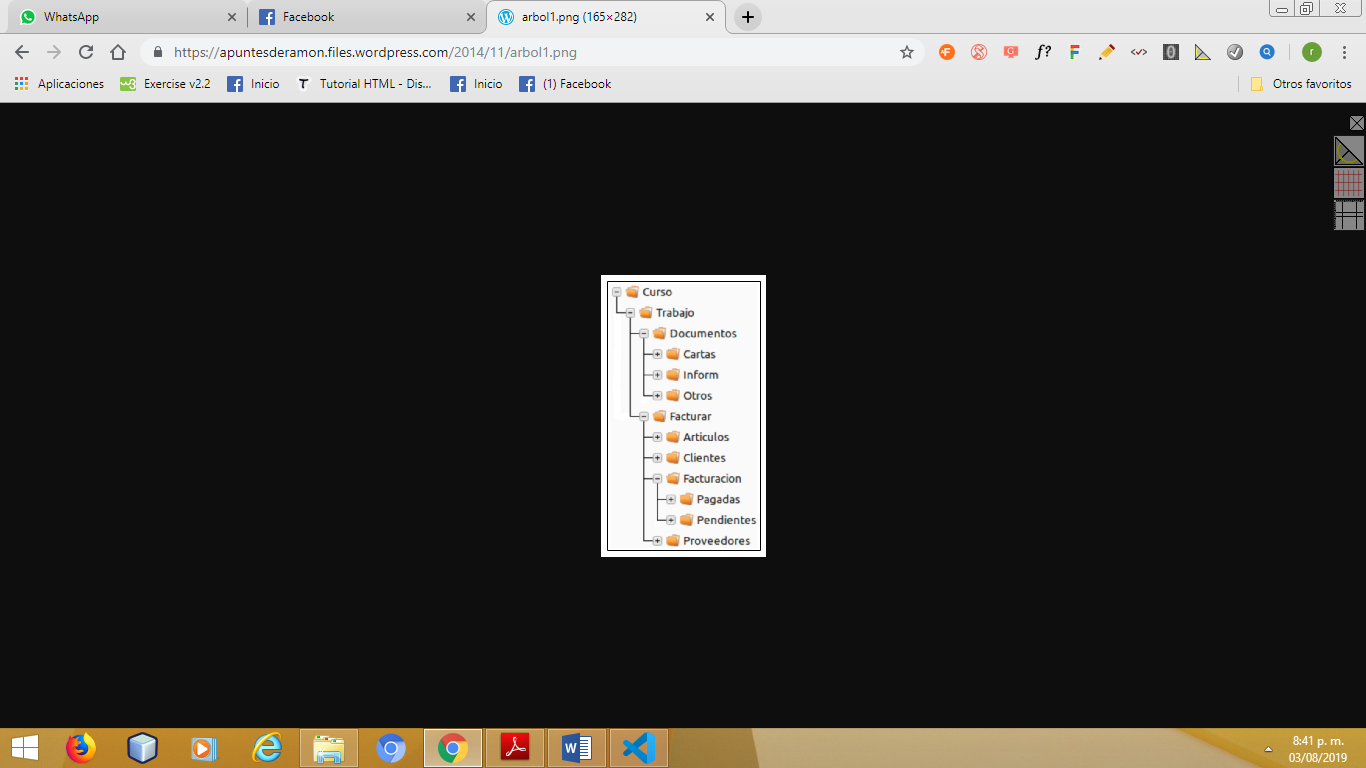
Es un software de control de versiones diseñado por Linux Torval, pensado en la eficiencia y confiabilidad del mantenimiento de versiones cuando estas tienen un gran número de archivos de código fuentes.

## **Flujo de trabajo**

Tus repositorios locales están compuestos por 3 árboles administrados de git.

1. **Directorio de trabajo**: contiene los archivos locales, aquí es donde podemos hacer cualquier cambio y no afectar el repositorio.
2. **Staging Area**: actúa como forma intermedia, es un área de ensayo donde se almacenarán los datos que han sido modificados y posteriormente agregados a esta área con la instrucción add . a esta área de trabajo. También se denomina índex.
3. **Head:**que apunta al último commit realizado.

**Para describir el flujo de trabajo utilizare la siguiente representación**



commi

add

**Repositorio Local**

**Area de trabajo**

**Directorio de trabajo**

### **Directorio de Trabajo:**

Son todos los archivos y carpetas que se encuentran en nuestro proyecto de manera local. Aquí se puede hacer cualquier cambio y no afectara nuestro repositorio.

### **Area de Trabajo:**

Como ya indicamos anteriormente es un espacio donde se almacenarán los datos que han sido modificados. Una vez hechos necesarios, pasamos nuestros archivos a esta área de trabajo **“Static Area”** utilizando la instrucción add

### **Repositorio (Commit):**

Es aquí donde le podemos dar nombre a nuestra versión de toda nuestra **Área de Trabajo.** Es decir, crea una copia de cómo quedara nuestro Repositorio en Producción.

# 

# **Operaciones Locales**

**Directorio git**

**REPOSITORIO LOCAL**

AREA DE TRABAJO

DIRECTORIO DE TRABAJO

Git checkout

Git commit

Git add

Dicho de otro modo, el flujo de trabajo básico de git es algo así:

1. Modifica una serie de archivos en tu directorio de trabajo.
2. Prepara los archivos, añadiendo a tu área de preparación.
3. Confirma los cambios, lo que toma los archivos tal y como están en el área de preparación y almacena esa copia instantánea de manera permanente en tu directorio local

DIRECTORIO DE TRABAJO

AREA DE TRABAJO

Directorio git

REPOSITORIO LOCAL

MODIFICADO

CONFIRMADO

commit

PREPARADO

add

Este proceso de paso por las diferentes áreas, es un ciclo completamente iterativo, que se va produciendo una y otra vez durante la vida del desarrollo de tu proyecto.

# **Repositorios**

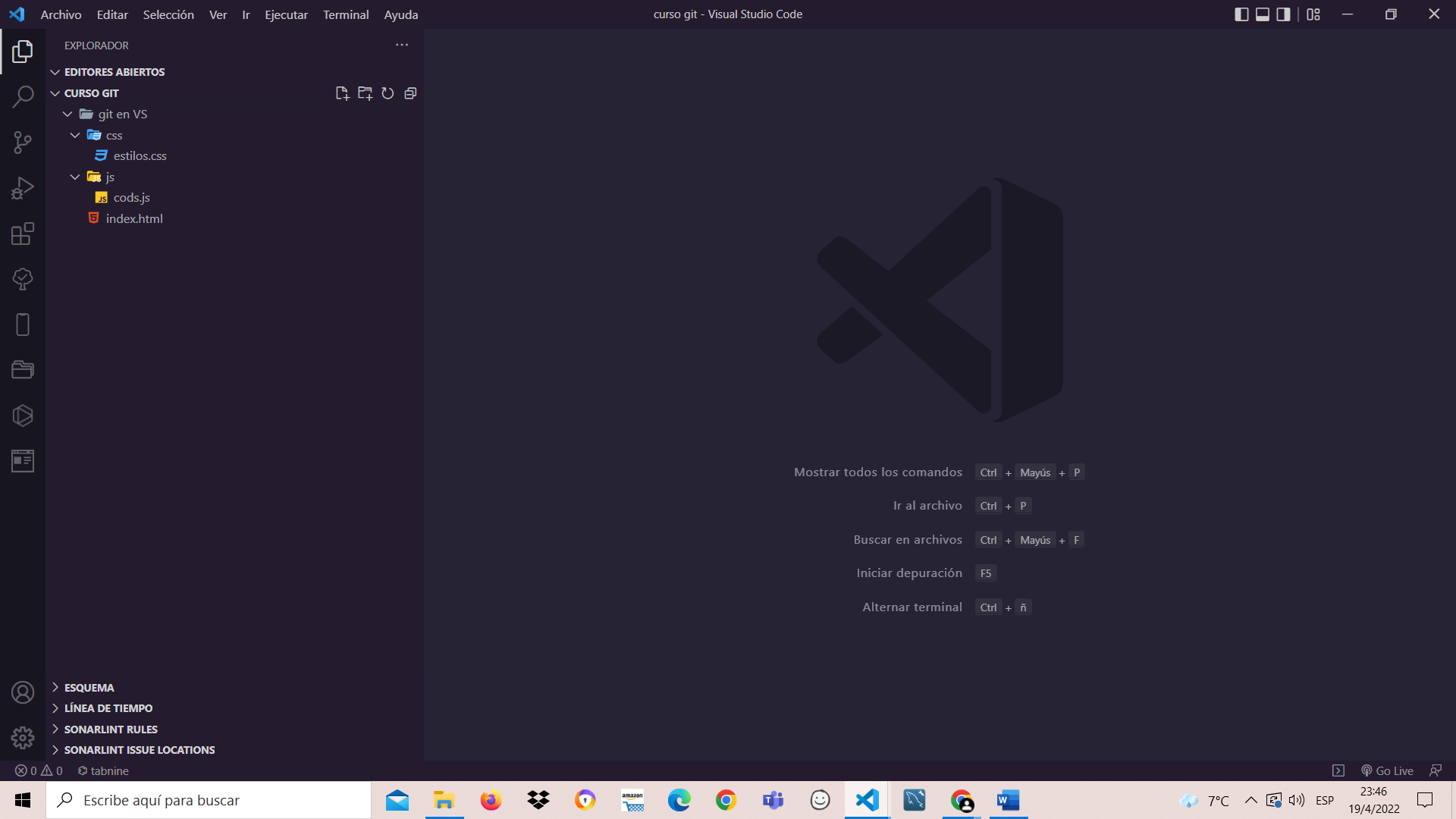
Un repositorio es el elemento base en Git, y representa la biblioteca donde se almacenan todas las versiones de los archivos de una aplicación o proyecto. Una app, una página web, serían ejemplos de grupos de archivos que podrían ubicarse dentro de un repositorio. En el caso de las aplicaciones de gran volumen, normalmente éstas se encuentran distribuidas en múltiples repositorios.

## **Obteniendo un repositorio Git**

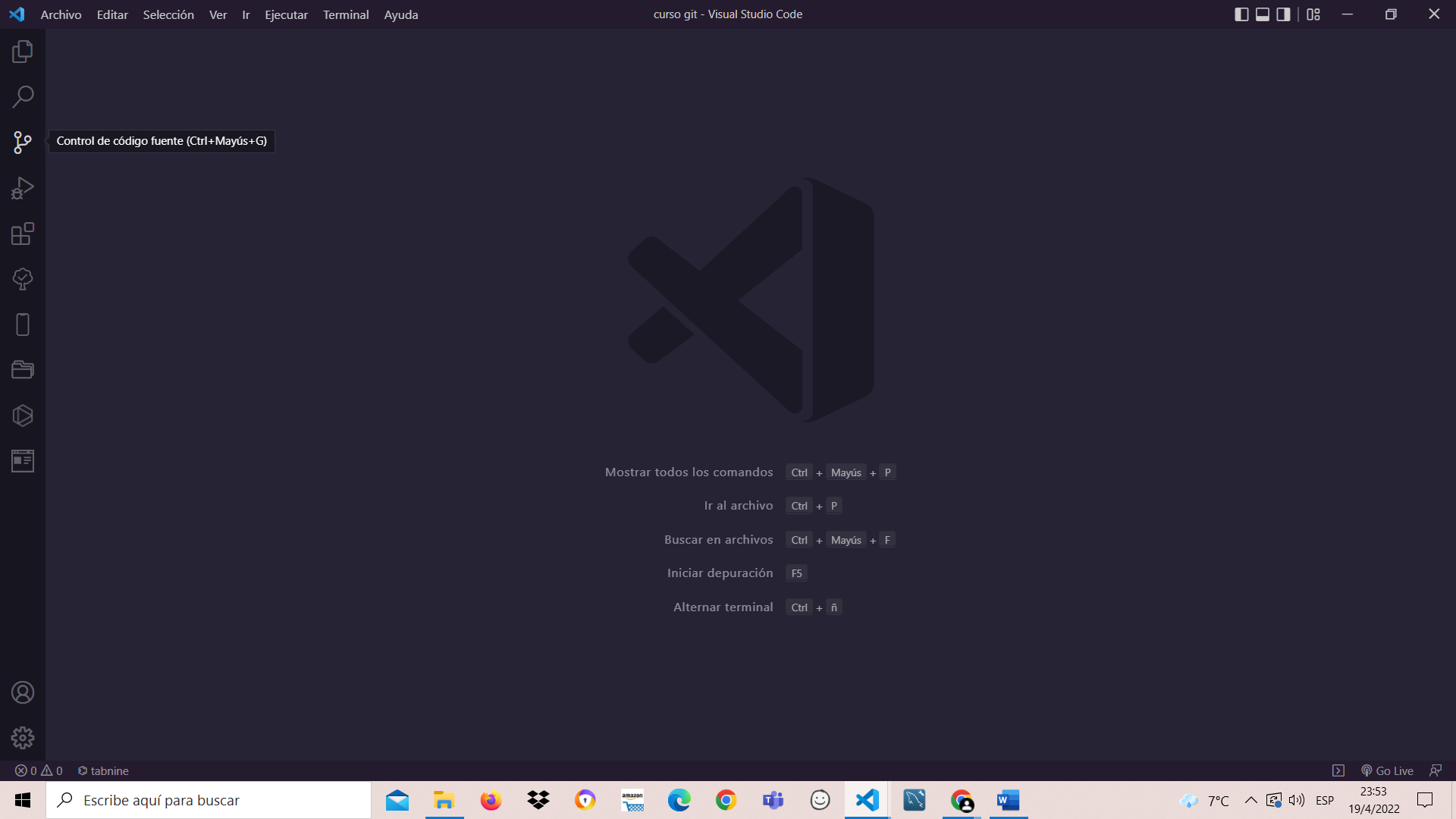
Puedes obtener un proyecto Git de dos maneras. La primera es tomar un proyecto o directorio existente e importarlo en Git. La segunda es clonar un repositorio existente en Git desde otro servidor.

## **Inicializando un repositorio en un directorio existente**

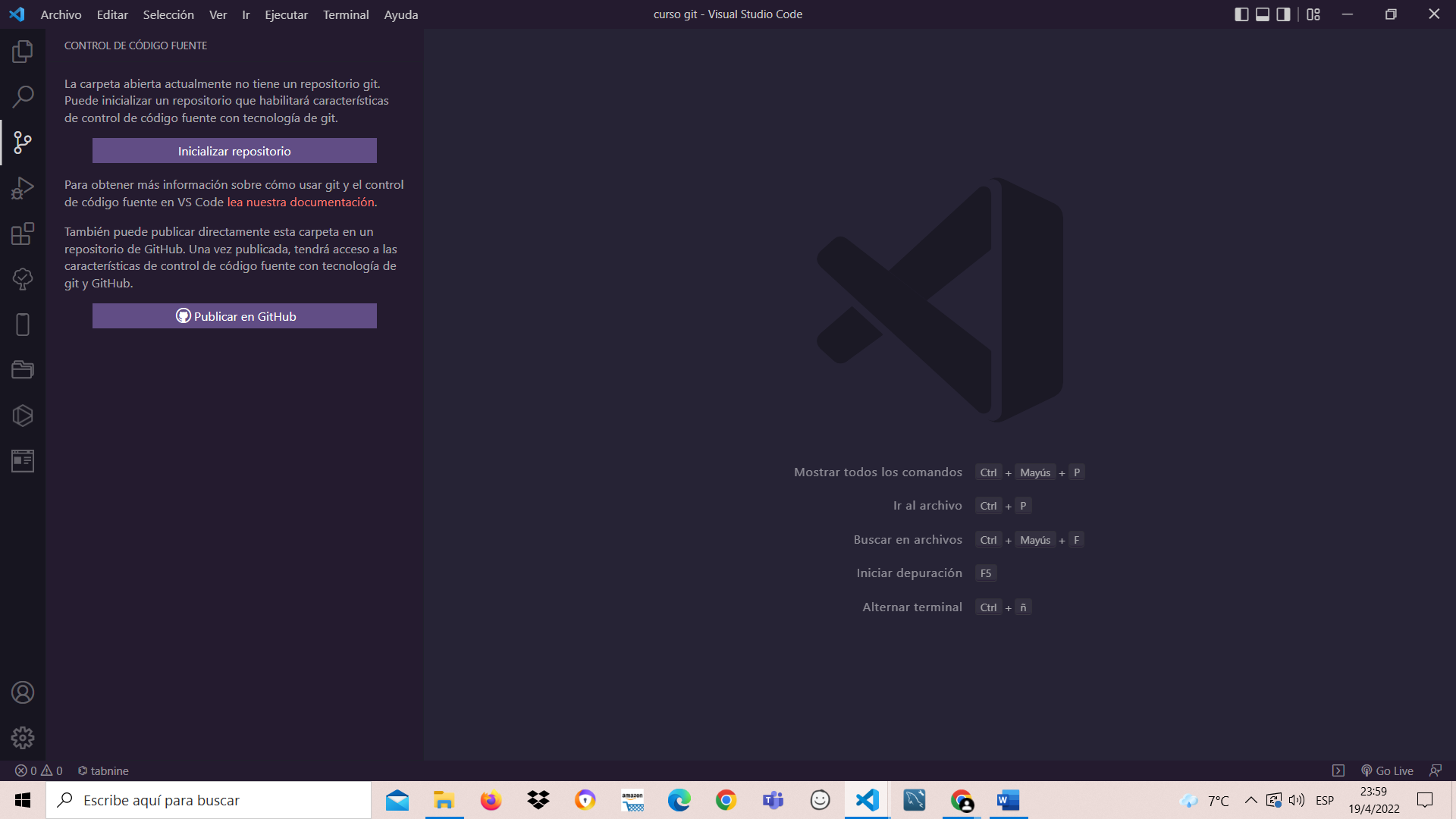
Asumiremos que para comenzar y poner en seguimiento nuestro proyecto tenemos la siguiente estructura



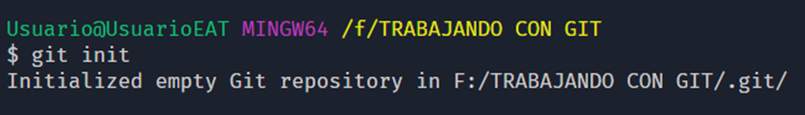
Ahora necesitamos que git inicialice nuestro repositorio para poner en seguimiento nuestro proyecto, esto lo podemos hacer buscando la opción control de código fuente que se encuentra en el panel lateral izquierdo de nuestro editor



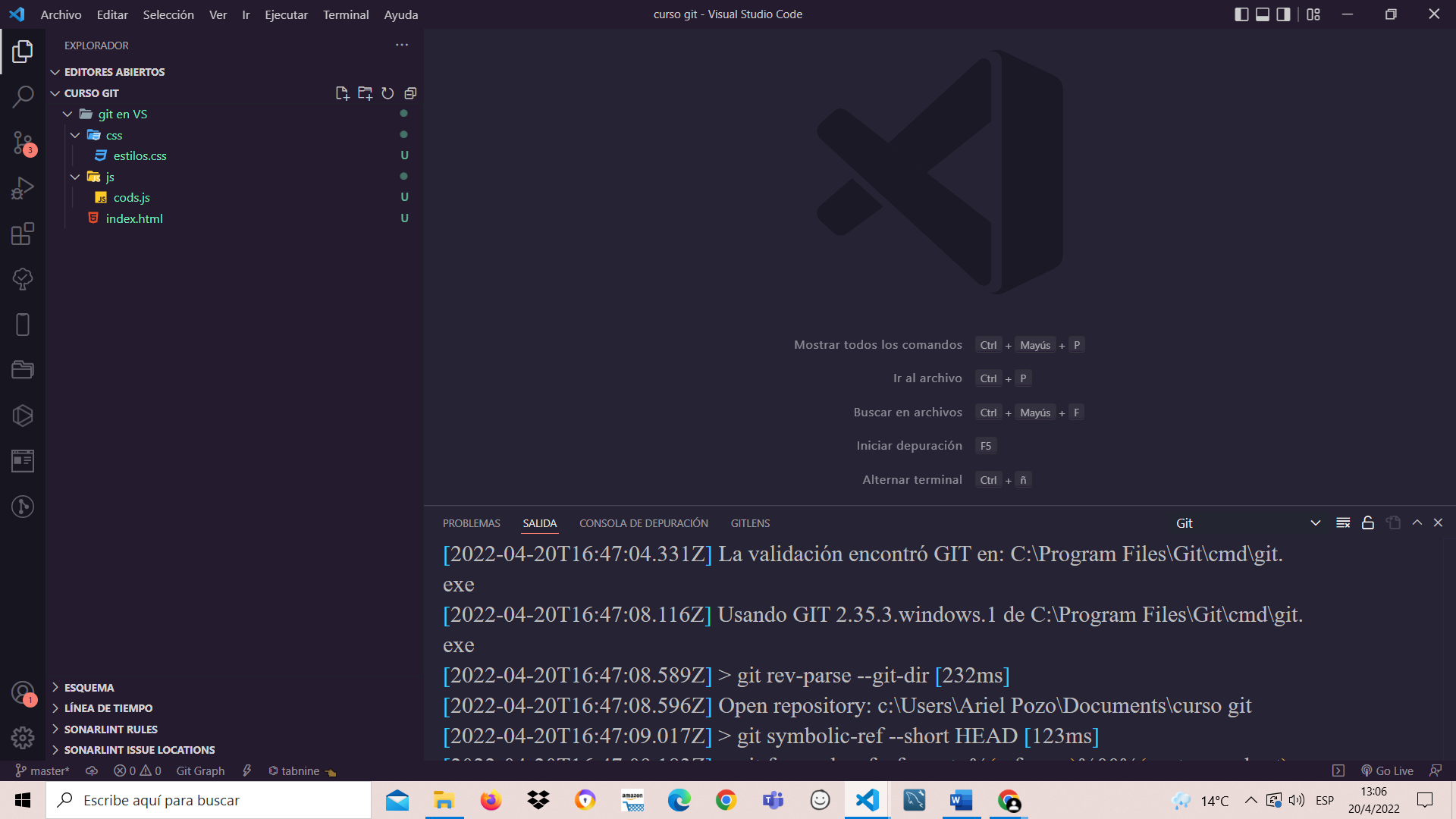
A continuación, nos dará la opción para poner en marcha el seguimiento de nuestro proyecto



Hacer click aqui

Esto es equivalente a haber ejecutado el comando 

Al hacer click en inicializar repositorio nos mostrara que el proyecto ya ha iniciado git de la siguiente manera



Archivo 3

Archivo 1

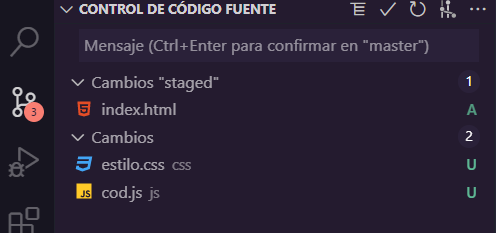
Archivo 2

Numero de archivos sin rastrear

Como se puede observar el icono de git en el panel lateral izquierdo nos indica que hay 3 archivos sin rastrear se puede observar también que en la parte lateral derecha de los archivos nos aparece una letra U indicándonos que esos son precisamente los archivos que están sin seguimiento y no están en el área de trabajo.

# **Revisando el Estado de tus Archivos**

El espacio principal para determinar qué archivos están en qué estado lo determina la siguiente imagen



Area de trabajo

Directorio de trabajo

La imagen anterior nos muestra en qué estado se encuentra actualmente nuestro archivo esto es equivalente a ejecutar el comando para que quede un poco más claro lo plasmaremos en un diagrama

**Directorio git**

**REPOSITORIO LOCAL**

AREA DE TRABAJO O **Static Area**

DIRECTORIO DE TRABAJO





Git add

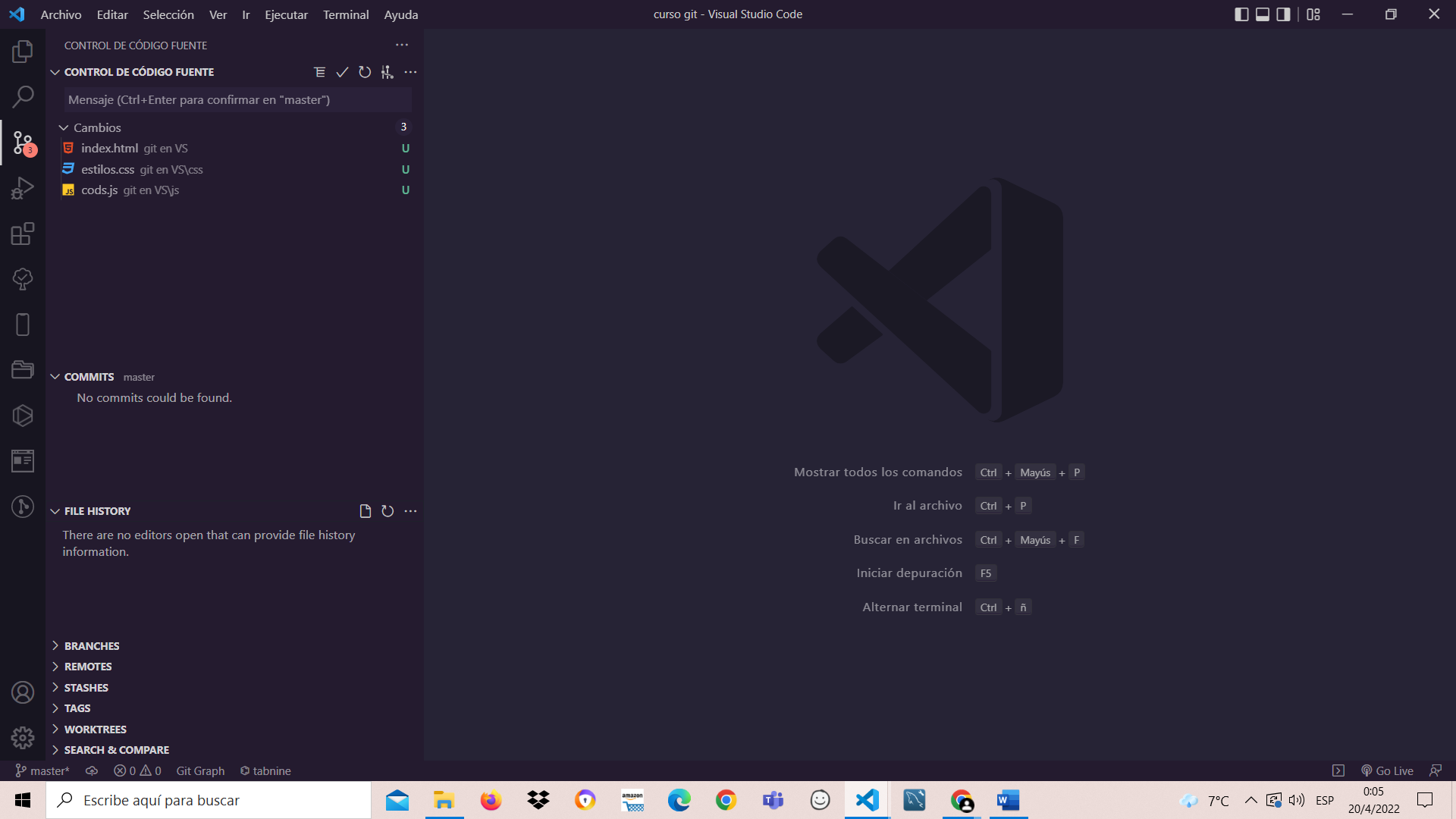
Git commit

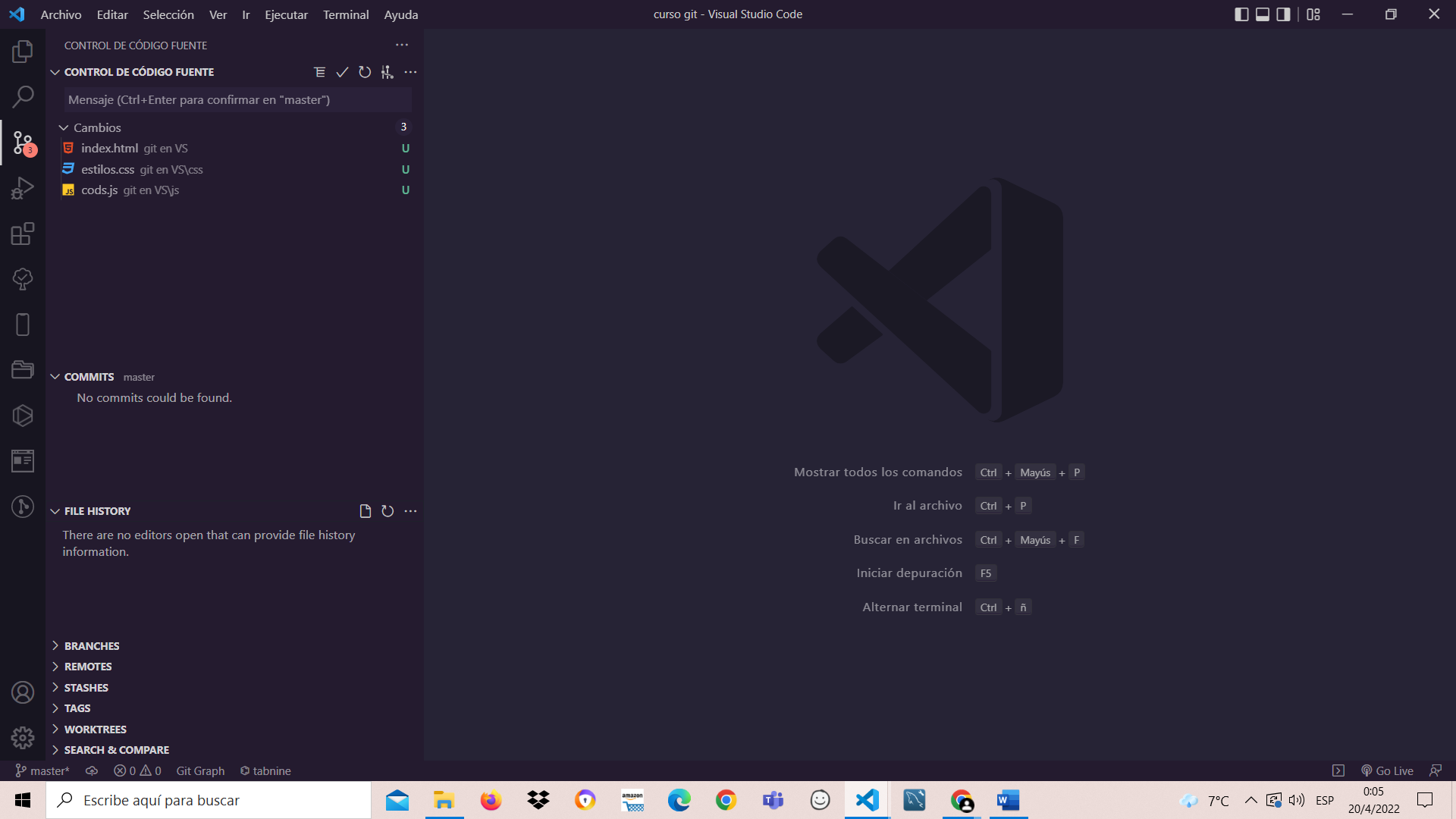
El diagrama anterior nos explica el comportamiento de esos archivos, el editor nos muestra el **DIRECTORIO DE TRABAJO** representado por el nombre y todo lo que este debajo de el hace parte de su contenido, por otro lado, el **AREA DE TRABAJO** está representado por y todo lo que este debajo de el hace parte de su contenido, es decir que el directorio de trabajo con tiene 2 archivos (estilos.css y cod.js) los cuales no están en seguimiento y el área de trabajo contine solo 1 archivo (index.html) y se encuentra preparado o en seguimiento.

# **Agregar los contenidos de archivos al staging área**

Existen 2 formas para agregar al área de trabajo los archivos, la primera será agregar archivos por archivos al staging área o agregar directamente todo en conjunto las modificaciones que se hayan hecho en cualquier archivo de nuestro directorio de trabajo (proyecto ) vamos a haber los 2

Lo primero que tenemos que hacer es dirigirnos al icono de control de código fuente

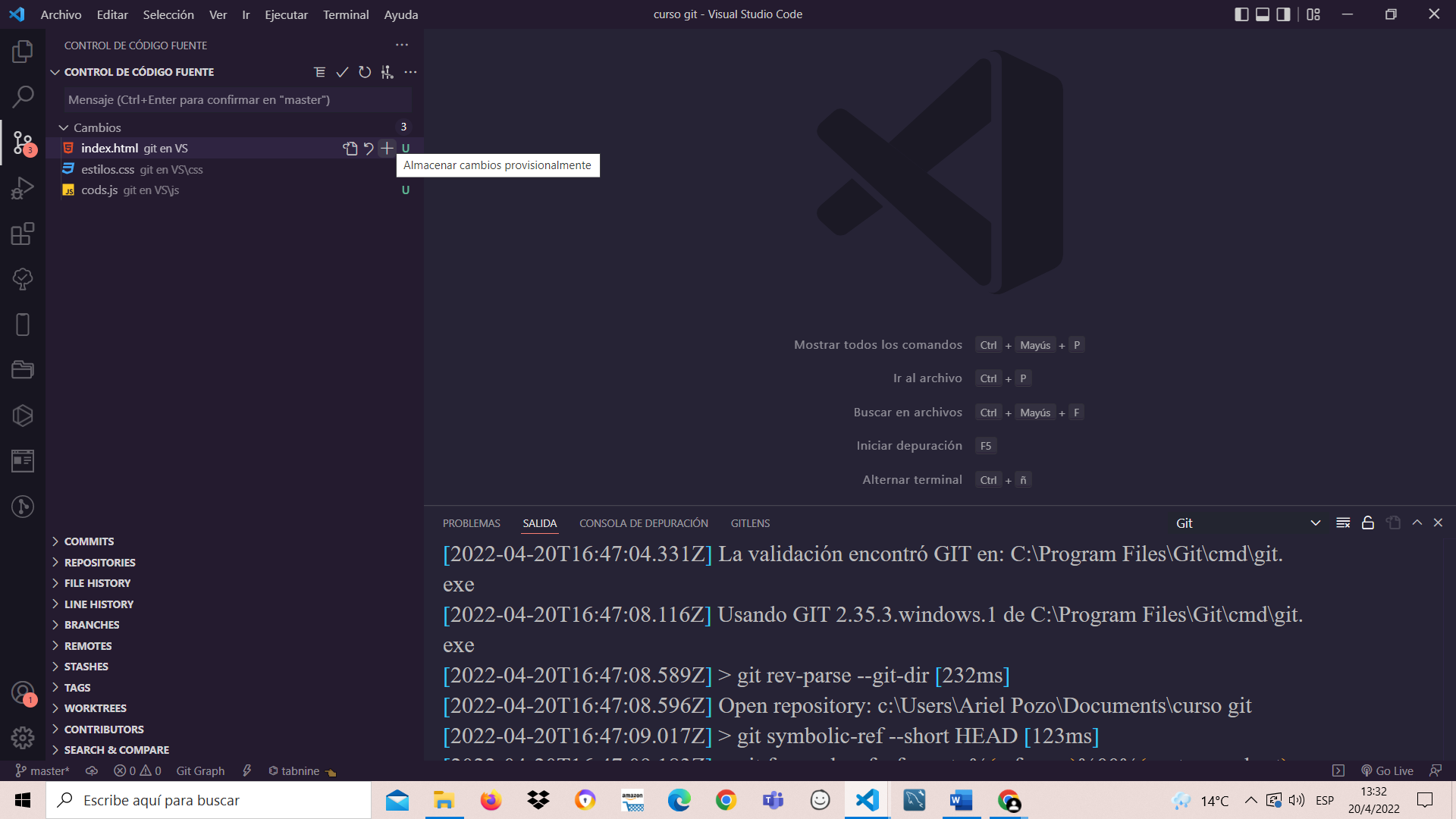




En él se puede observar nuevamente los archivos en los cuales no se encuentran en nuestra área de preparación sino en el directorio de trabajo ahora vamos agregar cada uno de ellos.

## **Agregar archivo por archivo al staging área**

Ahora vamos a asumir que por algún motivo necesitamos agregar solamente a nuestra área de trabajo el archivo que lleva por nombre index.html

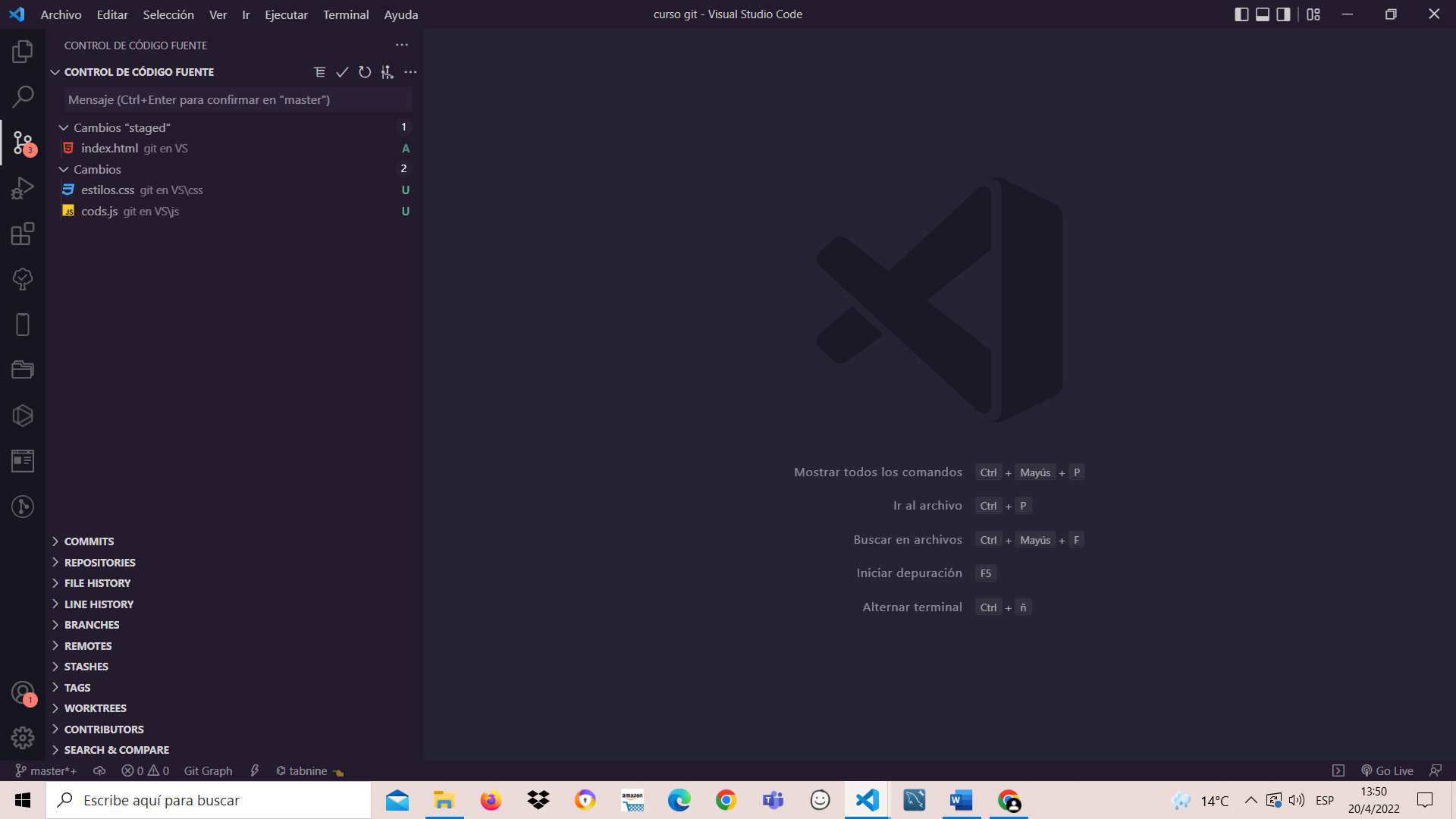


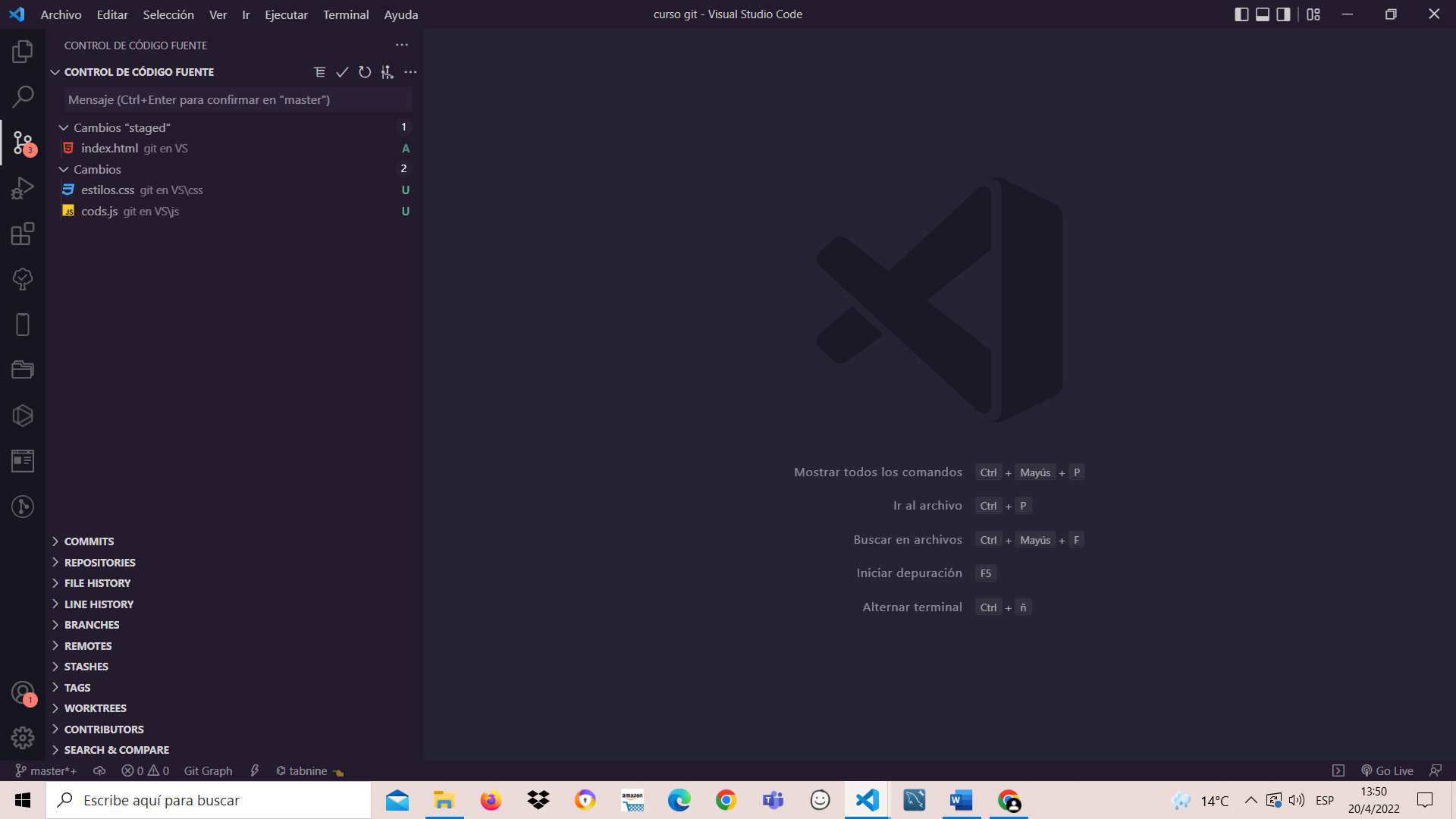
Git, nos proporciona una seria de opciones para poder hacer esto y muchas mas cosas

Si nos ubicamos encima del archivo que queremos agregar nos va a parecer 3 opciones

El símbolo + que nos aparece dentro de las opciones nos dice que ese nos permite almacenar cambios provisionalmente, es decir esa opción nos permite agregarlo a nuestra área de trabajo prácticamente al hacer esto estamos ejecutando la instrucción que asíamos por línea de comandos 

veamos:





2.indicador de archivo

3. Archivos que se encuentran en el

4. Archivo en seguimiento

2.indicador de archivo

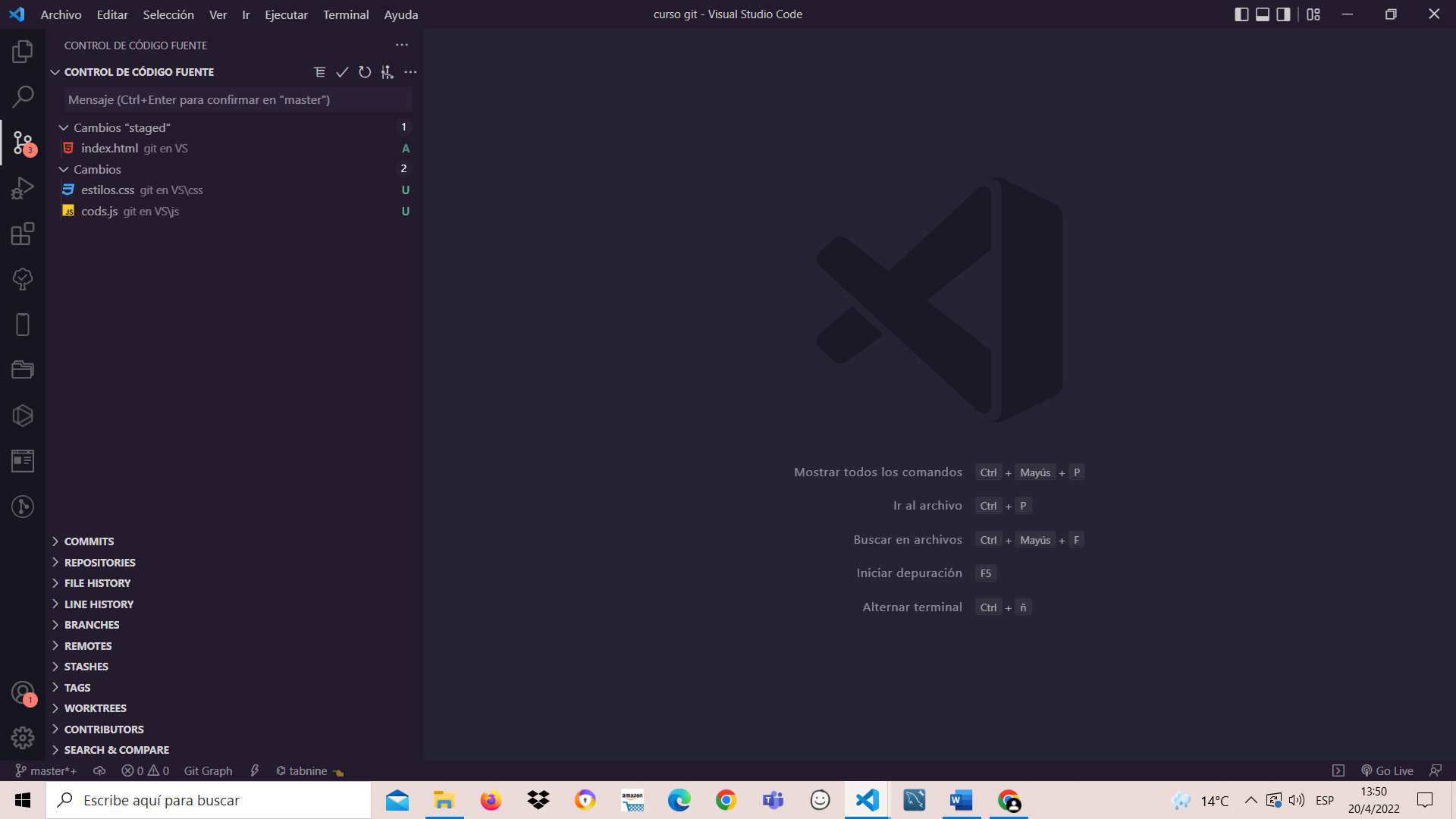
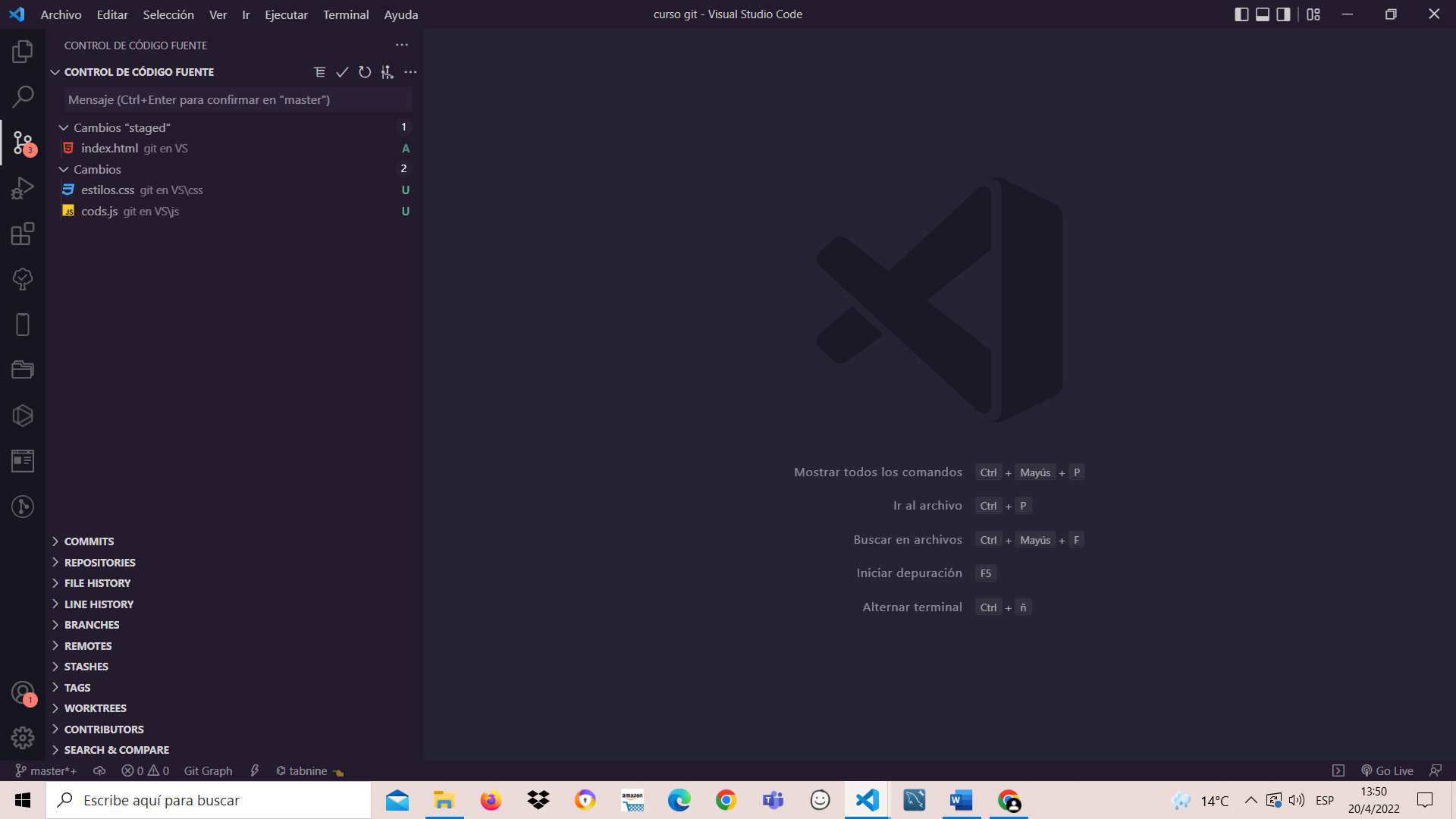
3.Archivos que se encuentra en el

4.Archivos sin seguimiento

1.AREA DE PREPARACION O STAGING ÁREA

1.Directorio de trabajo

Por fuera de área de preparación

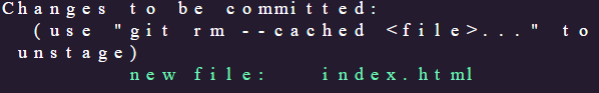
Como se puede observar al hacer clic en el + nos aparece una opción más que dice  indicandonos que ya esta en seguimiento y que hay cambios en nuestra area de trabajo en la parte lateral derecha nos aparce un numero indicandonos el numero o los numeros de archivo que se encuentra en el area de preparacion seguidamente debajo se encuentra el nombre del archivo con la letra A en la parte lateral derecha, ratificando efectivamente con esa letra que ese archivo ya esta preparado en nuestra area de trabajo.

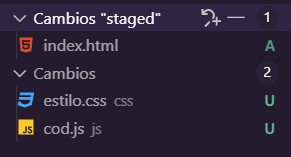
Seguidamente debajo del todo nos aparece la opcióncon un numero

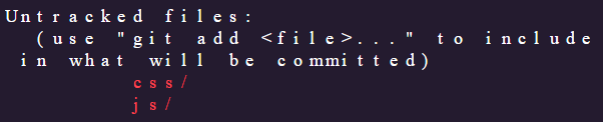
El cual nos indica el número de archivos que se encuentra sin rastrear y debajo del todo los nombres de los archivos que se encuentran en el con la letra U. en pocas palabras la imagen anterior lo que nos esta ejecutando automaticamente es el comando







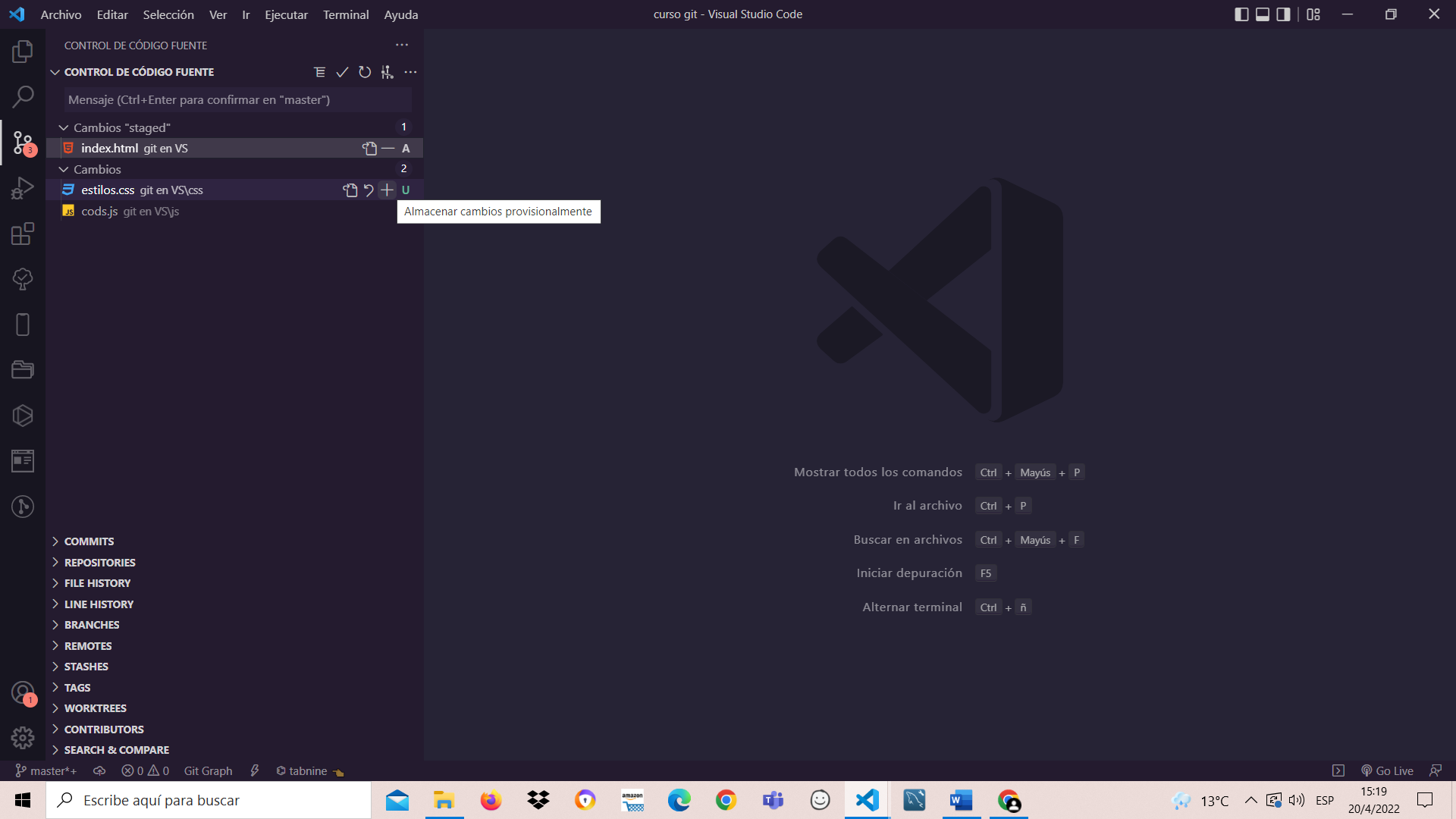


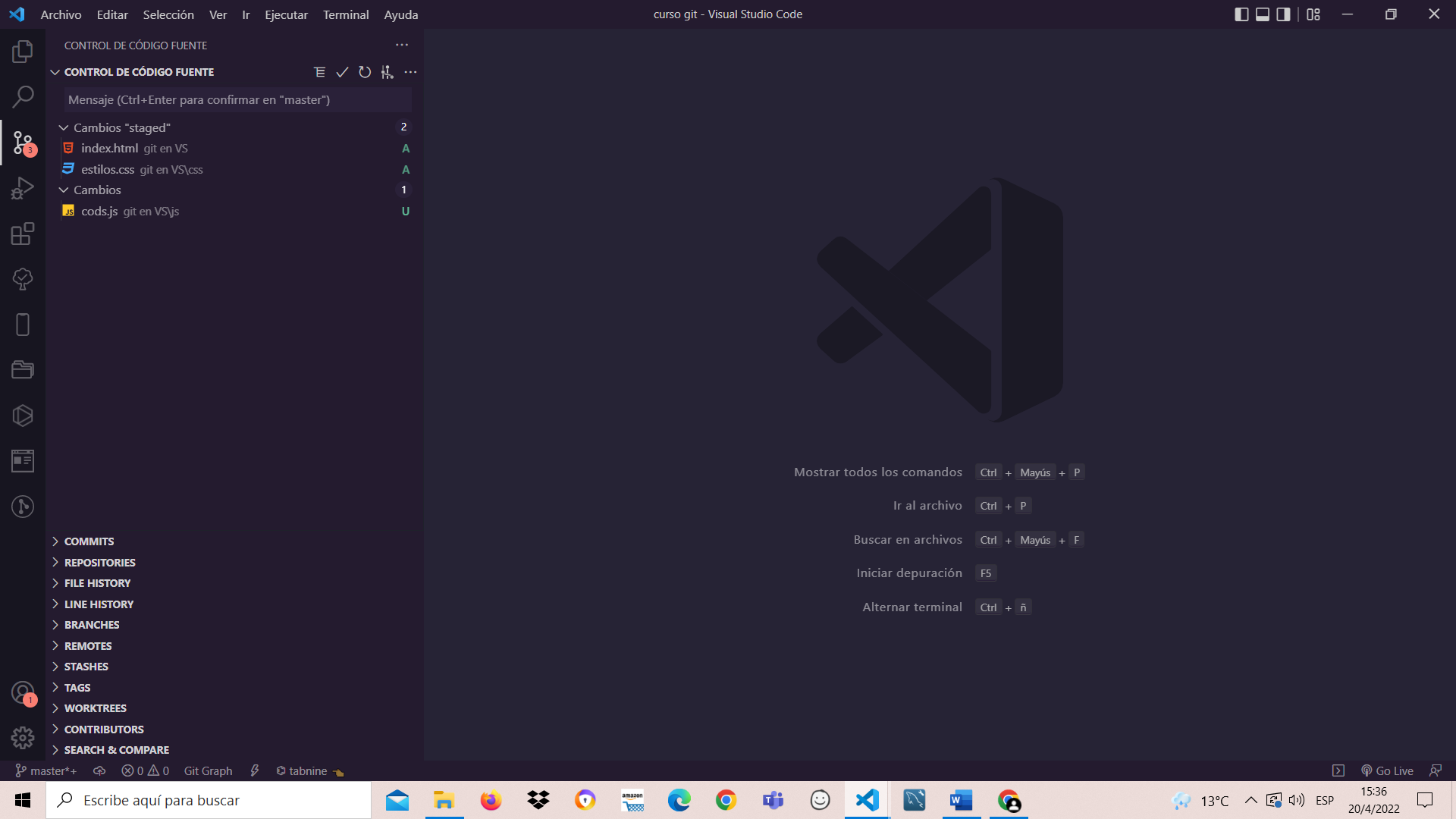


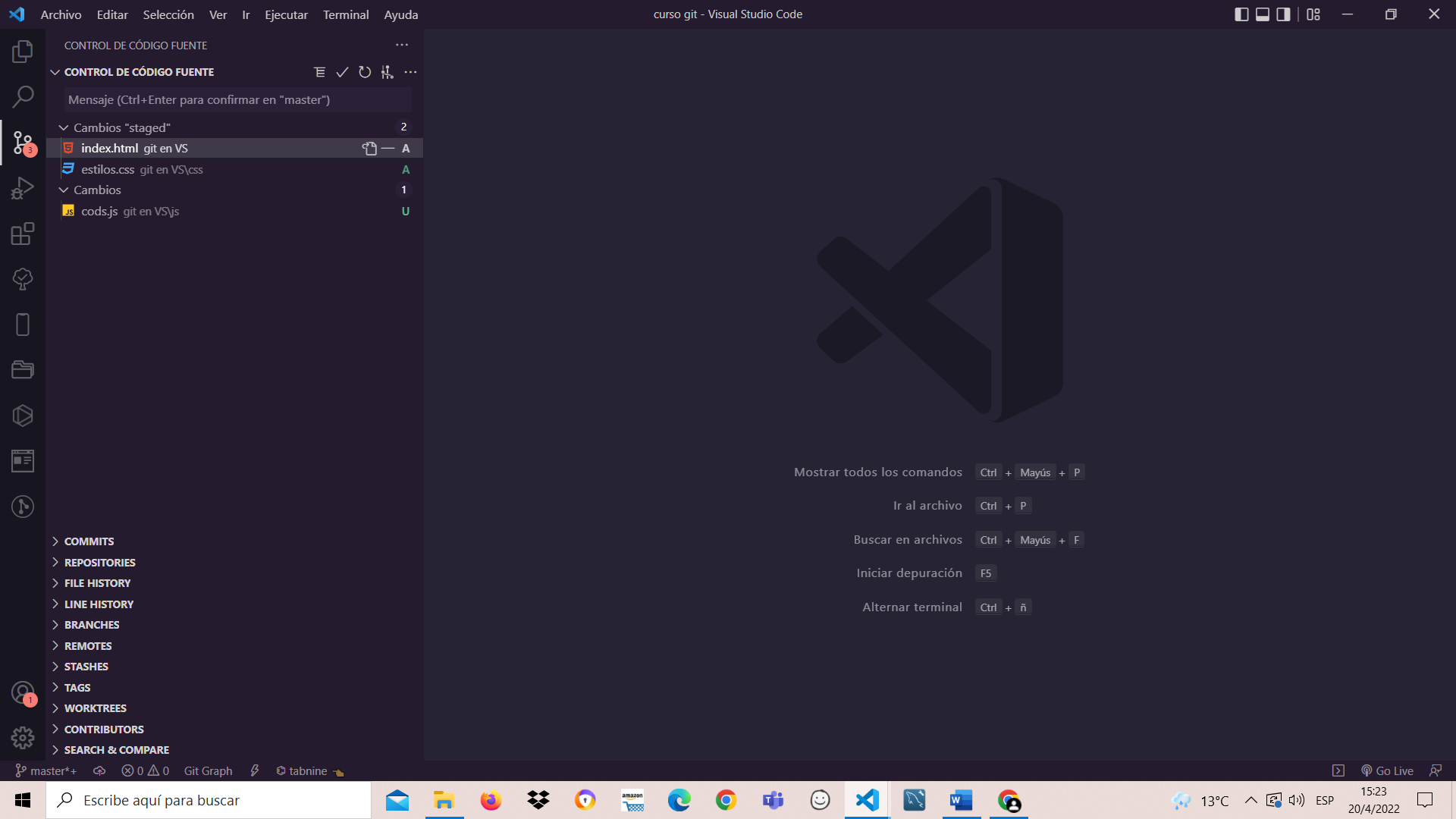
Como los archivos que no estan en seguimiento estan contenido dentro de una carpeta y ninguno de ellos esta en el area de preparacion al momento de hacer por comando

 Me muestra el nombre de las carpetas informandome que en esa carpetas se encuentran archivos sin preparar al que pertenece cada uno

Ahora agregaremos un nuevo archivo a nuestra area de trabajo esta vez lo hareamos con estilos.css veamos:

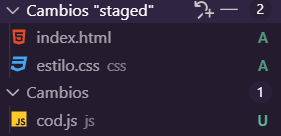


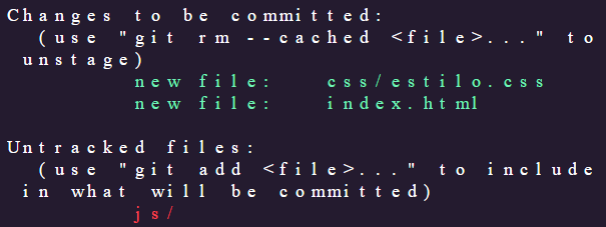


Hemos visto una vez más la integralidad de git de manera visual ahora vemos que en el área de trabajo se encuentran 2 archivos identificado por el numero y el nombre de los archivos contenidos en él, seguido de la letra A indicandome que los archivos estan preparados, tambien podemos notar que el directorio de trabajo precedido por el nombre **Cambios** tambien a cambiado y que ahora se encuentra un solo archivo.

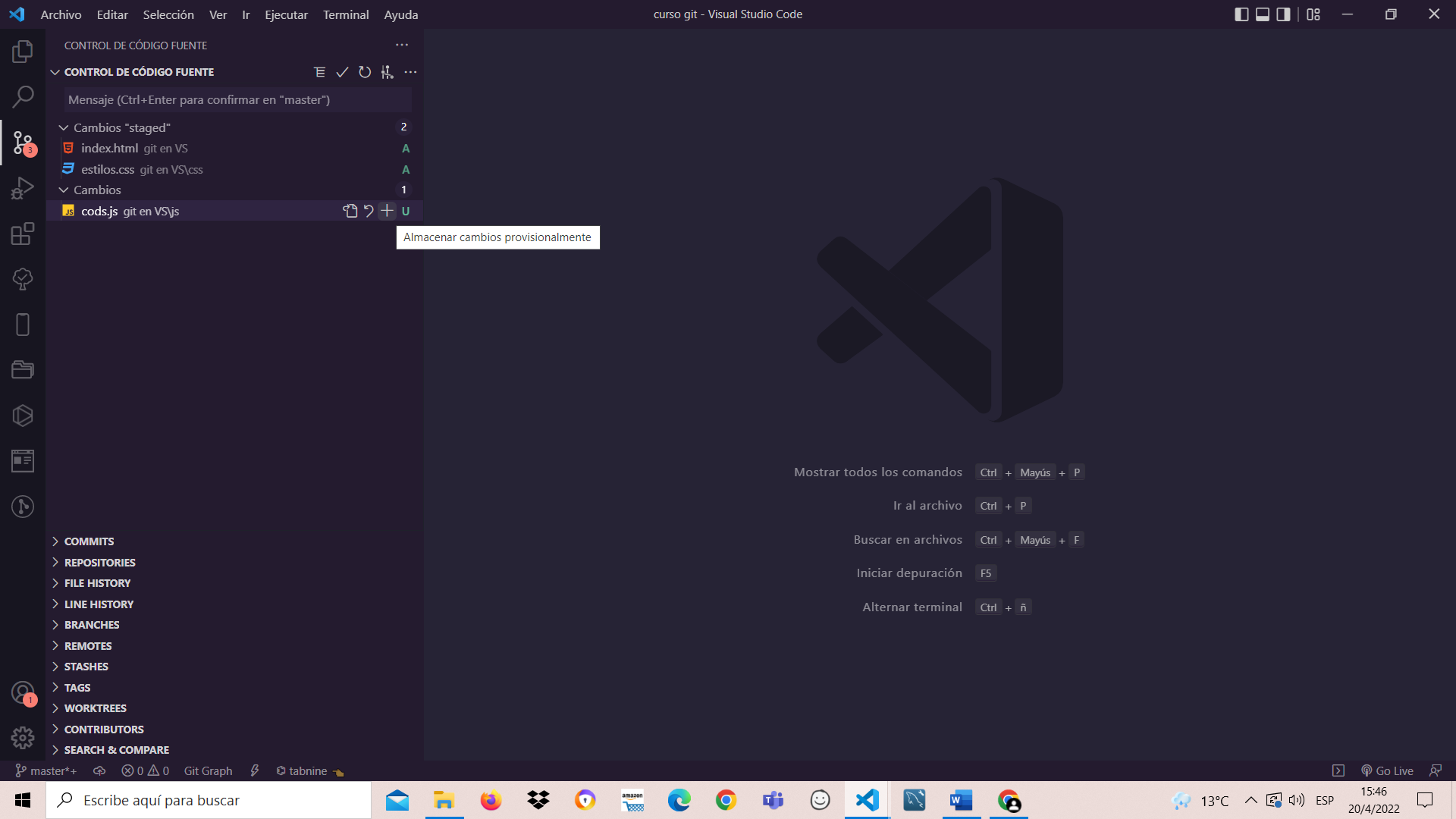
esto es equivalente hacer por comando la instrucción

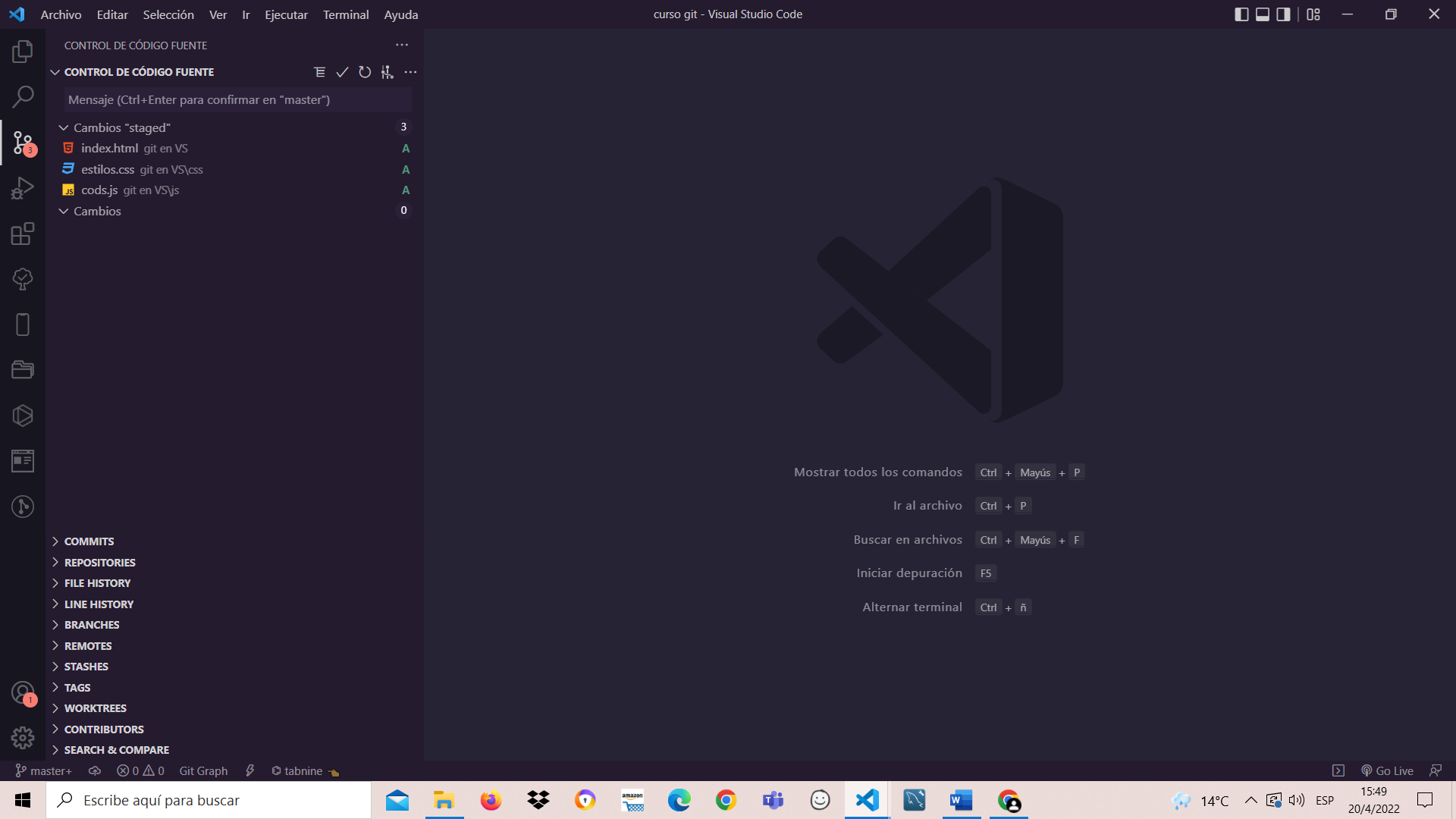
ya que el archivo estilo.css esta contenido dentro de la carpeta css





Por ultimo agregaremos el archivo que se encuentra por fuera del area de preparacion veamos:



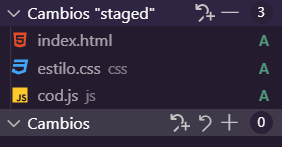
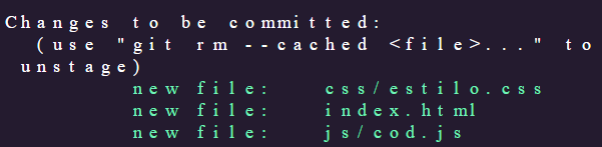


Hemos a gregado el ultimo archivo que estaba sin seguimiento ahora todos los archivos se encuentran en el area de preparacion, si hechamos un vistaso a nuestro directorio de trabajo ahora nos dice que se encuentra en cero indicandonos que se encuentra limpio.

esto es equivalente hacer por comando la instrucción

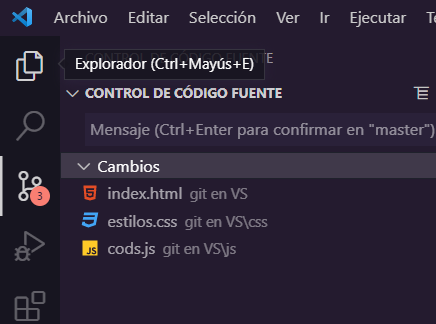






### **Agregar múltiples archivos al staging área**

Nuevamente tenemos todos los archivos en nuestro directorio de trabajo veamos



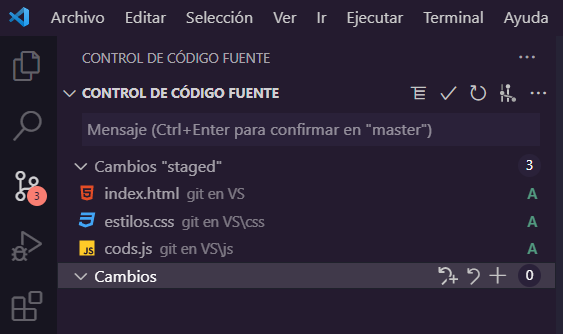
Para agregar todos los archivos que se encuentran en nuestro directorio de trabajo

Tenemos que ubicarnos encima del mismo de la siguiente manera



Como podemos observar nos muestra todas las opciones, cuando nos hubicamos sobre el simbolo + nosdice que podemos almacenar todos los cambios que se encuentra en ese directorio de trabajo al hacer click ahi inmediatemente pasan todos los archivos al

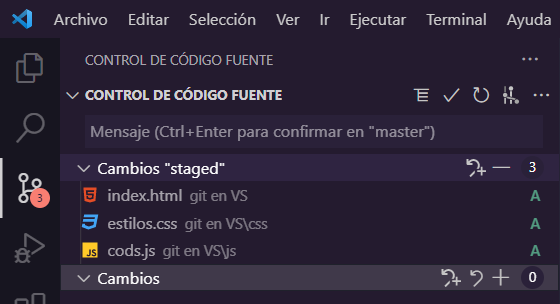
staging área esto es equivalente hacer por comando la instrucción veamos:



Como explicamos hace un instante, si miramos nuestra area de trabajo esta vacia, el indicador de archivos medice que tiene  “cero” archivos, si miramos nuestra area de trabajo  nos muestra que en el indicador de archivos ahí “tres” archivos en su ubicación y que ya se encuentra en seguimiento “preparados”

# **Preparar archivos modificados**

vamos a modificar un archivo que ya está preparado llamado index.html veamos que tenemos hasta el momento:



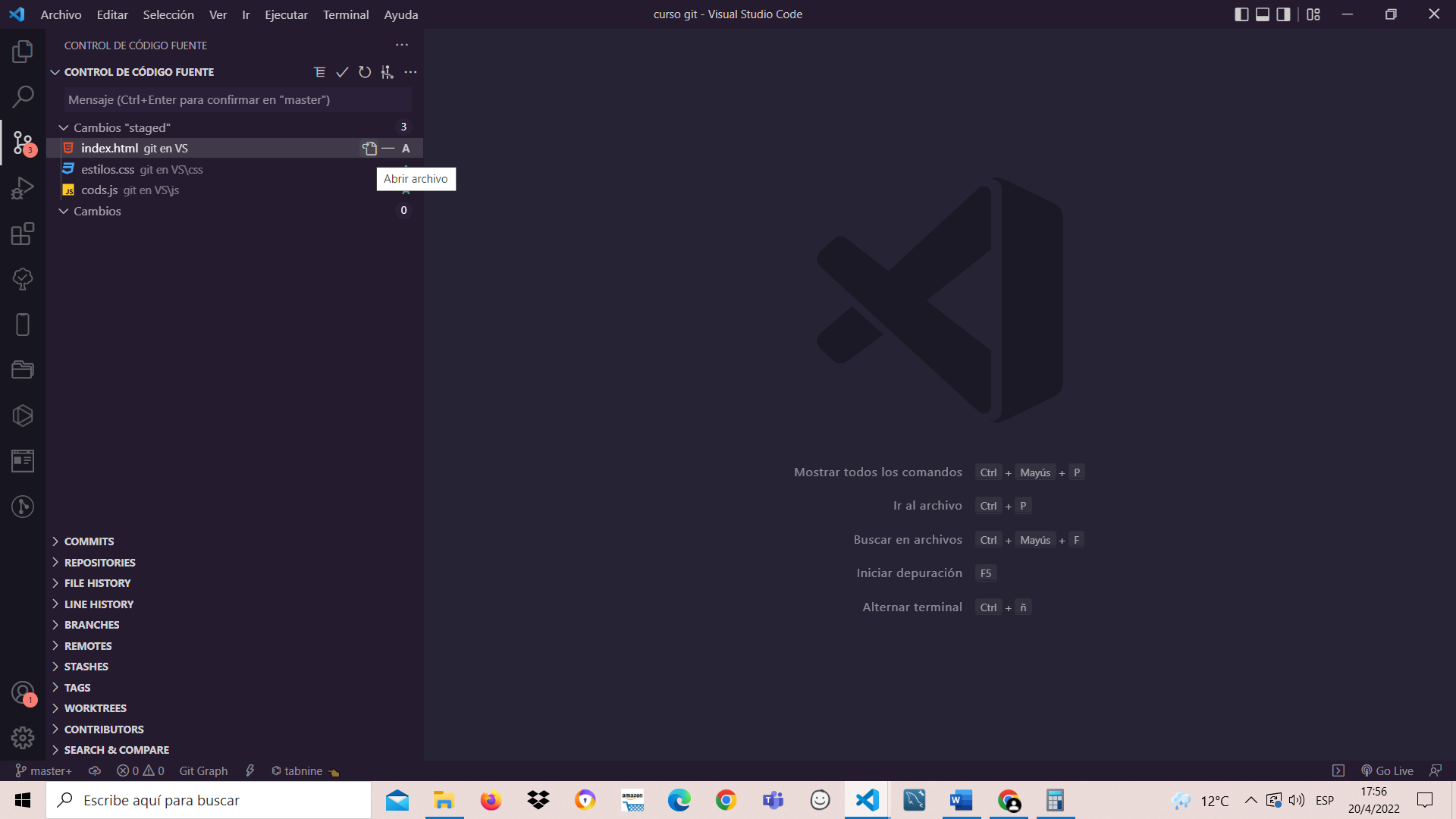
Como podemos observar todos los archivos se encuentran en nuestra área de trabajo y están preparados y en seguimiento, vamos a modificar uno de ellos veamos.

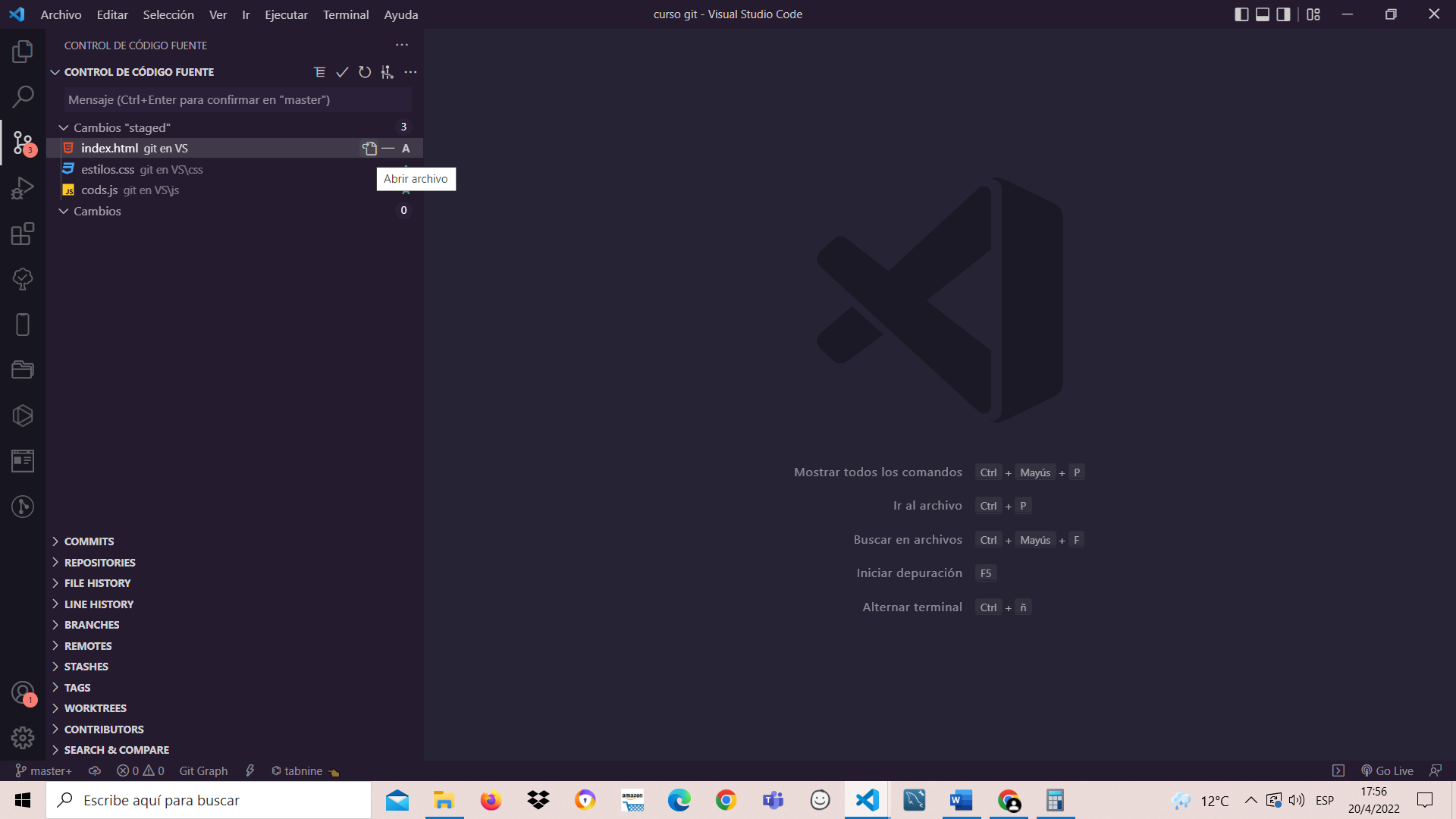
Para modificar el archivo index.html lo podemos hacer de 2 manera

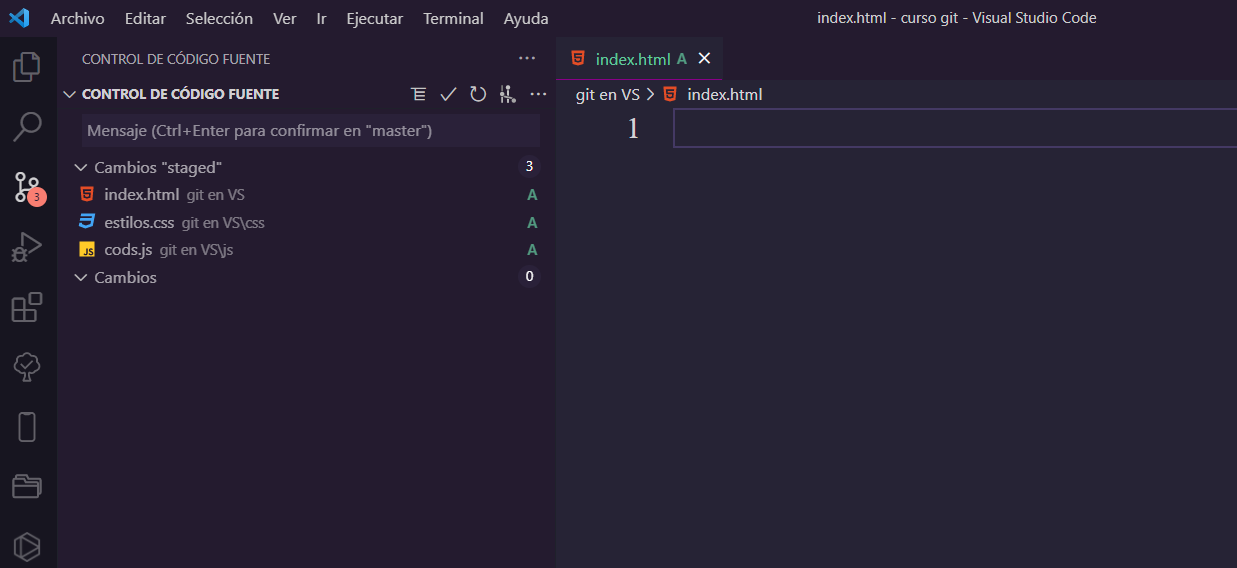
1 directamente desde las opciones que me brinda visual studio code al posicionarme encima del archivo y abrirlo directamente desde ahí.

2 dirigirnos al explorador de archivo abrir la carpeta que contiene el archivo y posterior mente abrirlo. Vamos hacerlo de las 2 maneras veamos:

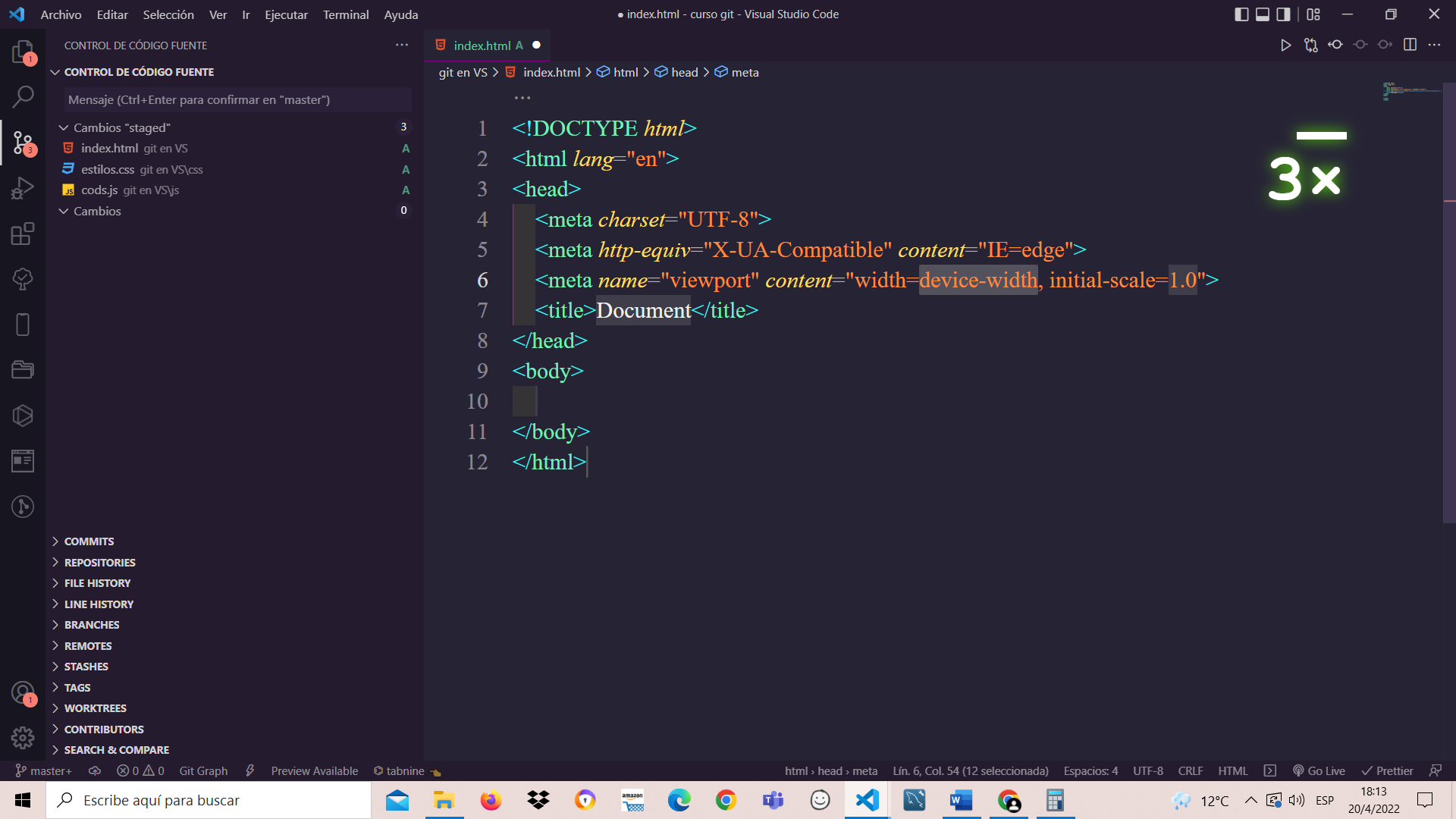
1.



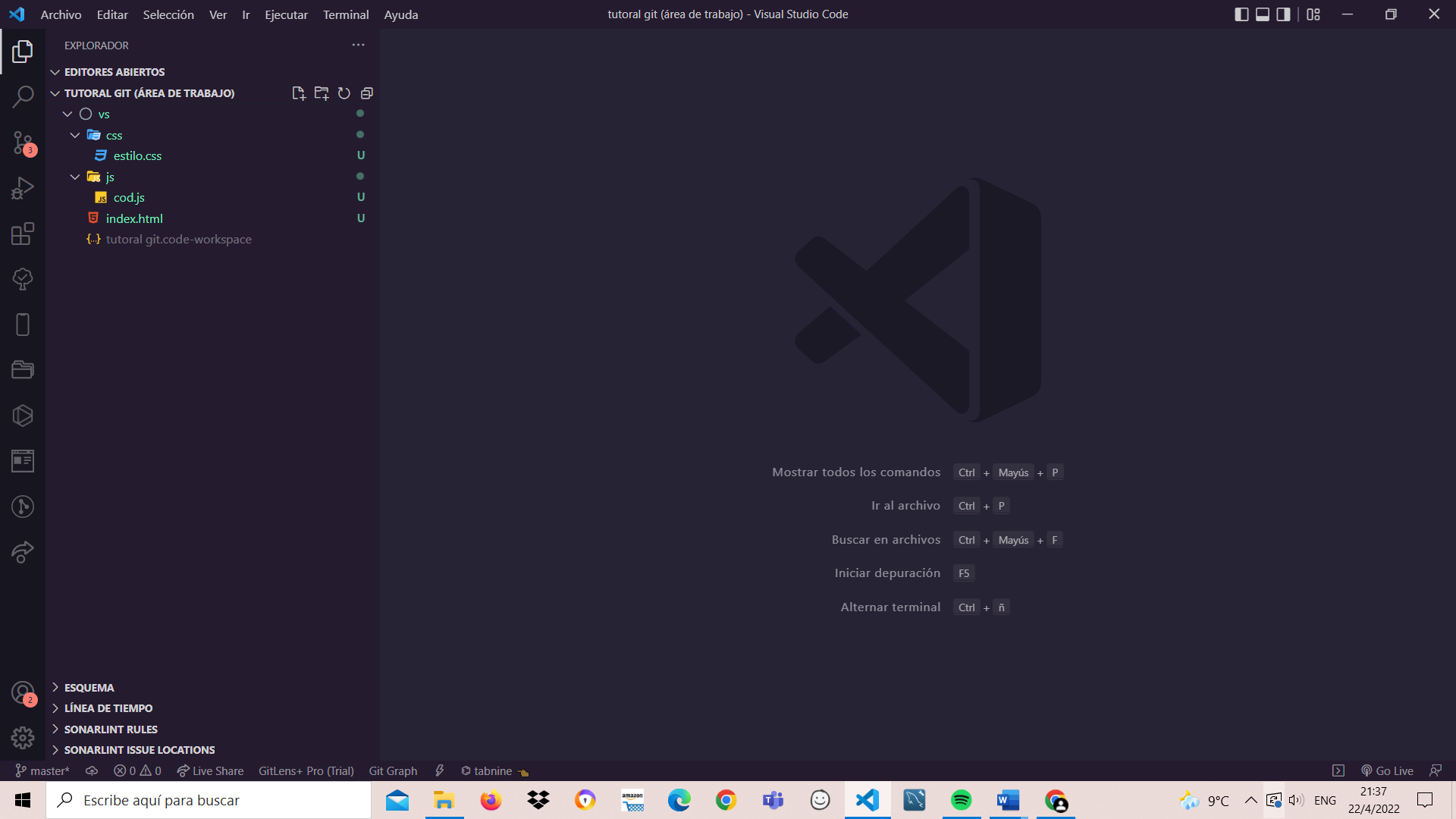
Al darle clik en la opción que podemos visualizar se nos abrirá el contenido del mismo veamos.



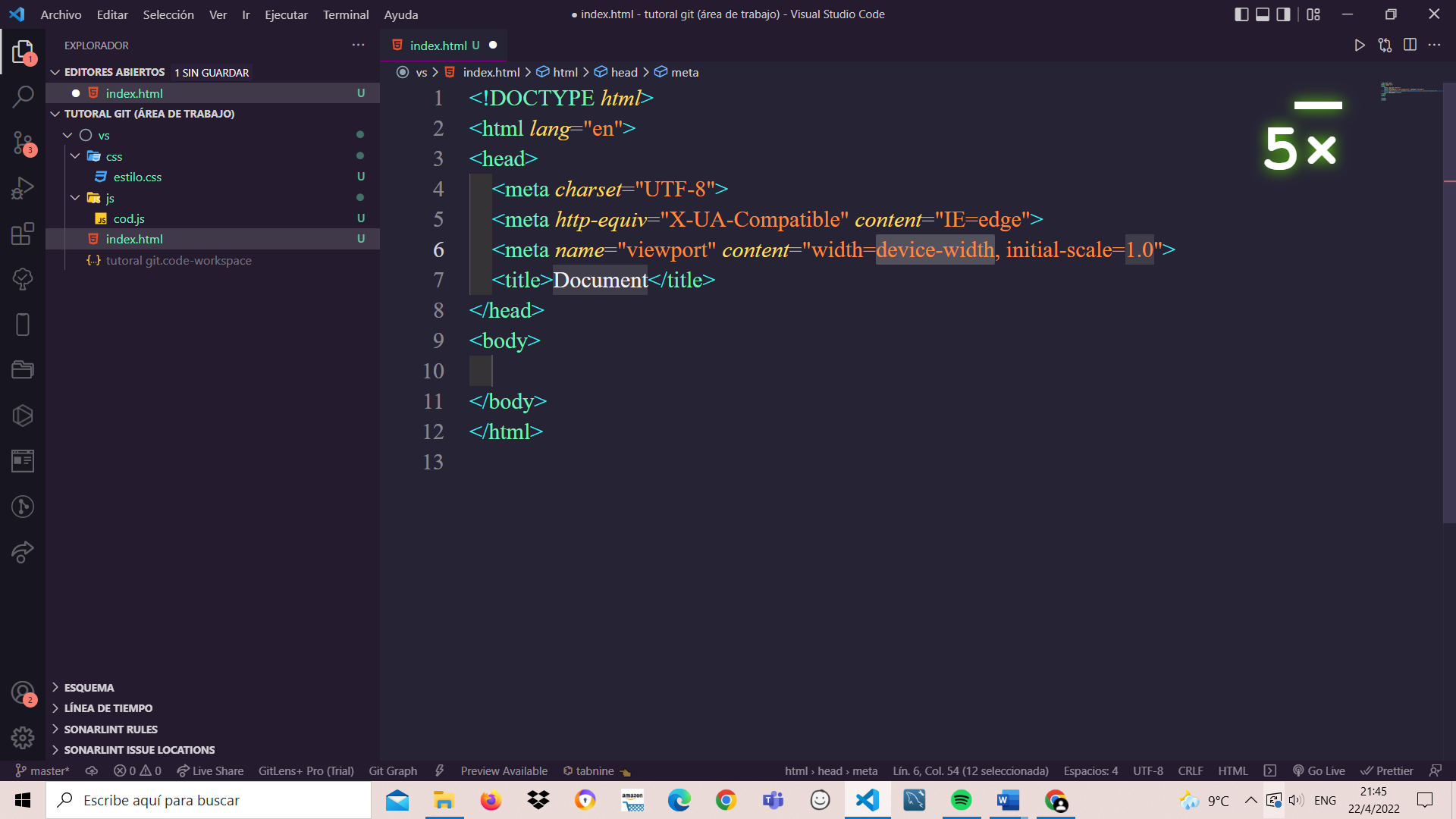
Apartir de ese momento podemos ver el archivo abierto y notamos que su contenido esta vacio, ya que no contiene ninguna linea de codigo, vamos a agregar algo para ver que pasa.

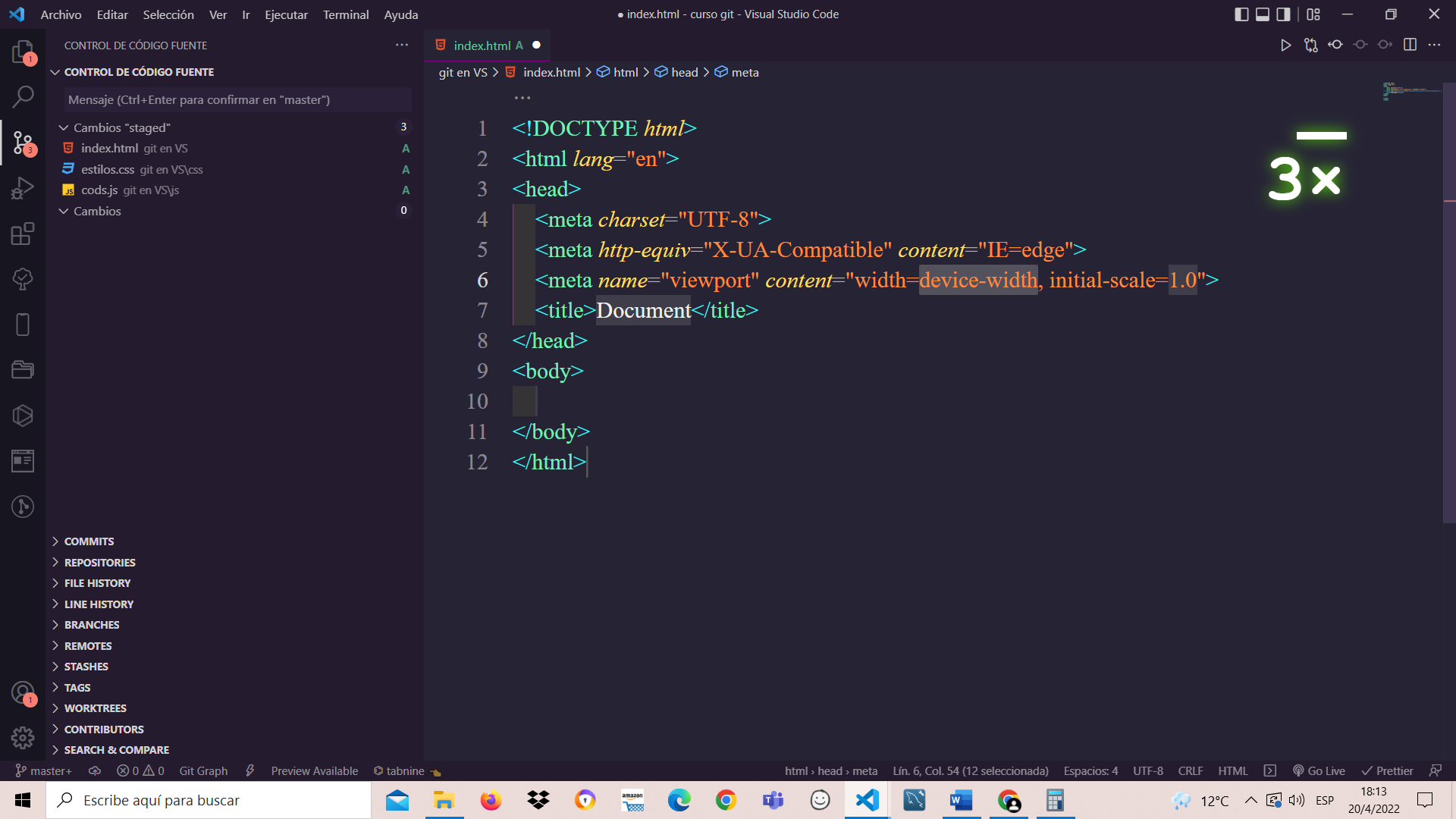


2.



Nos dirigimos al explorador de archivos, damos clic al archivo que vamos a modificar y editamos su contenido

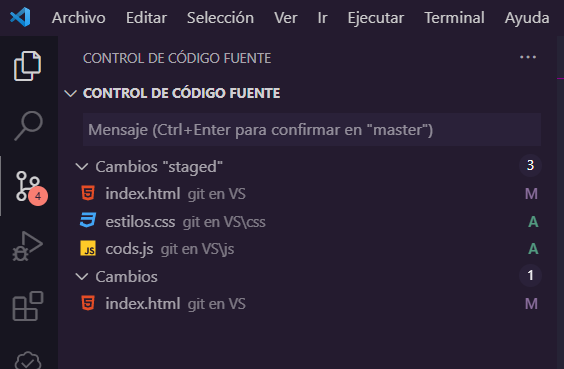


Al agregar contenido a nuestro archivo podemos notar en la pestaña donde aparce el nombre del mismo un punto blanco indicando que las modificaciones en el archivo no se han guardado .

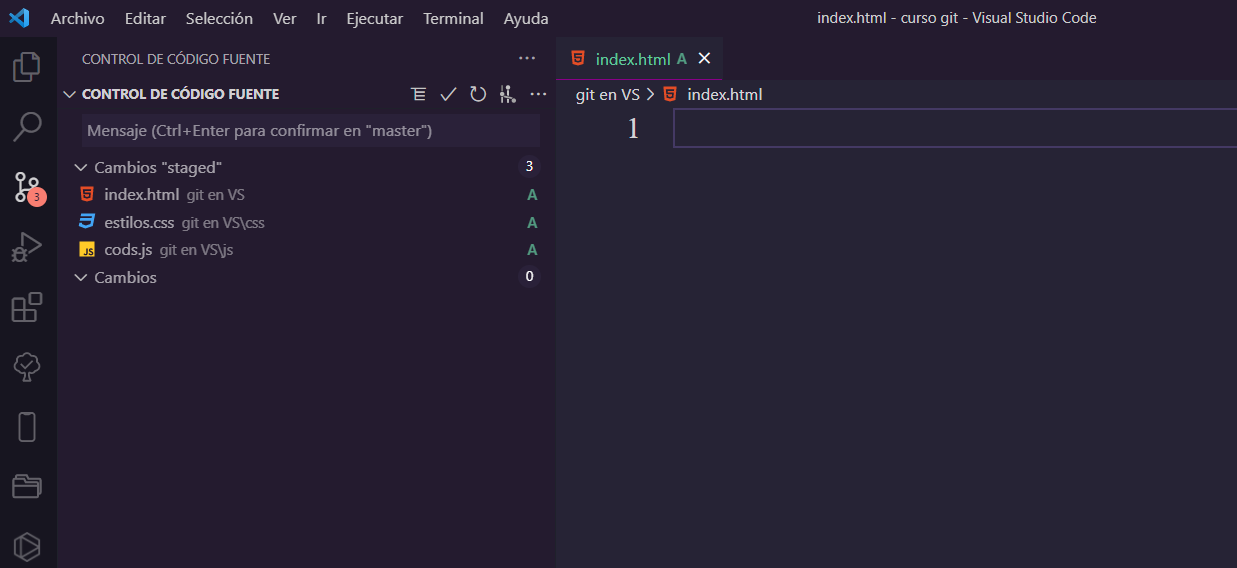
Una vez guardado el archivo veamos que pasa



Si nos fijamos en el control de codigo nuevamente podemos ver:



Notamos que una las letras que identifican el estado de cada uno de los archivos ha cambiado, el estado del archivo index.html es una letra M indicando ahora que el archivo que se encuentra en nuestra area de trabajo a sido modificado, si observamos nuestro directorio de trabajo observamos el mismo archivo indicando que las modificaciones que se hicieron en el archivo despues de almacenar el ultimo cambio en nuestra area de trabajo no se encuentra en seguimiento, para entender mejor veamolo de manera grafica.

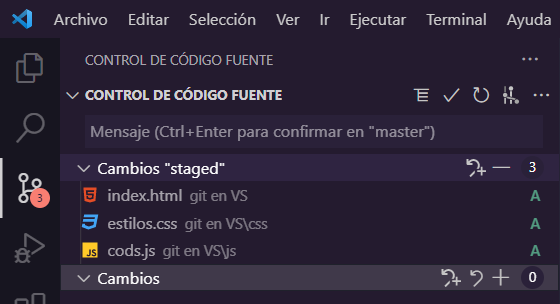


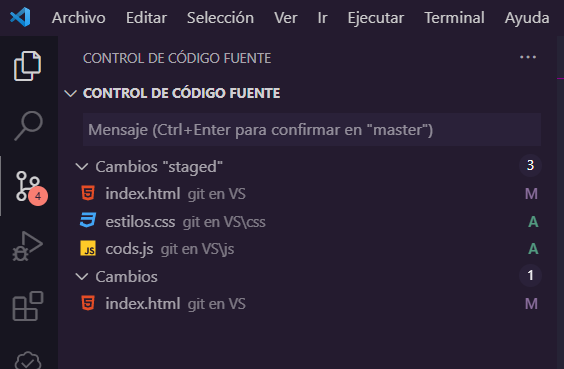




A

M



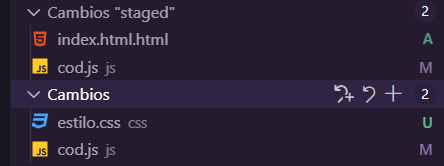


La imagen anterior me muestra el estado en los que se encontraban los archivos en un momento dado, la imagen de la izquierda me representa que ese archivo sin contenido se encuentra en el area de preparacion, la imagen de la derecha me representa que ese archivo fue modificado y es presisamente ese contenido que se encuentra sin seguimiento.

# **Estado Abreviado**

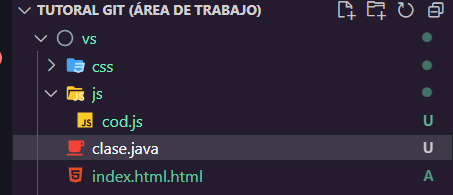
antes que nada para ver los estados de los archivos es equivalente a ejecutar el comando

Este tipo de estado ya lo hemos venido viendo a lo largo de este curso y son básicamente las letras que identifican el estado de los archivos, hasta el momento hemos visto 3



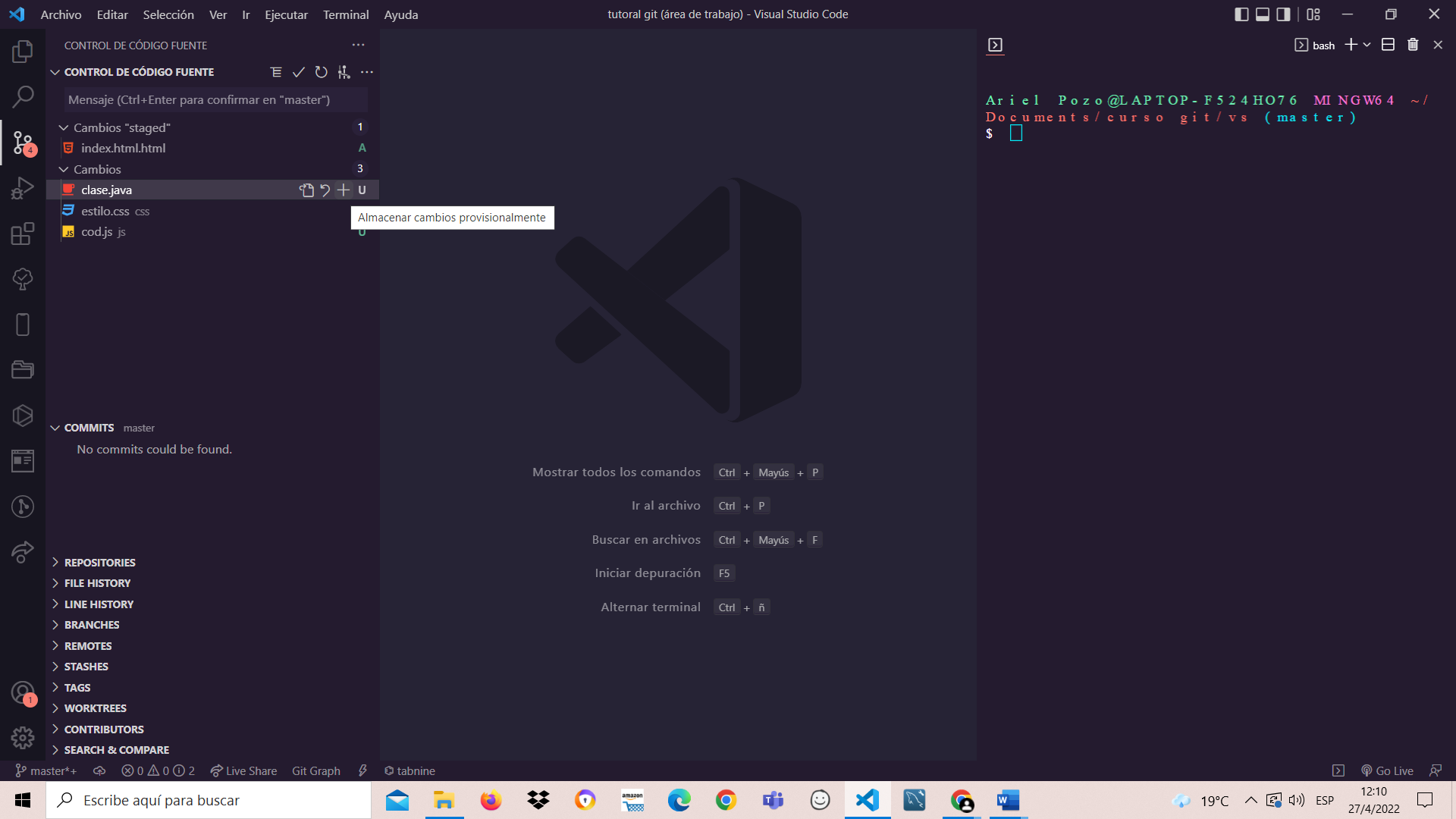
|  |  |
| --- | --- |
| U | **Sin seguimiento** |
| A | **Preparado** |
| M | **modificado** |

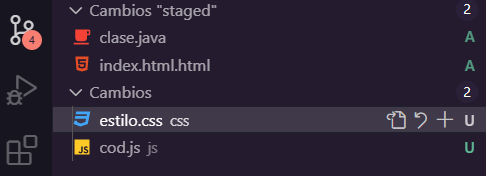
Vamos a ver 1 mas, como indicamos anteriormente nos falta ver un estado, el cual es la letra D y representa un archivo que a sido eliminado veamos.

Vamos a agregar un nuevo archivo a nuestro proyecto

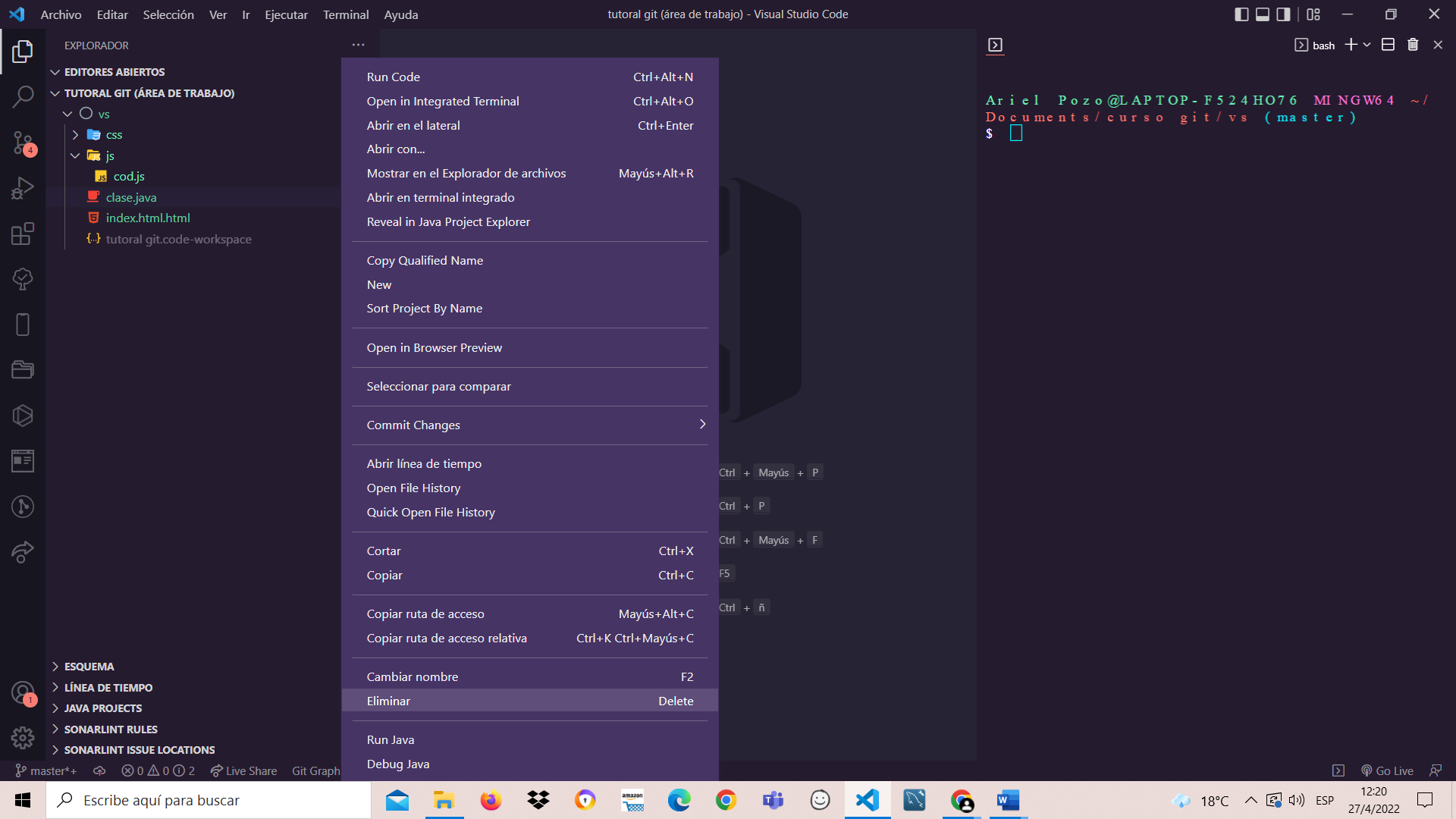
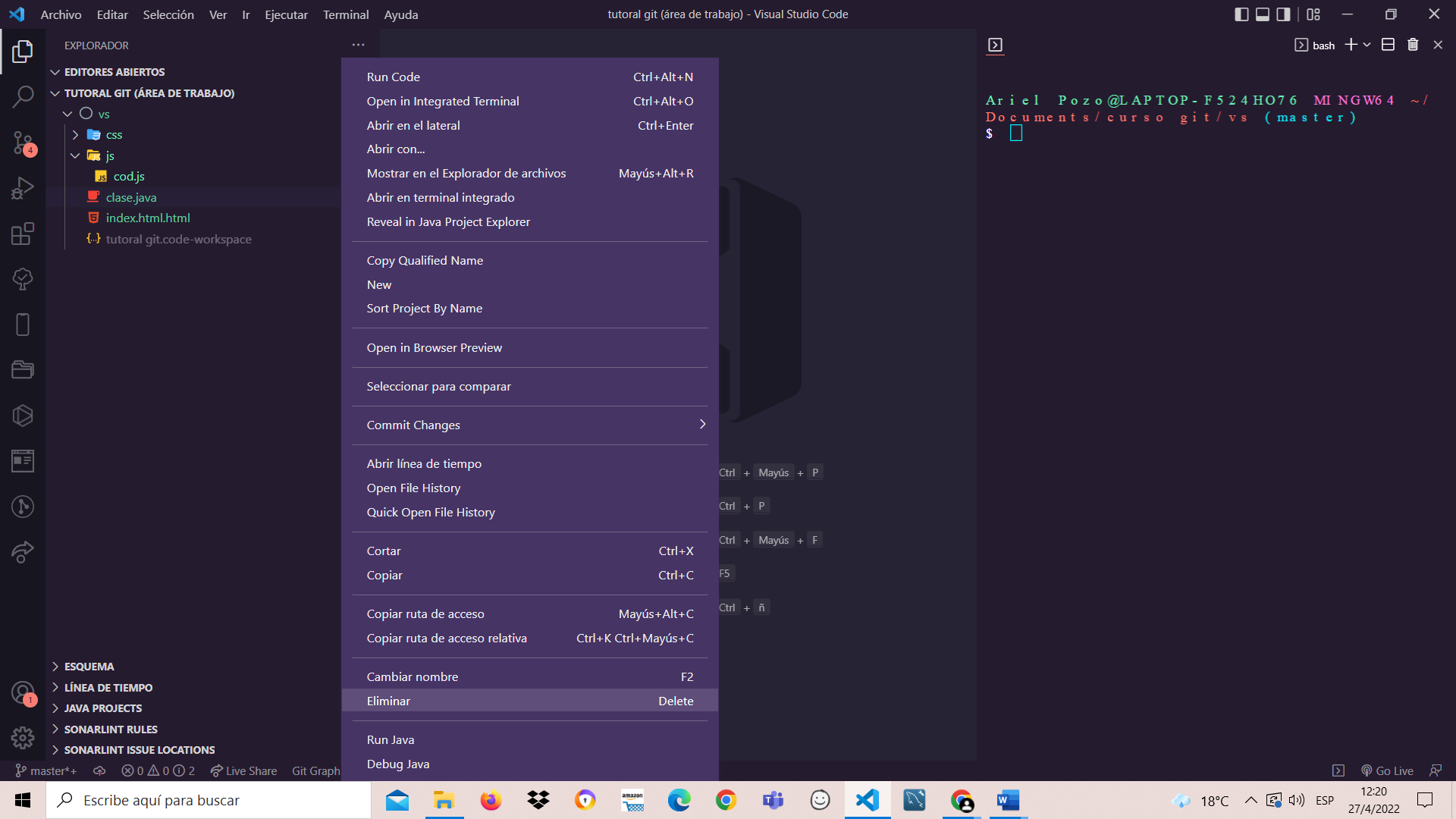
Nuevo archivo

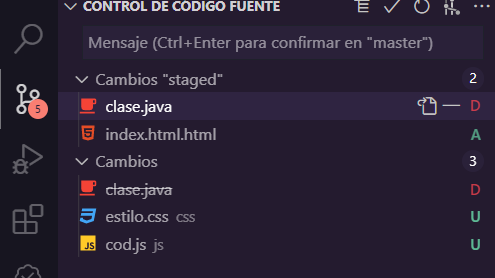
Para poder ver este estado el archivo debe estar en nuestra área de trabajo veamos





Ahora imaginemos que por algún motivo necesitamos eliminar ese archivo veamos





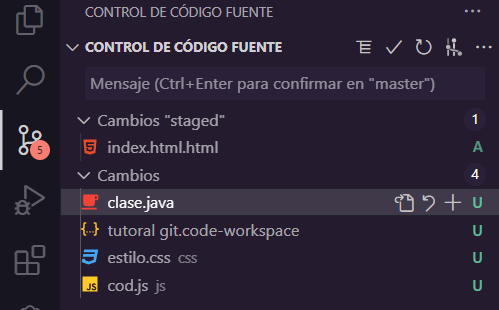
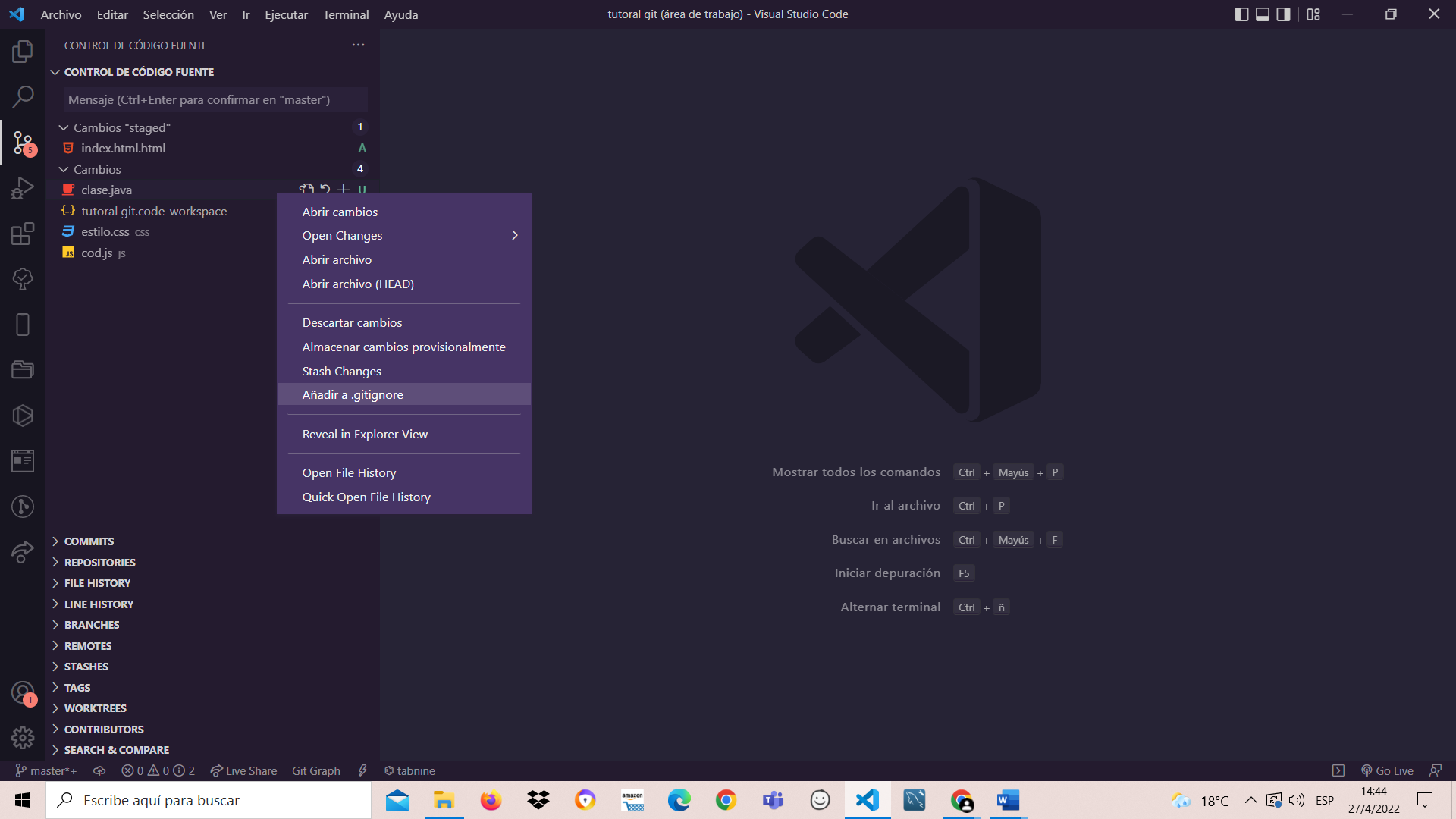
Esto es equivalente hacer por comando la instrucción

mas adelante explicaremos con mas detalles la eliminacion de un archivo en seguimiento

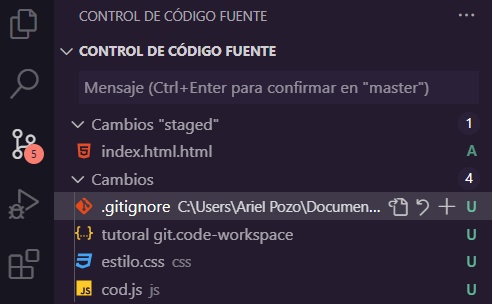
# **Ignorar Archivos**

A veces, tendrás algún tipo de archivo que no quieres que Git añada automáticamente o más aun, que ni siquiera quieras que aparezca como no rastreado. Este suele ser el caso de archivos generados automáticamente como trazas o archivos creados por tu sistema de compilación. En estos casos, puedes crear un archivo llamado .gitignore que liste patrones a considerar. En este caso visual studio code nos genera el archivo .gitignore automáticamente solo seleccionando el archivo que queremos ignorar, lo primero que tiene que hacer es asegurarce que eñ archivo a ignorar este por fuera del área de preparacio}

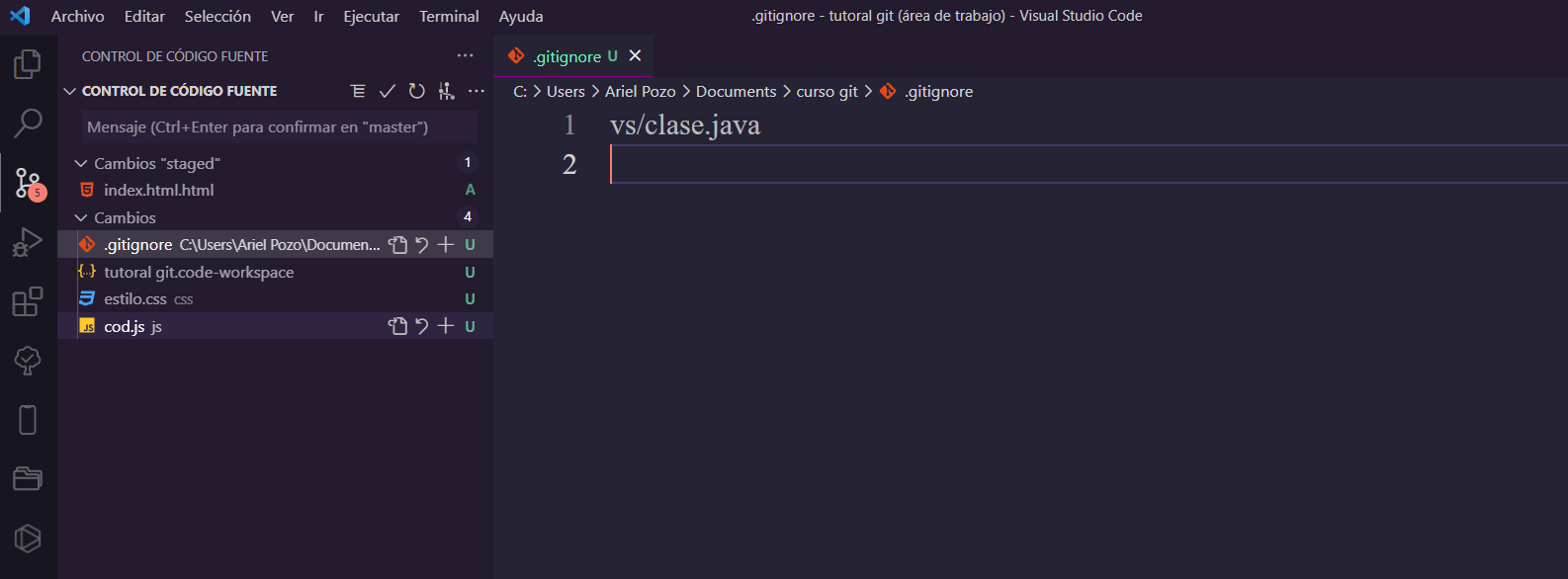
veamos



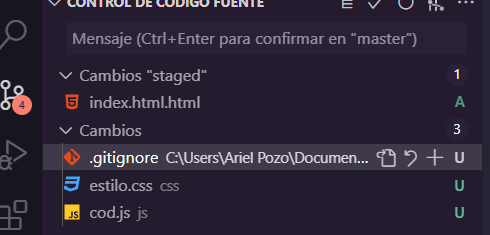
en este caso tenemos un archivo clase.java, a la cual queremos que sea parte de los archivos ingnorados, si damos clic en añadir a .gitinore nos genera un archivo con esa extension la cual contendra la lista de los archivos ignorados veamos.



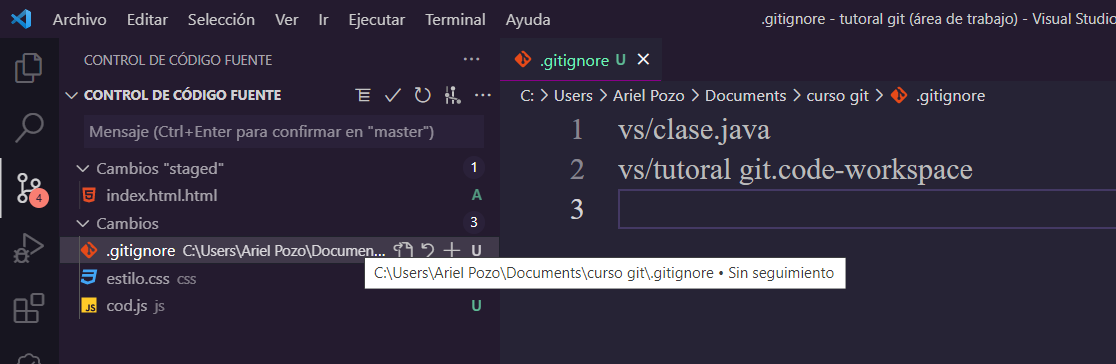
Archivos a ignorar

Si miramos el contenido del archivo .gitignore miraremos una lista de archivos contenidos dentro de el, la cual me indica los archivos que no queremos que se le apliquen seguimiento es decir los archivos que git innore vemamos.

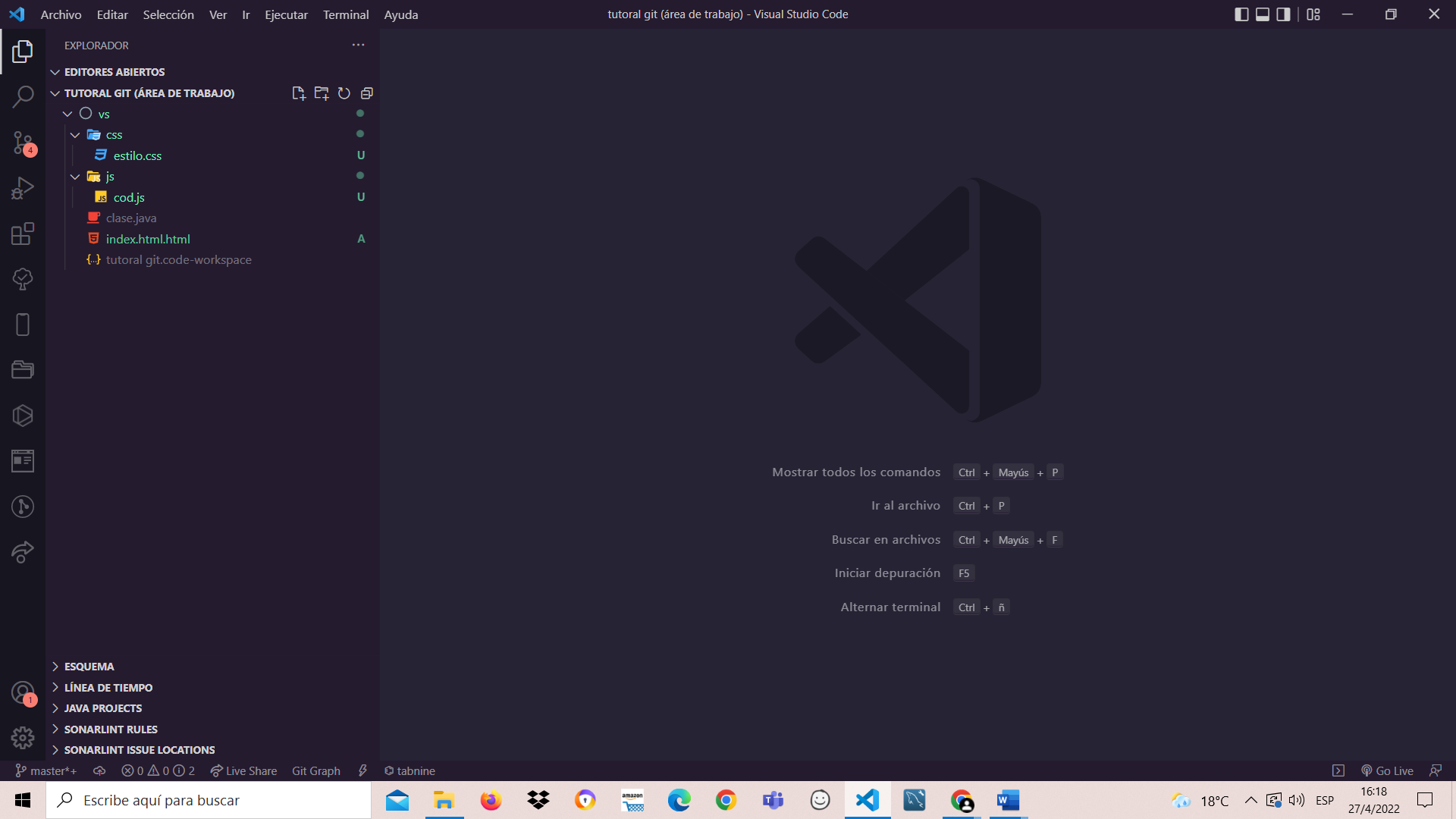
Como podemos observar el contenido del archivo .gitinore contiene en la linea 1 el nombre del archivo que queremos ignorar en este caso le estamos indicando a git que ignore el archivo que lleva por nombre clase.java y como podemos ver ese archivo ya no aparace mas en nuestro control de codigo ahora vamos a umentar la lista y añadiremos el archivo que lleva por nombre

veamos

En este momento hemos agregado el archivo mencionado al .gitignore y si miramos el contenido del mismo vemos

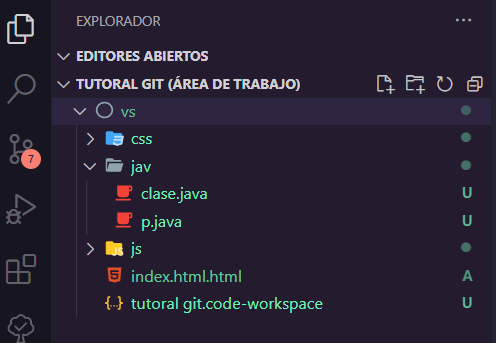


Que Se ha agregado una linea mas en este caso se encuentra en la linea 2 el archivo que queremos ingnorar ademas tambien podemos ver que ya no se encuentra dentro de nuestro directorio de trabajo.



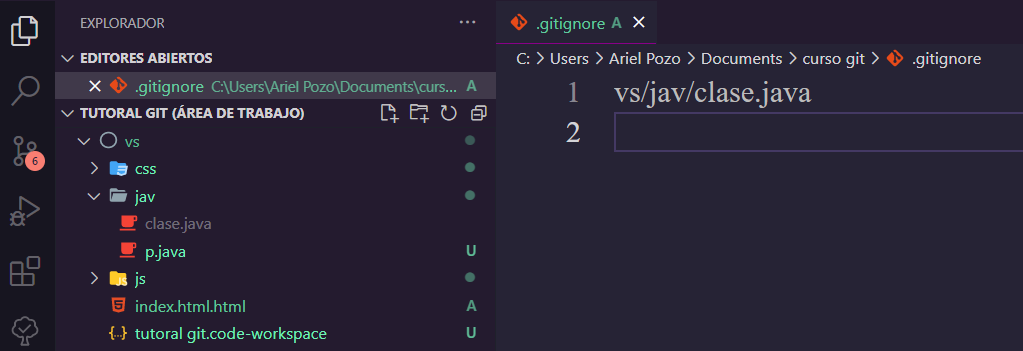
La imagen anterior es el visualizador de archivos en el podemos observar los dos archivos que se encuentrar agregados en el .gitinore en el notamos que el estado precisamente de esos 2 archivos ya no se encuentran activos indicando que git los esta innorando y no lo ha puesto en seguimiento.

Ahora vamos acrear una carpeta y dentro de el los archivos a ignorar.



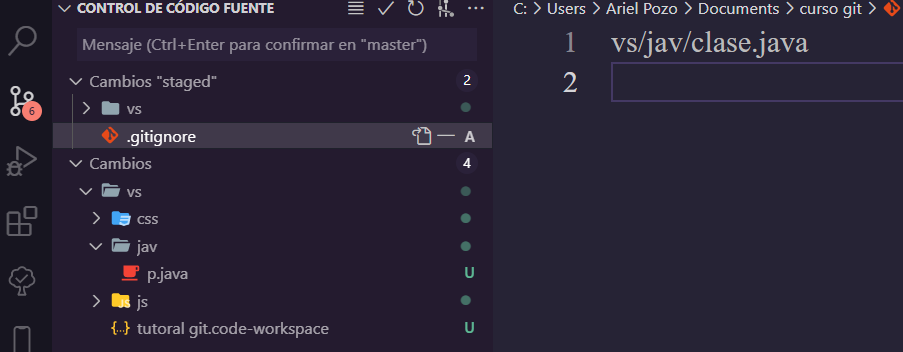
Archivos nuevos

Cómo podemos observar dentro de la carpeta jav, tenemos 2 archivos los cuales reciben el nombre de clase y p, ambos con la terminación .java vamos a empezar a trabajar con ellos.



En la imagen anterior se puede observar en el explorador de archivos que la clase.java no contiene ningun estado esdecir que no contiene ninguna letra (U A M D).

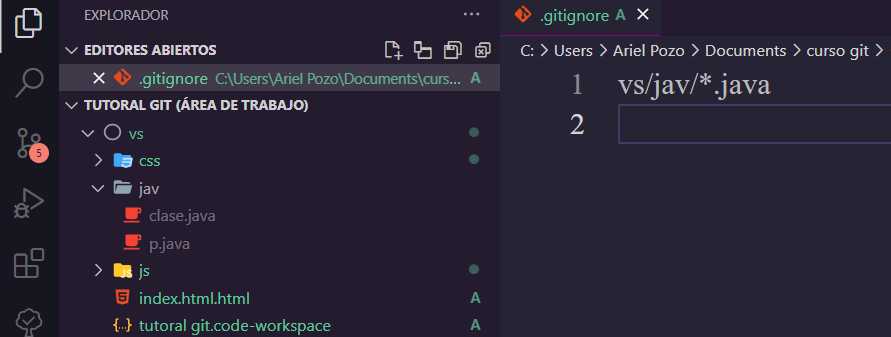
al abrir el archivo .gitignore encontramos que solo ignorara el archivo que recibe por nombre clase.java, y el otro continuara esperando a que sea preparado para ser parte de nuestra área de trabajo esto lo podemos confirmar atreves del control de código fuente



En el se puede observar que el archivo en mencion ya no se encuentra visible, solo se observa que dentro de la carpeta jav esta el archivo que lleva por nombre p.java esto se debe aque ya hace parte de los archivos innorados que se encuentra en .gitignore

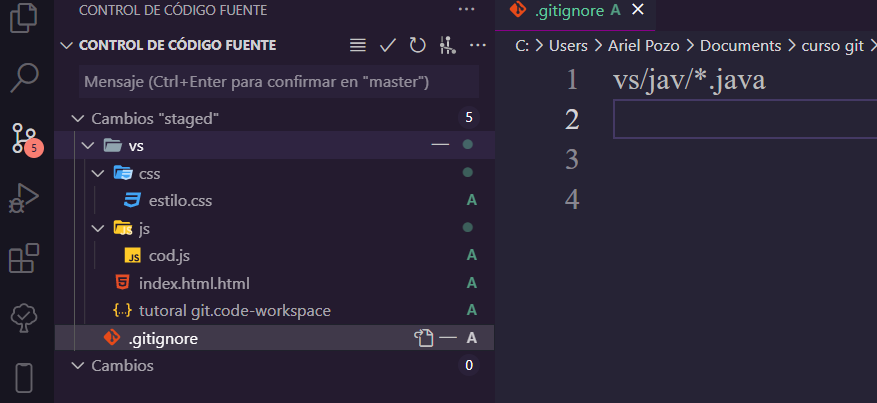
## **Ingnorar Multiples Archivos**

Ahora si lo que necesitamos es ignorar los dos archivos y no estar listando archivo por archivo basta con especificar en el archivo .gitignore la siguiente instrucción \*.java

****

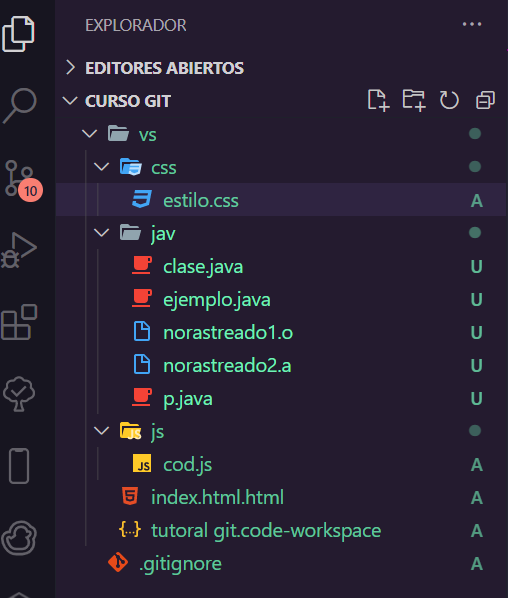
En la imagen anterior se puede observar en el explorador de archivos que la clase.java y p.java no contiene ningun estado esdecir que no contiene ninguna letra (U A M D).

instrucción \*.java indica que ignorara todos los archivos terminado con extensión .java esto lo podemos confirmar atreves del control de código fuente

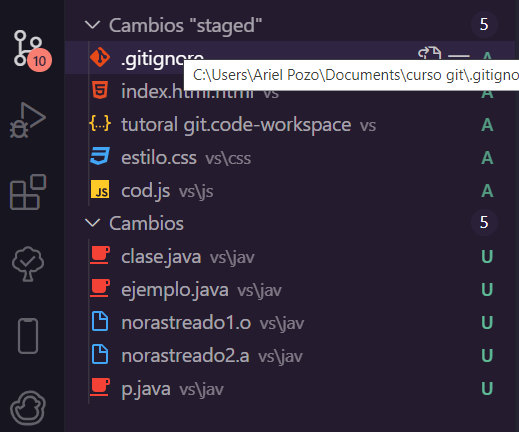


En el se puede observar que en nuestro directorio de trabajo los archivos mencionados no aparecen visible en nuestra área de preparación ni en nuestro directorio de trabajo debido a que ya hace parte de los archivos ignorados.

### **Extensión especifica**

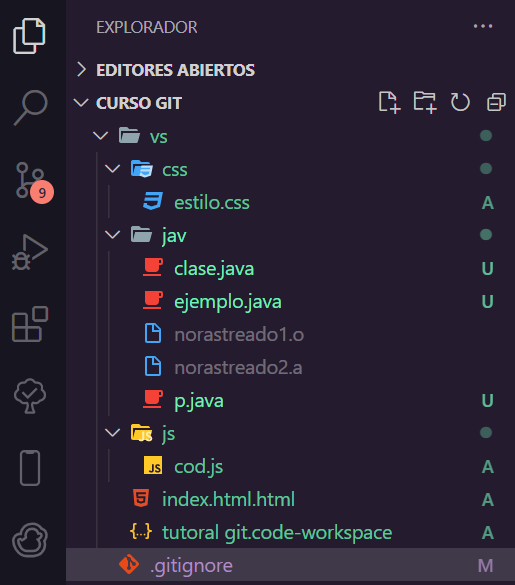
Ahora nuevamente nuestro proyecto tiene esta estructura

Como podemos observar dentro del directorio jav tenemos 5 archivos tres de ellos terminados en extensión .java y dos mas terminados con extensión o Y a respectivamente, todos estos archivos están inicializados por git pero aun no están siendo rastreado esto lo podemos confirmar al dirigirnos a nuestro control de código fuente veamos:

Se puede observar que nuestro directorio de trabajo contiene los 5

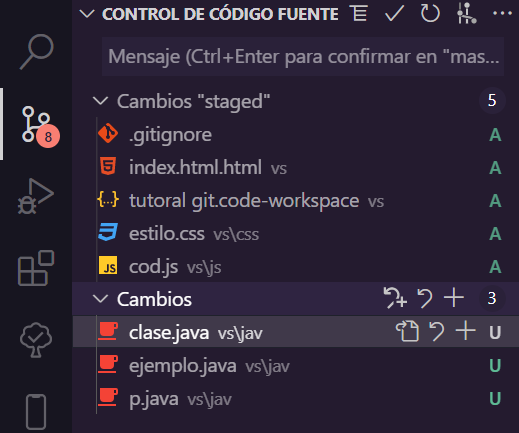
archivos y que los mismos se encuentran sin seguimiento ya que se encuentra representado por la letra U

Ahora vamos a suponer que por algún motivo necesitamos ignorar los archivos que llevan por nombre norastreado1.o Y norastreado2.a

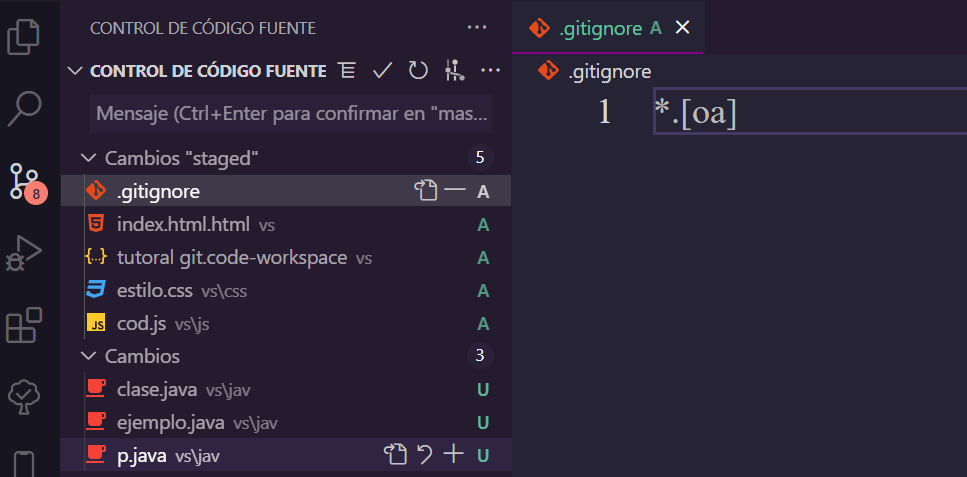
En la nueva imagen podemos ver que en el visualizador de archivos los archivos

mencionados ya se encuentran ignorados ya que no contiene ninguna letra que identifica el estado precisamente de esos archivos esto lo podemos confirmar en nuestro control de código fuente veamos.

## 

****

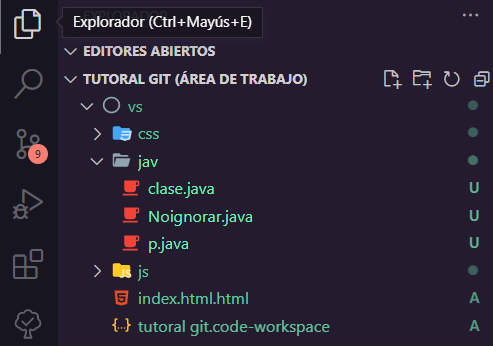
Ahora podemos visualizar que nuestro directorio de trabajo solo se encuentran los archivos con terminación .java y ya no se ven de manera visual los archivos antes mencionados, esto se debe a que están siendo ignorados porque le hemos dicho a nuestro .gitignore que ignore todos los archivos terminados en a Y o veamos :



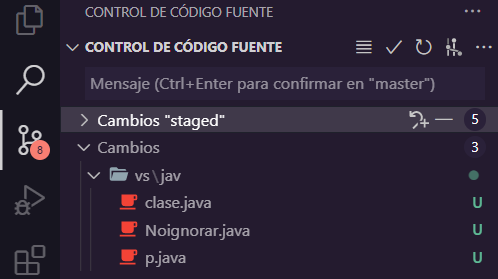
Como ya indicamos anteriormente la línea 1 me indica que ignorara todos los archivos que se encuentre con terminación .o Y .a

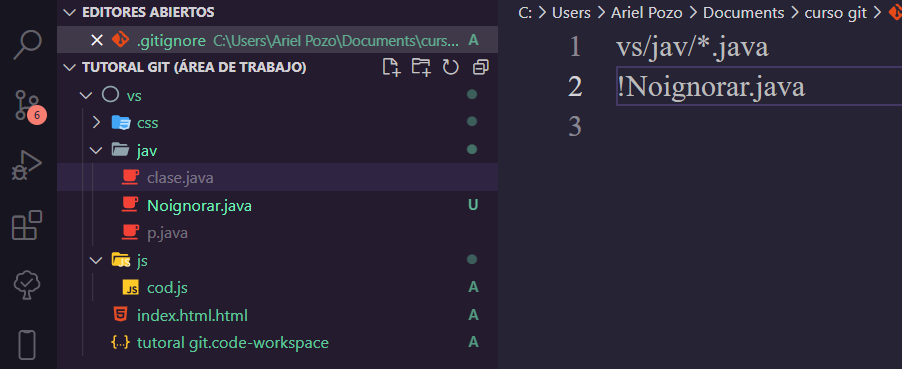
## **Negación de un Archivo Ignorado**

Vamos a gregar otro archivo a nuestro proyecto con el nombre Noignorar.java para dejar claro como funciona veamos

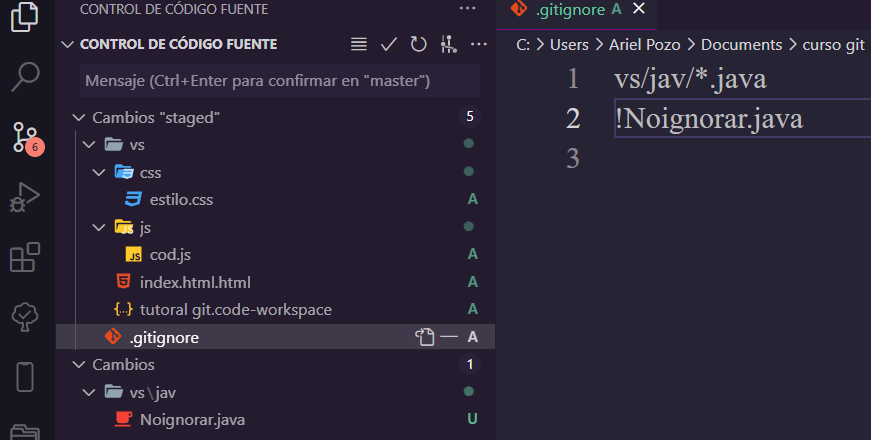


Nuevo archivo



Como se puede observar se a creado un nuevo archivo dentro del directorio jav y los archivos que se encuentran contenidos en el aun no estan en el area de preparacion vamos abrir nuestro archivo .gitignore y lo editaremos para posteriormente explicarlo

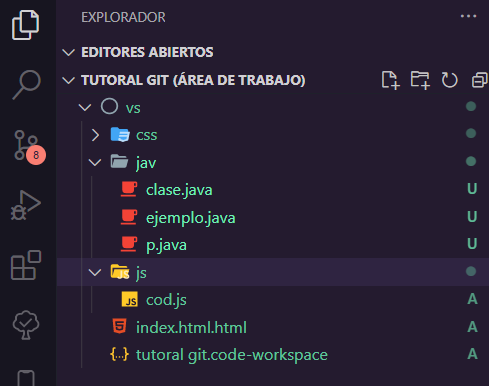
En la imagen anterior se puede observar que nuestro archivo .gitignore contiene 2 lineas de codigo, en la primera linea me indica que va ignorar todos los archivos que terminen con la extension .java y en la segunda linea me indica que el simbolo ! negara todo lo que venga después de el, en este caso la instrucción !Noignorar.java indica que ese archivo no se debe ignorar pese a que en la línea anterior se están ignorando los archivo terminados en .java. si miramos el explorador de archivos nos dice que los archivos clase.java y p.java no contienen un estado para ellos mismo esdecir que no contiene ninguna letra (U A M D). indicandome que git no se va inicializar para ellos, adiferencia del archivo que lleva por nombre Noignorar.java que el estado de su archivo con tiene la letra (U) haciendo referencia que git esta inicializado para ese archivo pero que aun no esta enseguimiento. esto lo podemos confirmar atreves del control de código fuente



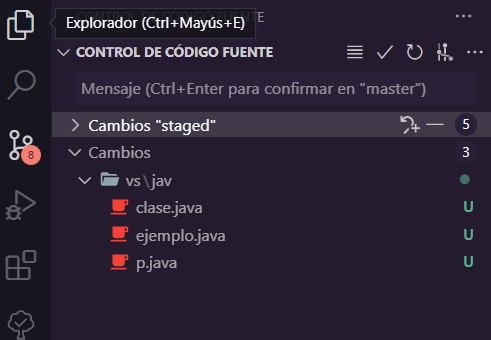
En el se puede observar que todos los archivos que sea diferente a Noignorar.java no se visualizaran en el directorio de trabajo como en el área de trabajo, indicando que ya hace parte de archivos ignorados.

**Ignorando todos los archivos de un directorio**

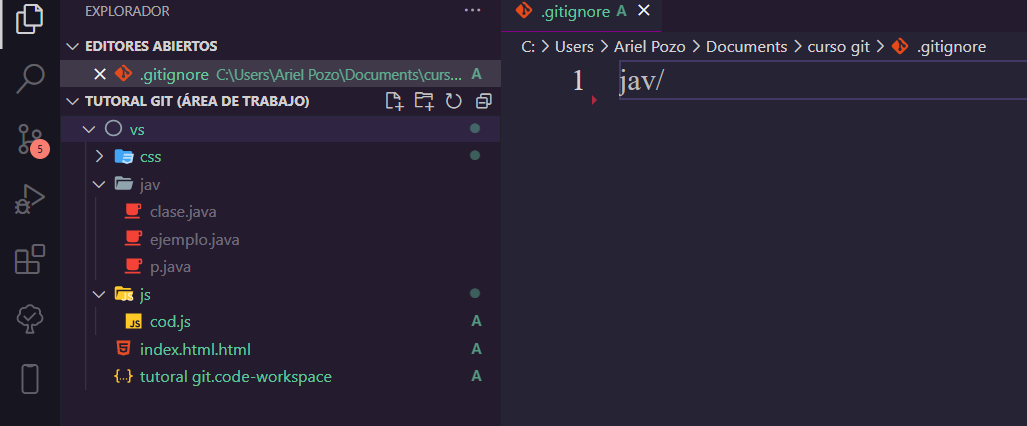
Si queremos que git ignore todos los archivos de una carpeta solamente especificamos el nombre de la carpeta terminado con el / como se indica a continuación.



Asumimos nuevamente que nuestro proyecto tendra la estrutura de la imagen anterior, en el observamos que el directorio jav contiene 3 archivos todos con extension .java y que todos se encuentrar rastreado pero sin seguimiento ya que el estado de sus archivos asi lo indican con la letra (U), y asi lo confirma nuestro control de codigo



Ahora procederemos a ignorar todo el contenido de su directorio.



Sin mucho que decir en la linea 1 se ejecuta la instrucion indicando que todos los archivos

Que estan dentro del directorio jav seran ignorados si miramos el explorador de archivos nos dice que los archivos (clase.java ejemplo.java p.java) no contienen un estado para ellos mismo esdecir que no contiene ninguna letra (U A M D). indicandome que git no se va inicializar para ellos. esto lo podemos confirmar atreves del control de código fuente

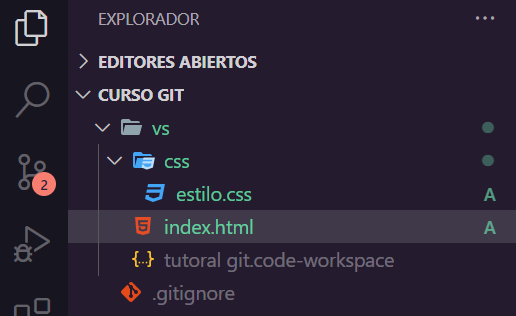
Tal como lo indica la imagen anterior el directorio de trabajo

Me indica que tiene 0 archivo visible para mostrar

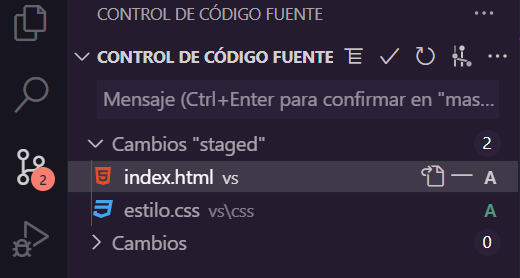
# **Ver los Cambios Preparados y No Preparados**

Si el estado de los archivo no proporciona una información completa ,y lo que quieres ver es una información mas exacta de lo qué ha cambiado, no solo cuáles archivos lo han hecho si no ¿Qué has cambiado, pero aun no has preparado? y ¿Qué has preparado y está listo para confirmar? A pesar de la integración de git en visual studio responde a estas preguntas de forma muy general listando el nombre de los archivos, la integración de git en visual studio también proporciona una opción que te muestra las líneas exactas que fueron añadidas y eliminadas línea por linea, es decir, el parche. Veamos con algunos ejemplos.

Asumiendo que nuestro proyecto tiene la siguiente estructura:

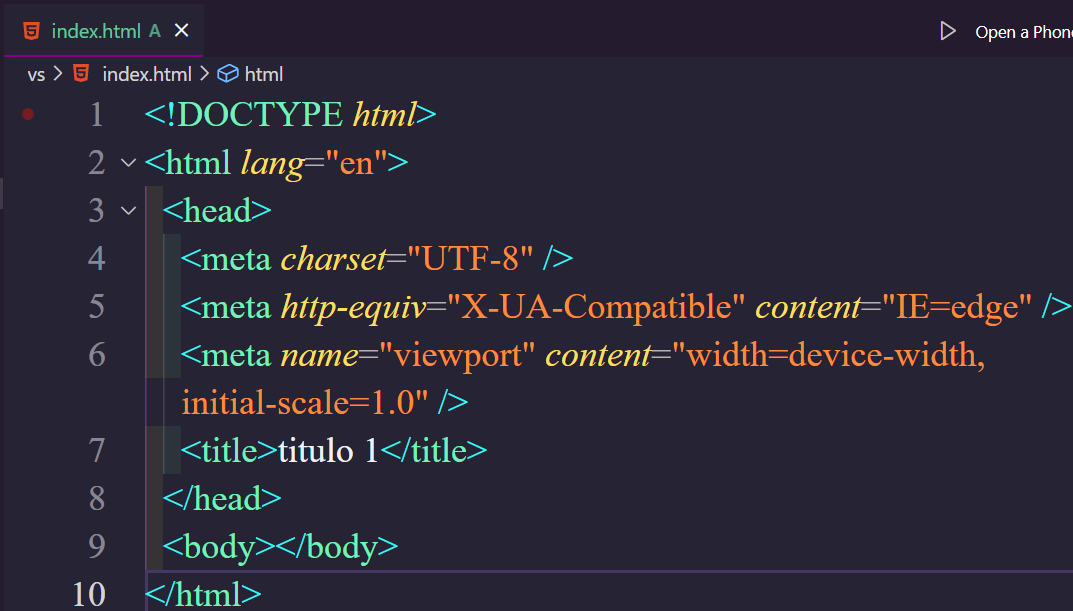


En el se observa los dos archivos el archivo index.html que se encuentra en el directorio raiz y el archivo css que se encuentra en el directorio de el mismo nombre y ambos se encuentra en seguimiento indicado atraves de las letras que identifican el estado de sus archivos, esto lo podemos comprobar atravez del control de codigo fuente.

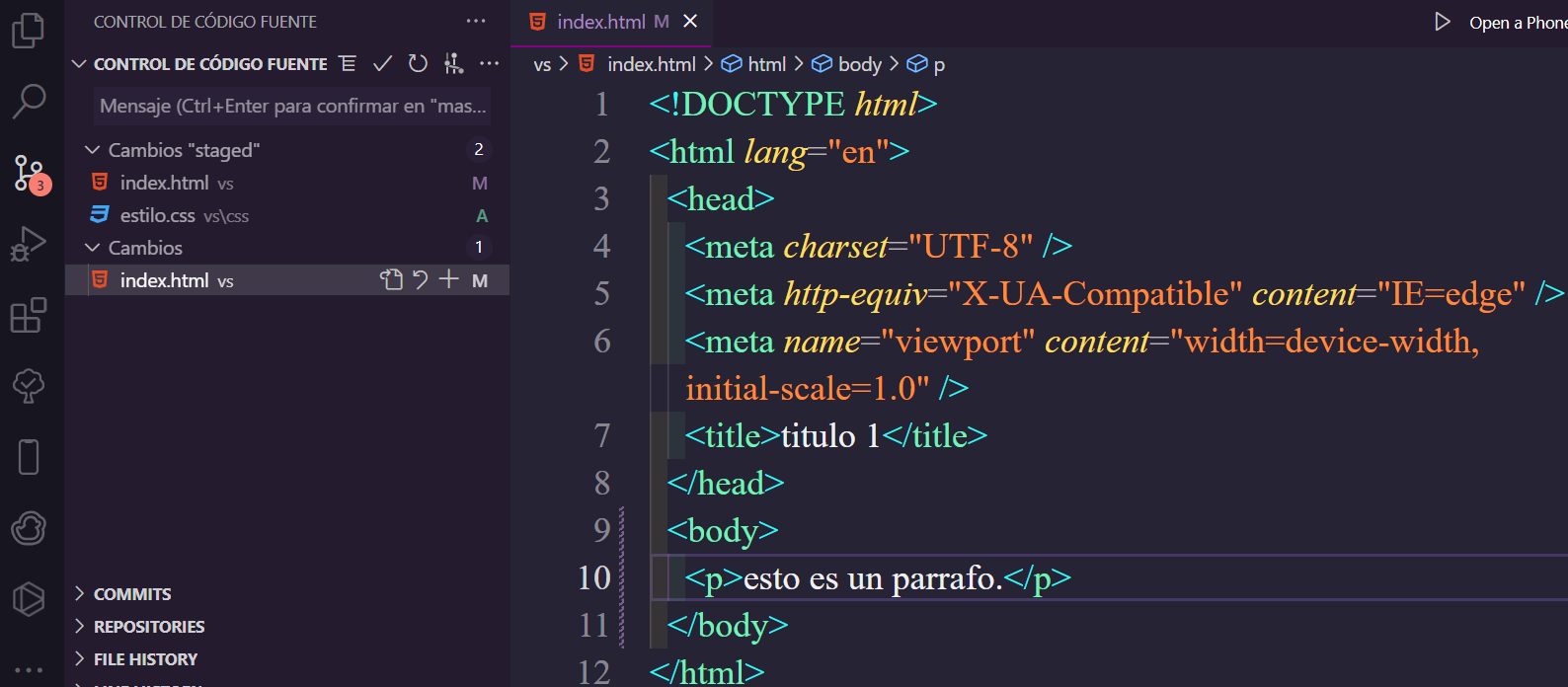


## Cambios No Preparados

partiendo que el archivo index.html tiene la siguiente estructura



Vamos a editarlo

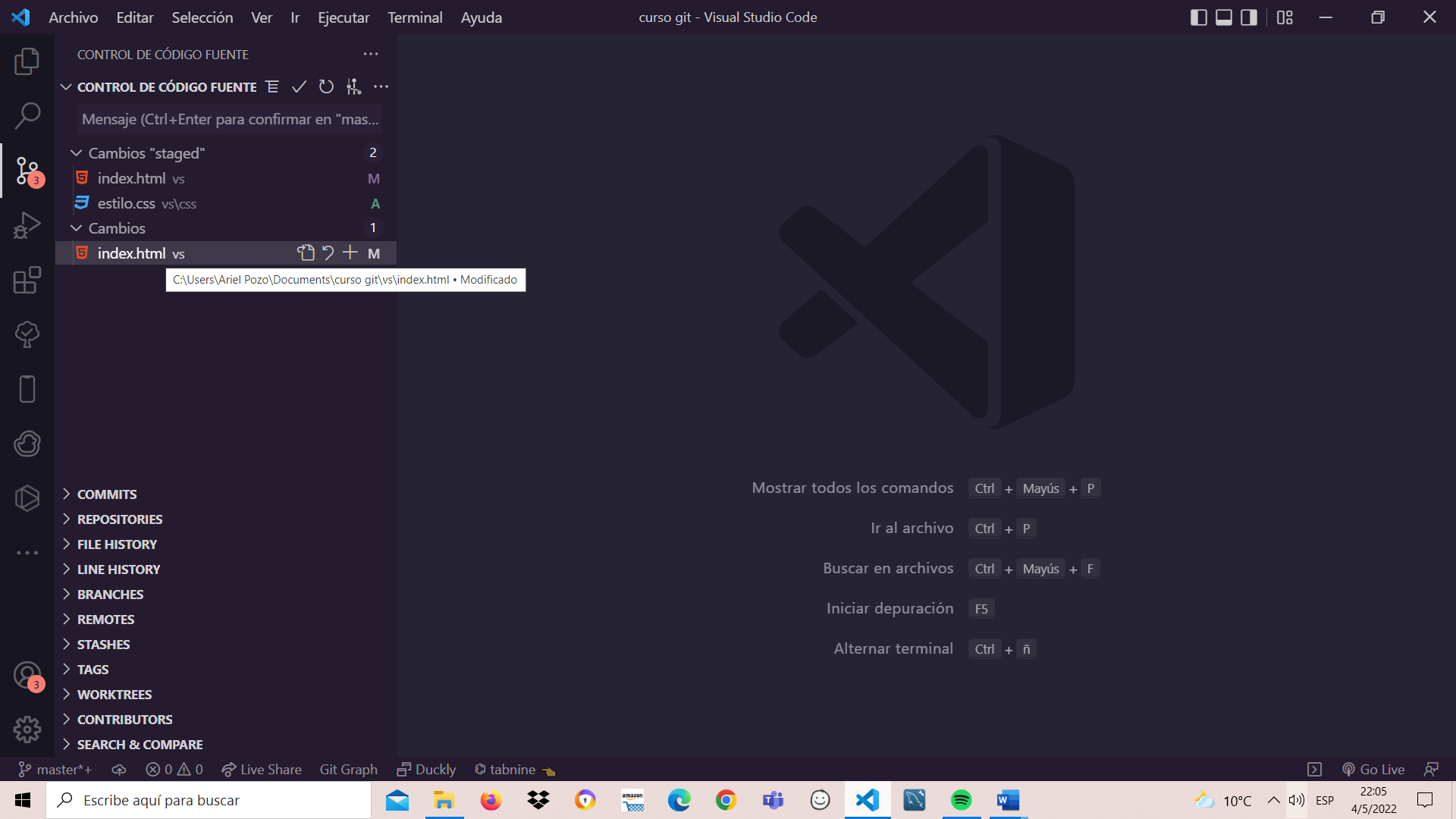


En la segunda imagen podemos ver que hemos agregado en la linea 10 una nueva etiqueta con relacion a la primera imagen, en el control de codigo fuente podemos ver

En nuestro directorio de trabajo que el archivo index.html a sido modificado ya que su estado se encuentra en M indicandome precisamente que a sufrido una modificacion hasta aquí no ahí nada raro imaginemos que tenemos en cada archivos ciento de lineas de codigo y necesitamos saber las diferencias que ahí entre el archivo modificado y el que se encuentra en nuestra area de trabajo

¿pero como hacemos esto ? veamos:

Primero tenemos que dirigirnos al control de codigo fuente y dentro de nuestro directorio de trabajo seleccionar el archivo que queremos de la siguiente manera



Al darle clic al archivo en cuestion solo sobre los limites de la franja verde se abrira la venta del archivo en 2 columnas indicando el archivo que se encuentra en el area de trabajo y el archivo modificado, para entenderlo mejor veamolo en un ejemplo.

