# Proyecto web blog con astro

## Entorno de desarrollo:

para iniciar a trabajare en astro es necesario tener instalado algunas cosas como:

-node y npm

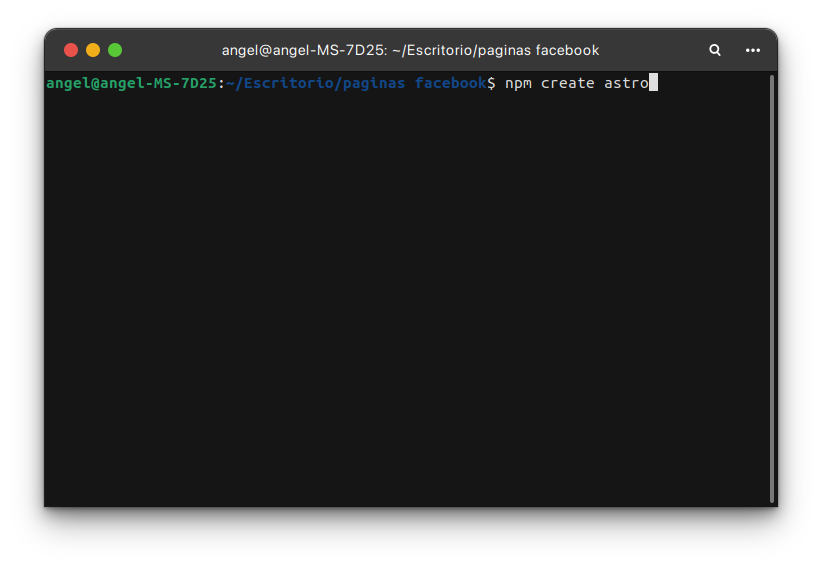
-editor de codigo, en este caso vscode

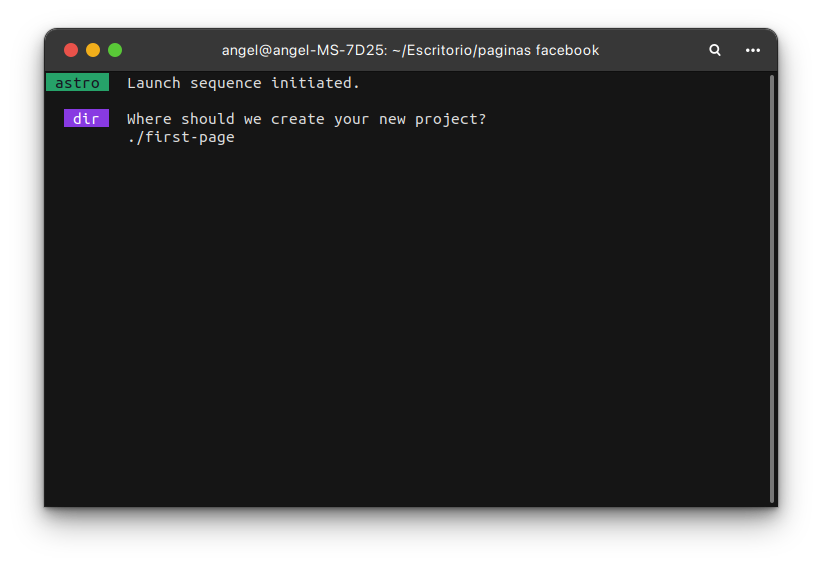
-git

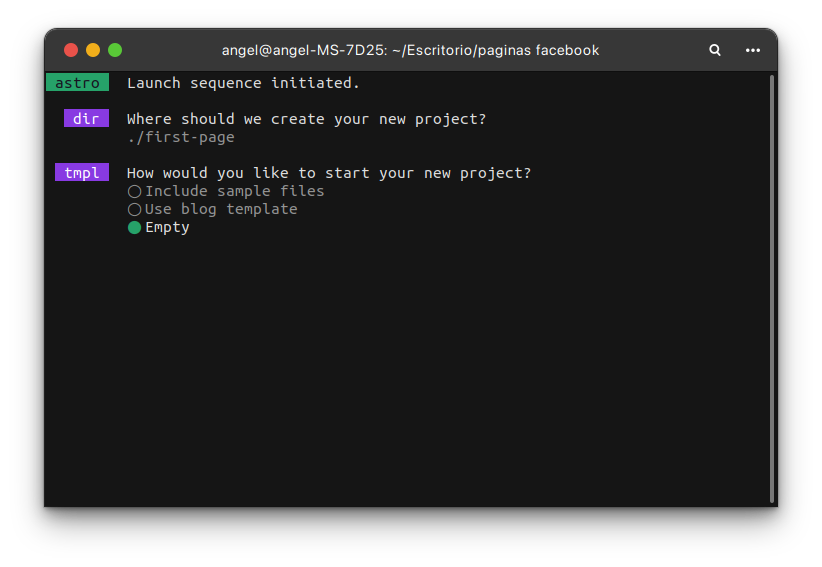
## Configuración inicial:

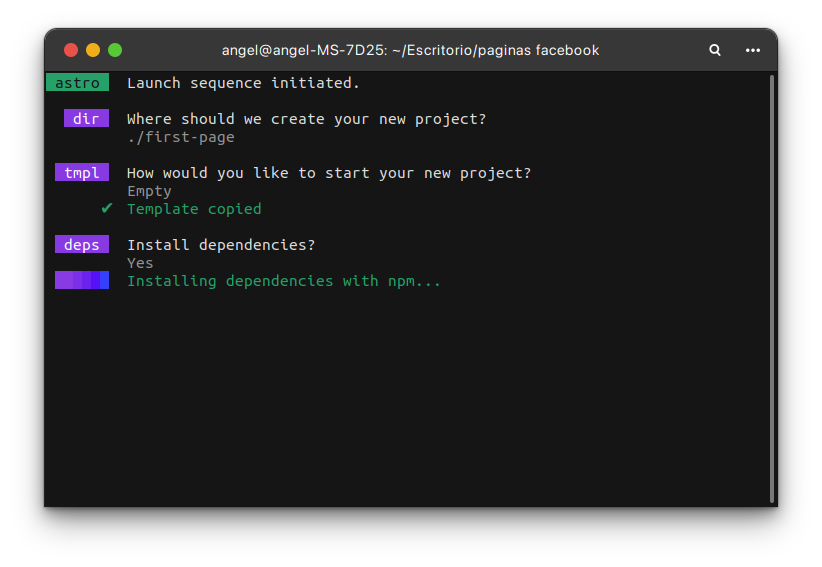
para iniciar con la creacion de un proyecto en la ruta ya establecida en astro es necesario crear dicho proyecto mediante el siguiente comando:

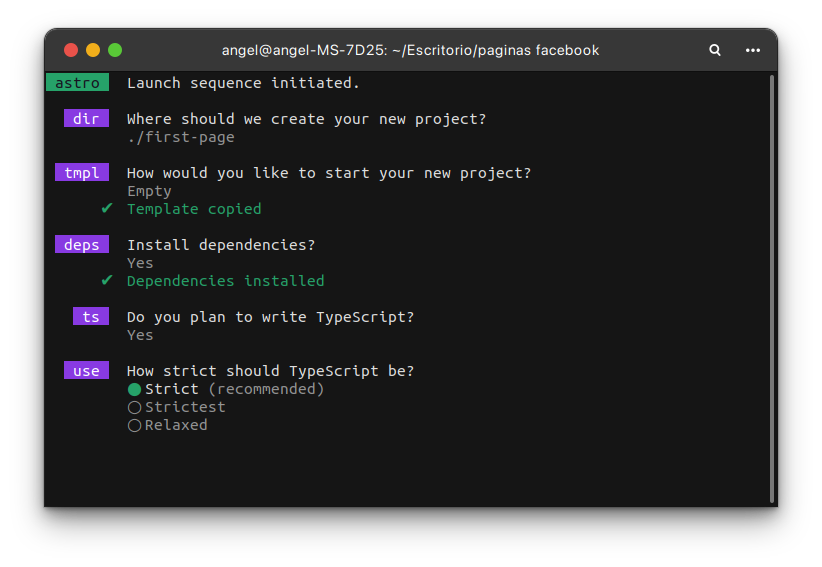
npm create astro

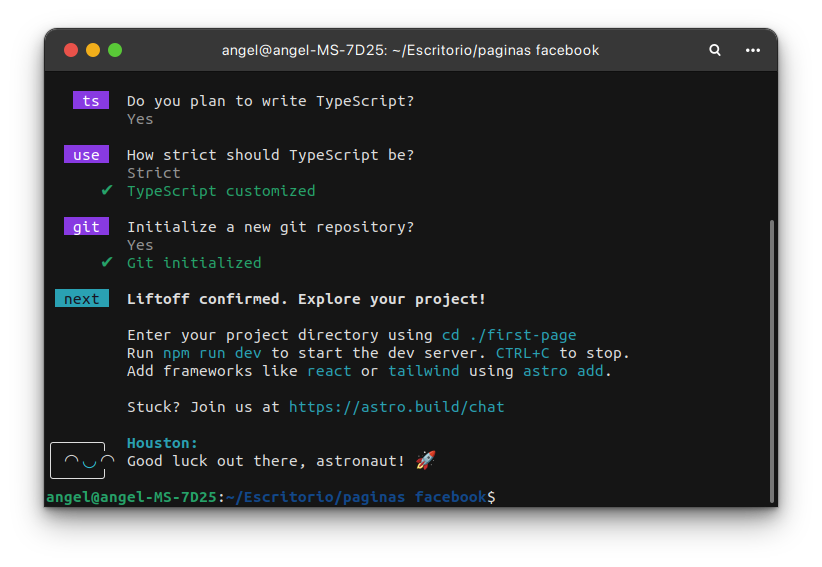
les preguntara como quieren que se llame la carpeta que contendrá el proyecto:

despues de eso les preguntara que si es necesario incluir un ejemplo, un blog o dejarlo vacio, queda a criterio pero es recomendable dejarlo vació para entender mejor la logica.

Instalamos las dependencias

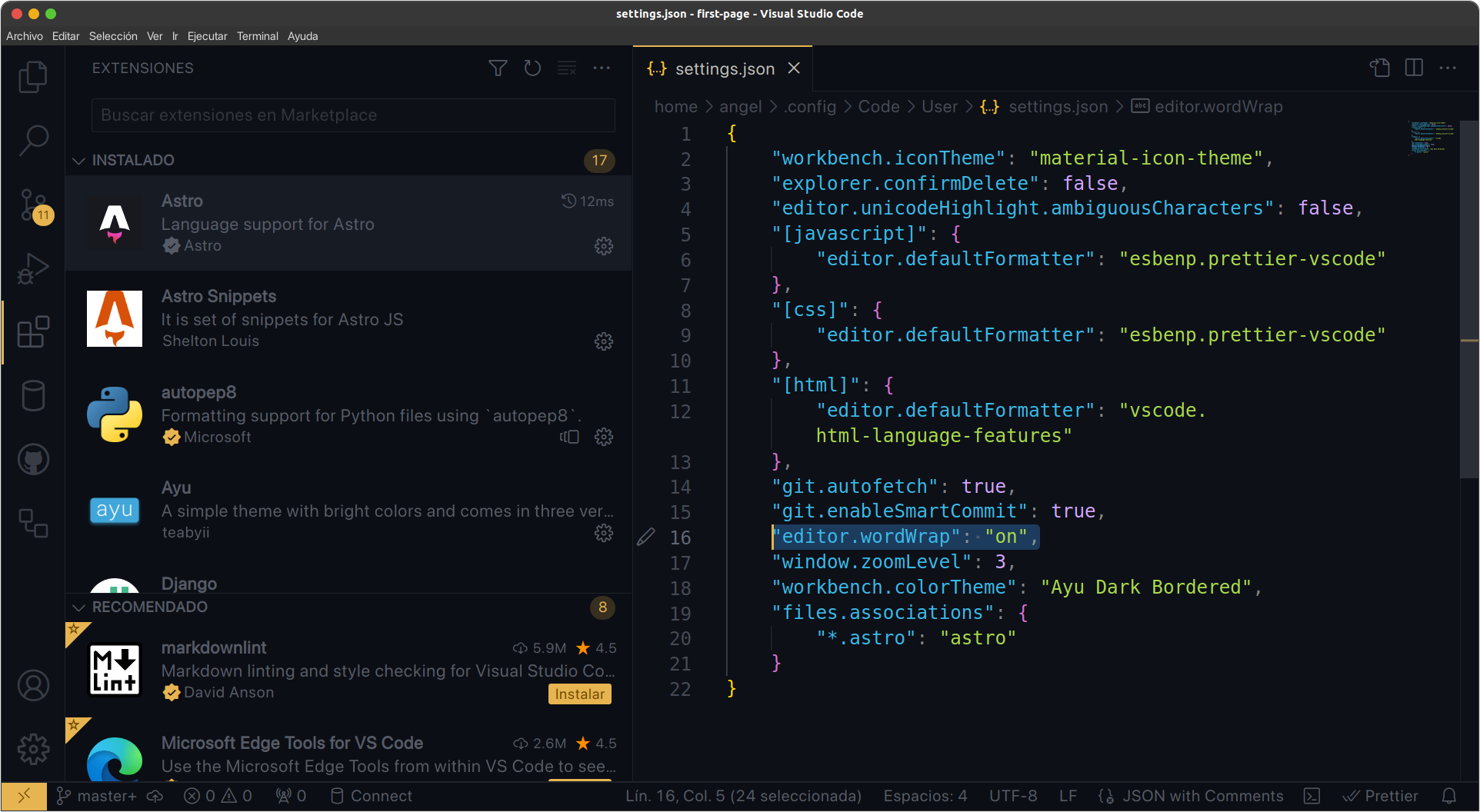
marcamos el uso rcomendado de typescript para que en un futuro que el proyecto cresca nos sirva de ayuda.

Iniciamos nuestro repositorio y ya desde este momento podremos empezar a realizar nuestro proyecto con libertad.

Abrimos el proyecto desde vscode y podemos ver la estructura del proyecto el cual en mayor medida utilizaremos la carpeta de public y el src.

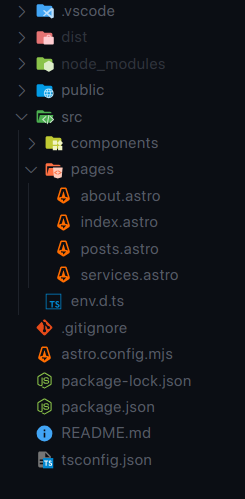
Nota:

para que nuestro vscode se adapte de mejor manera recomendamos añadir las estenciones siguientes ademas de la configuracion de json en el user json:

el wordwrap nos ayudara a formatear el codigo de acuerdo al tamaño de la pantalla de nuestro editor de manera dinamica.

## Paginas en Astro:

para que nosotros creamos nuevas paginas en astro es muy sencillo como el simple hecho de agregar un nuevo archivo en la carpeta **src/pages** y automáticamente ya existiría en las rutas una nueva pagina la cual podemos visitar.

Cabe recalcar que una estructura típica en los archivos de astro estarán conformados por lo siguiente.

*---*

const title = "Mi primera pagina"

*---*

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" type="image/svg+xml" href="/favicon.svg" />

<meta name="viewport" content="width=device-width" />

<meta name="generator" content={Astro.generator} />

<title>{title}</title>

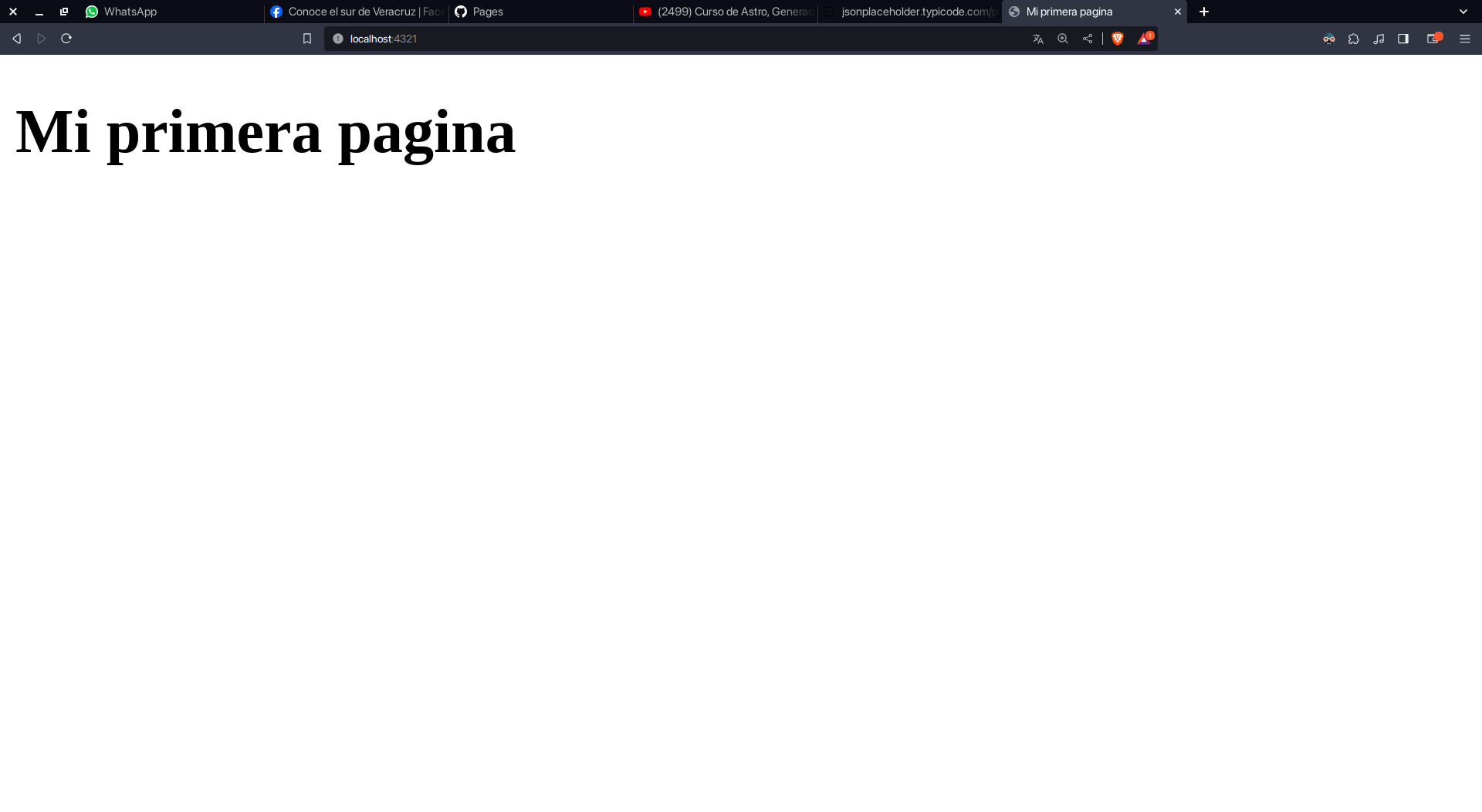
</head>

<body>

<h1>{title}</h1>

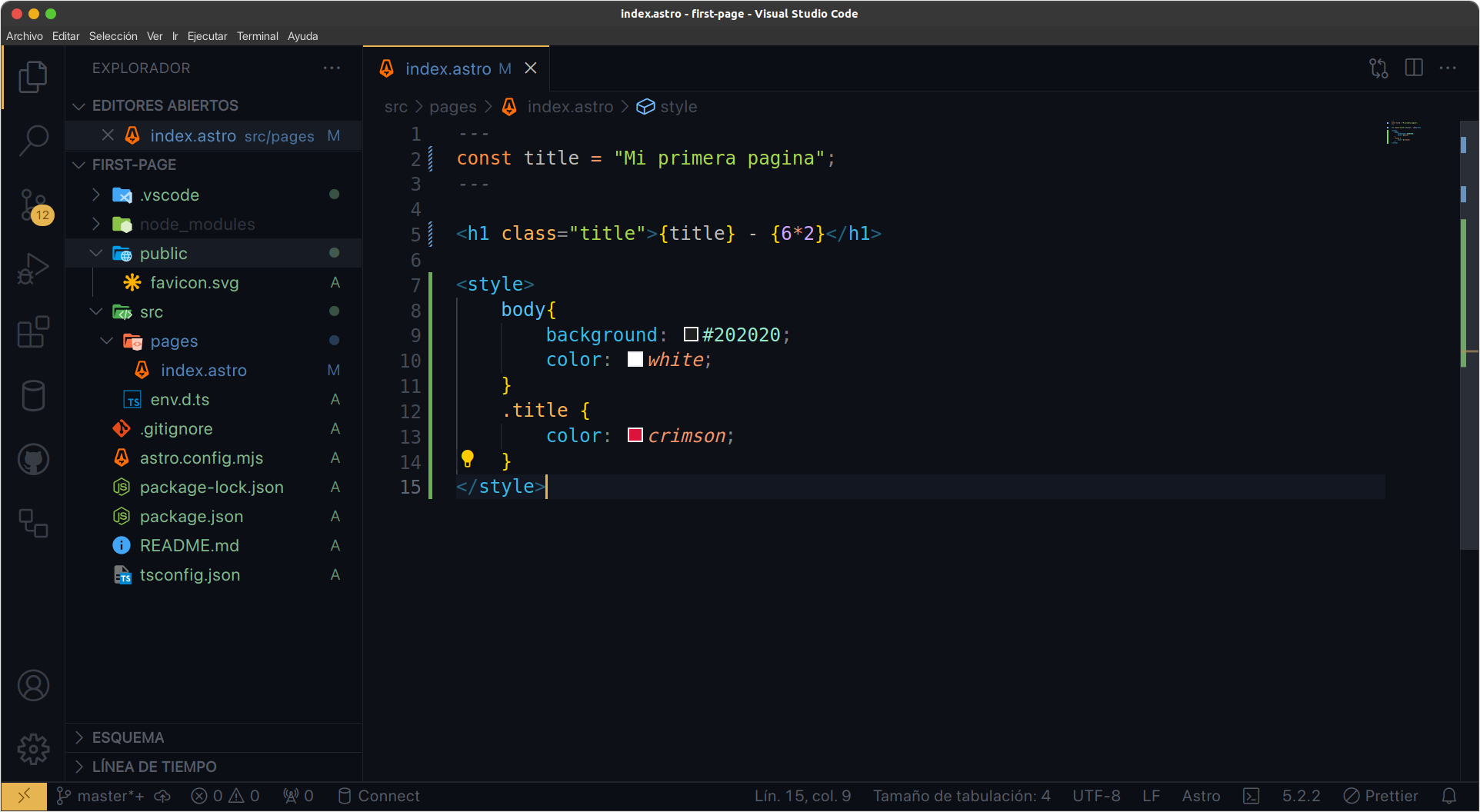
</body>

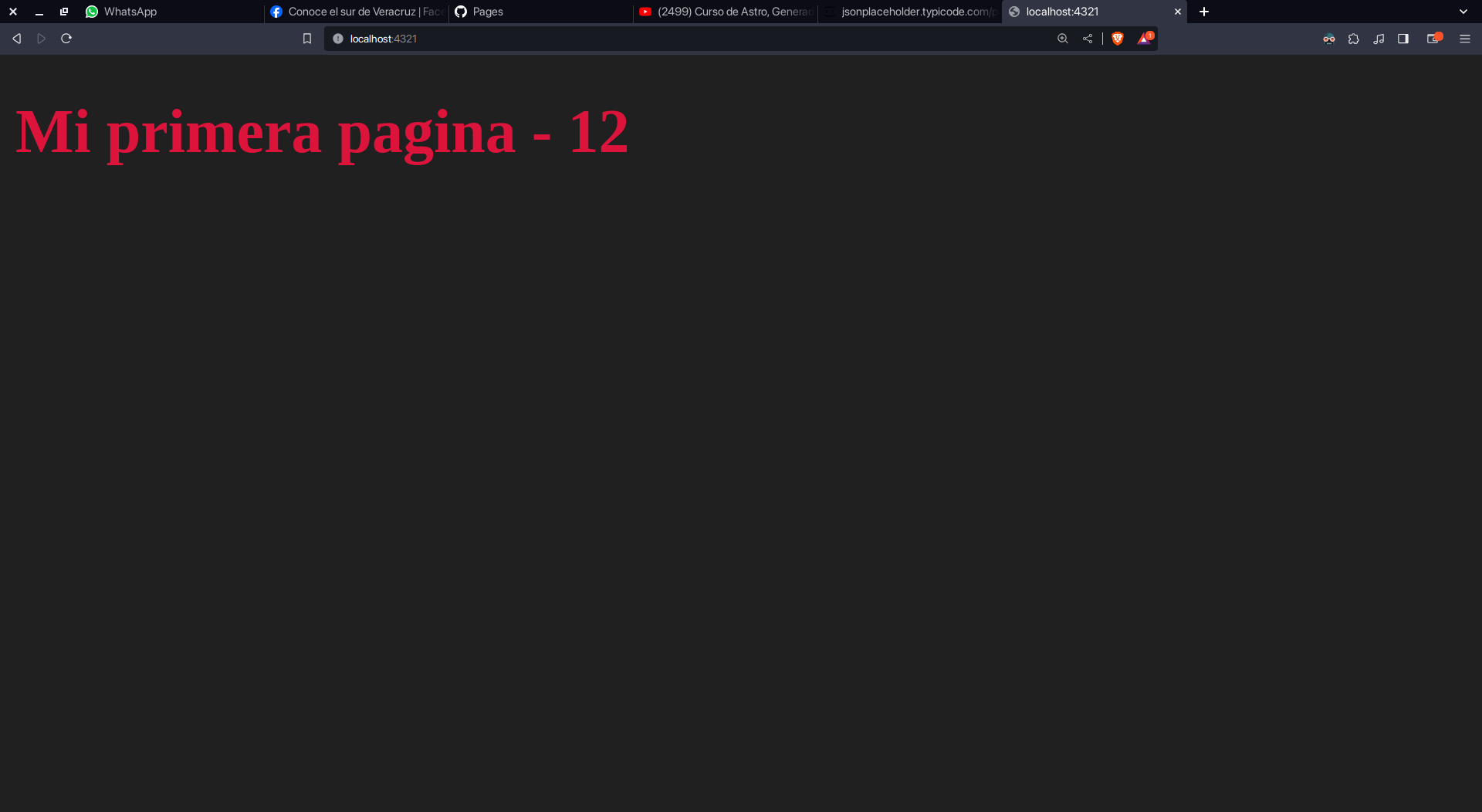
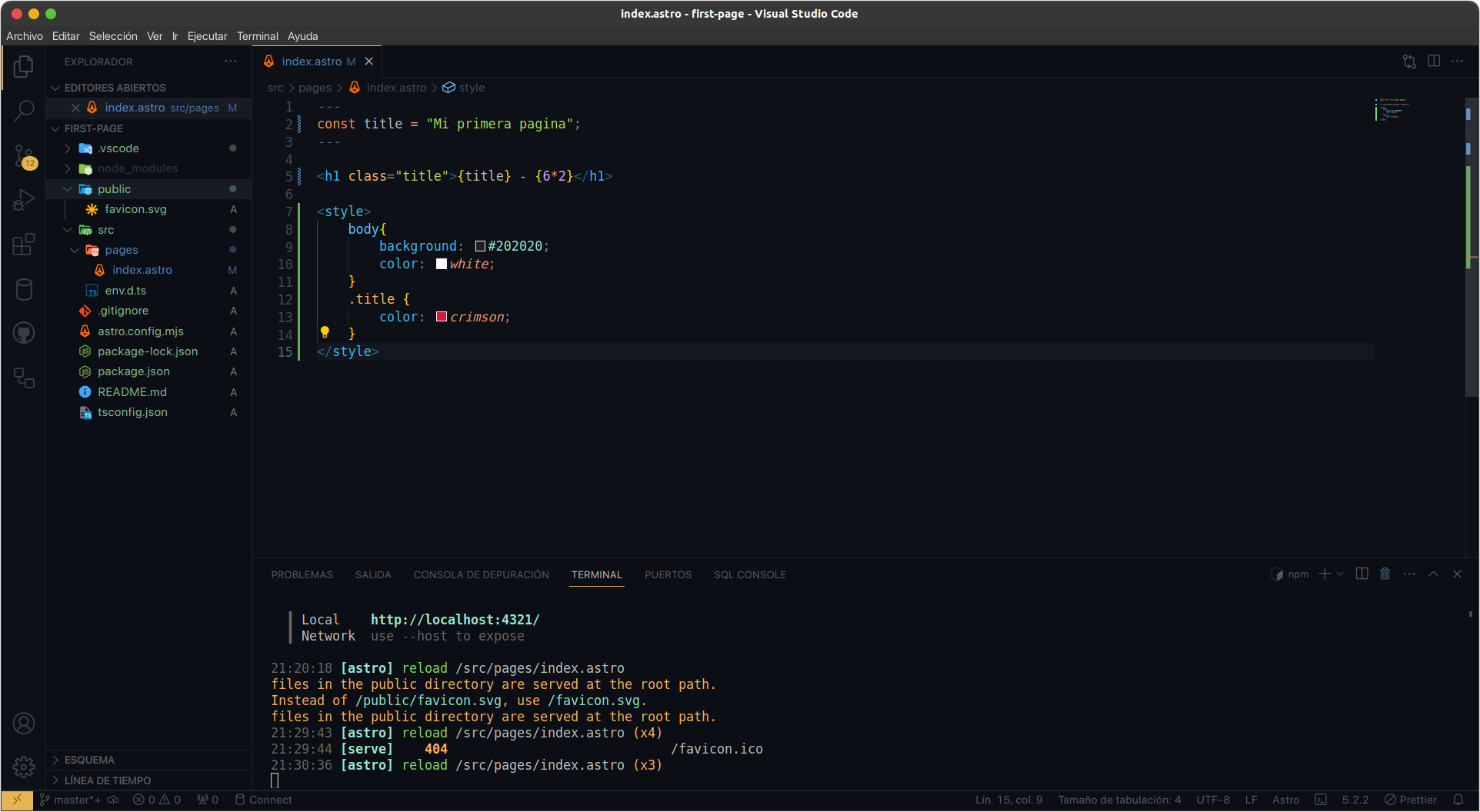
</html>

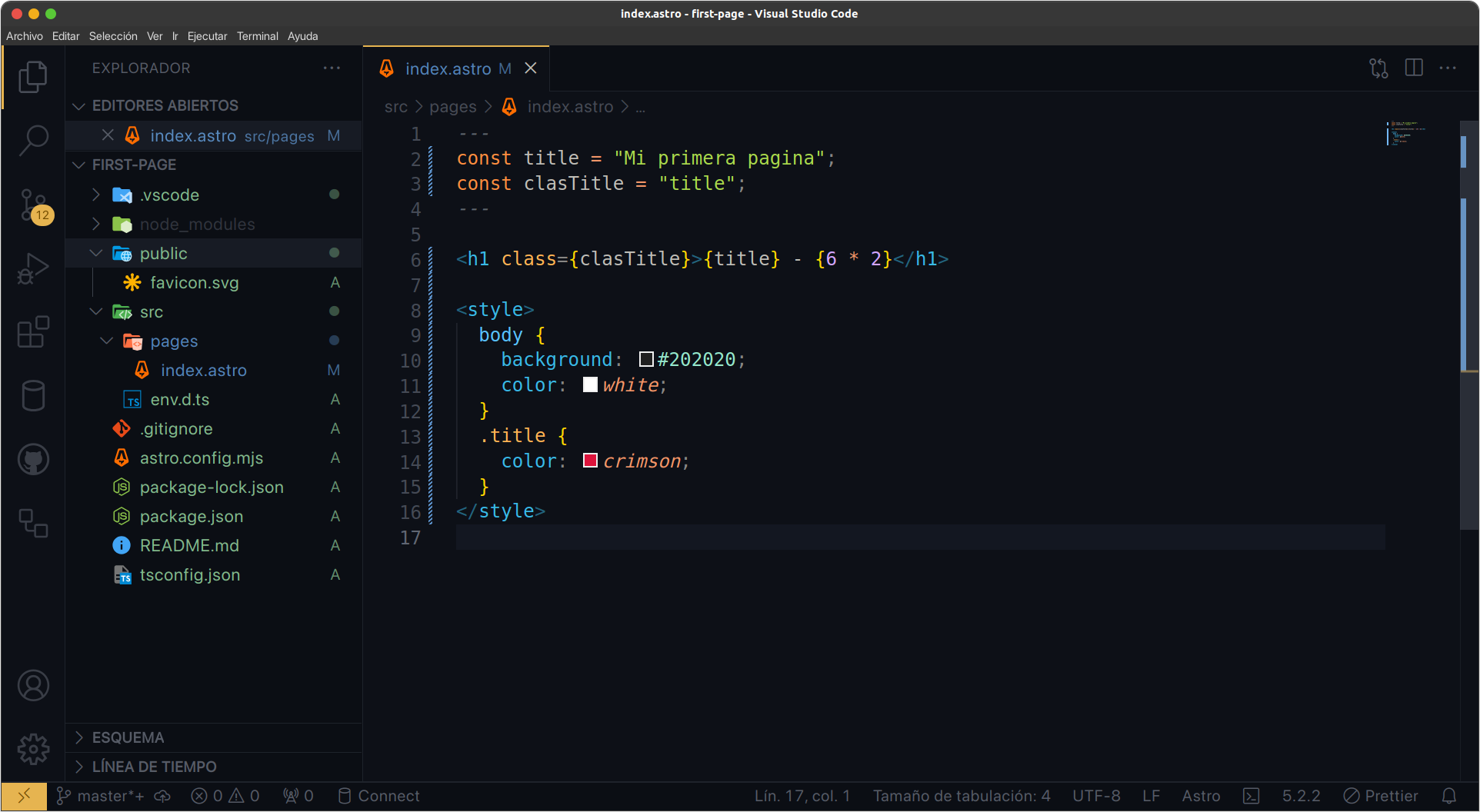
entre “---” el codigo que usara logica de js para la conexión de datos el cual la estructura de astro podra leer entre llaves y se vera de la siguiente manera:

## Sintaxis JSX en astro build & fetch:

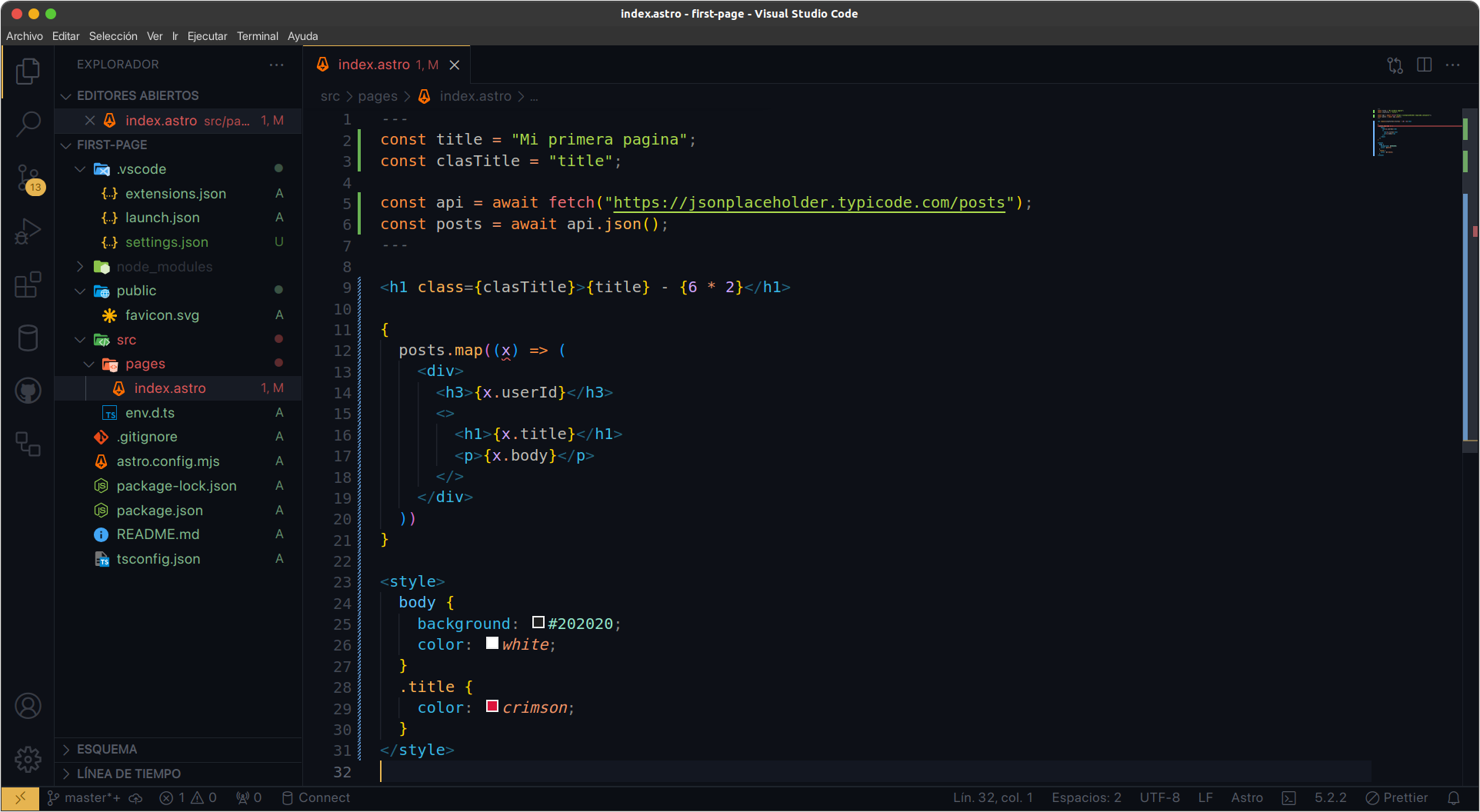
Hay que recalcar que astro puede interpretar sintaxis de javascript que posteriormente el framework traspasara a html puro pero que en el desarrollo nos ayudara mucho para mostrar informacion, por ejemplo usando las llaves {}.



como puedes ver al inicio se utiliza codigo de js pero al final astro convierte todo a contenido estatico de html con el fin de volver ligero el contenido.

Podemos ademas de eso utilizar la sintaxis para añadir las clases directamente desde una constate.

Otra cosa que podemos hacer y que nos ayudara de grana manera es la obtención de datos desde una api y recorriendolo en nuestra pagina de manera rapida y sencilla por ejemplo utilizando la api de <https://jsonplaceholder.typicode.com/posts>

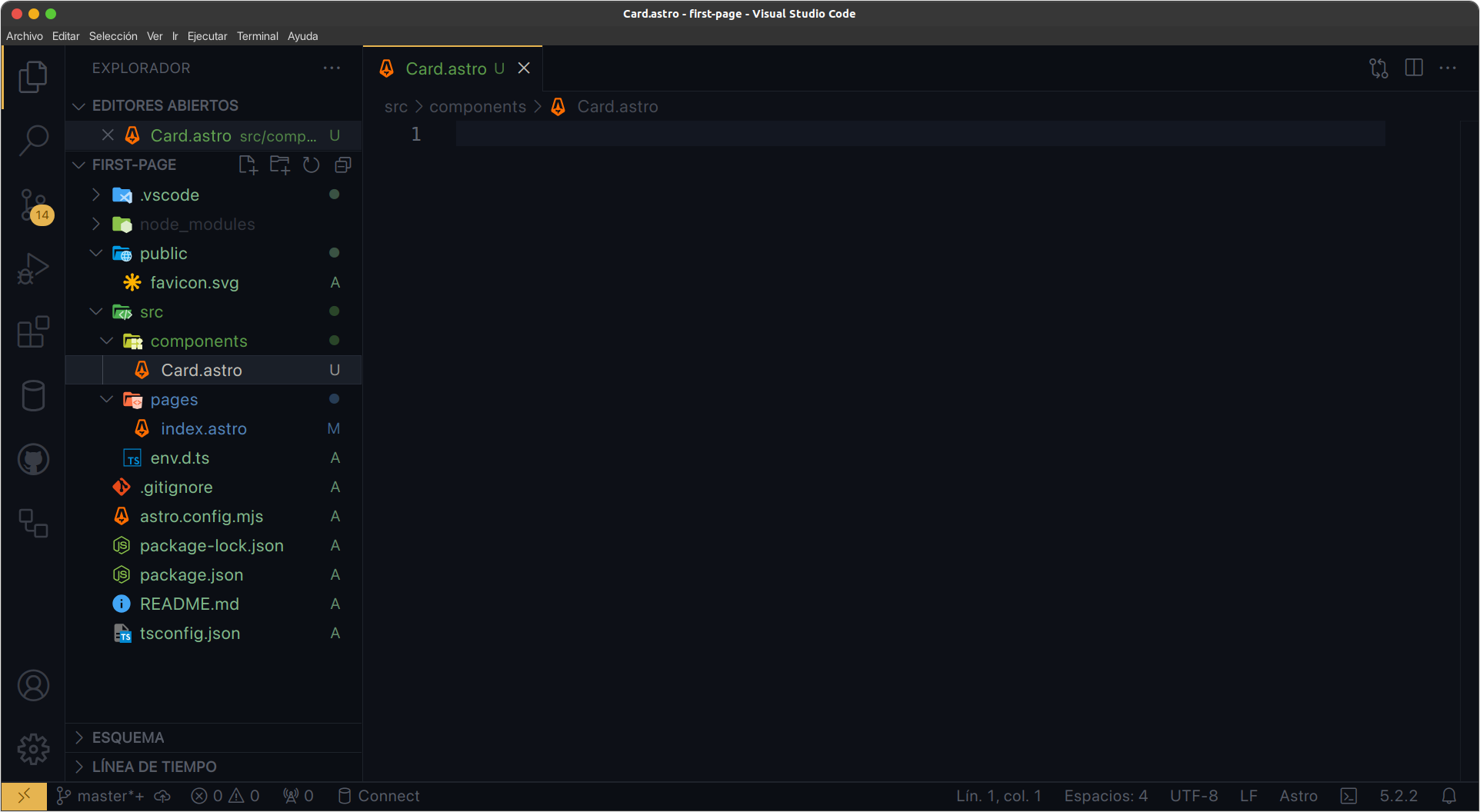


de esta forma podremos recorrer datos que al final Astro va a pasar a simple y crudo html.

## Astro components

En astro se utiliza una filosofia de reutilizacion de codigo que nos facilitara de gran manera la creacion de las paginas webs que estamos creando así como otros frameworks de js, para eso utilizaremos algo que conocemos como componentes, eso de la siguiente manera.

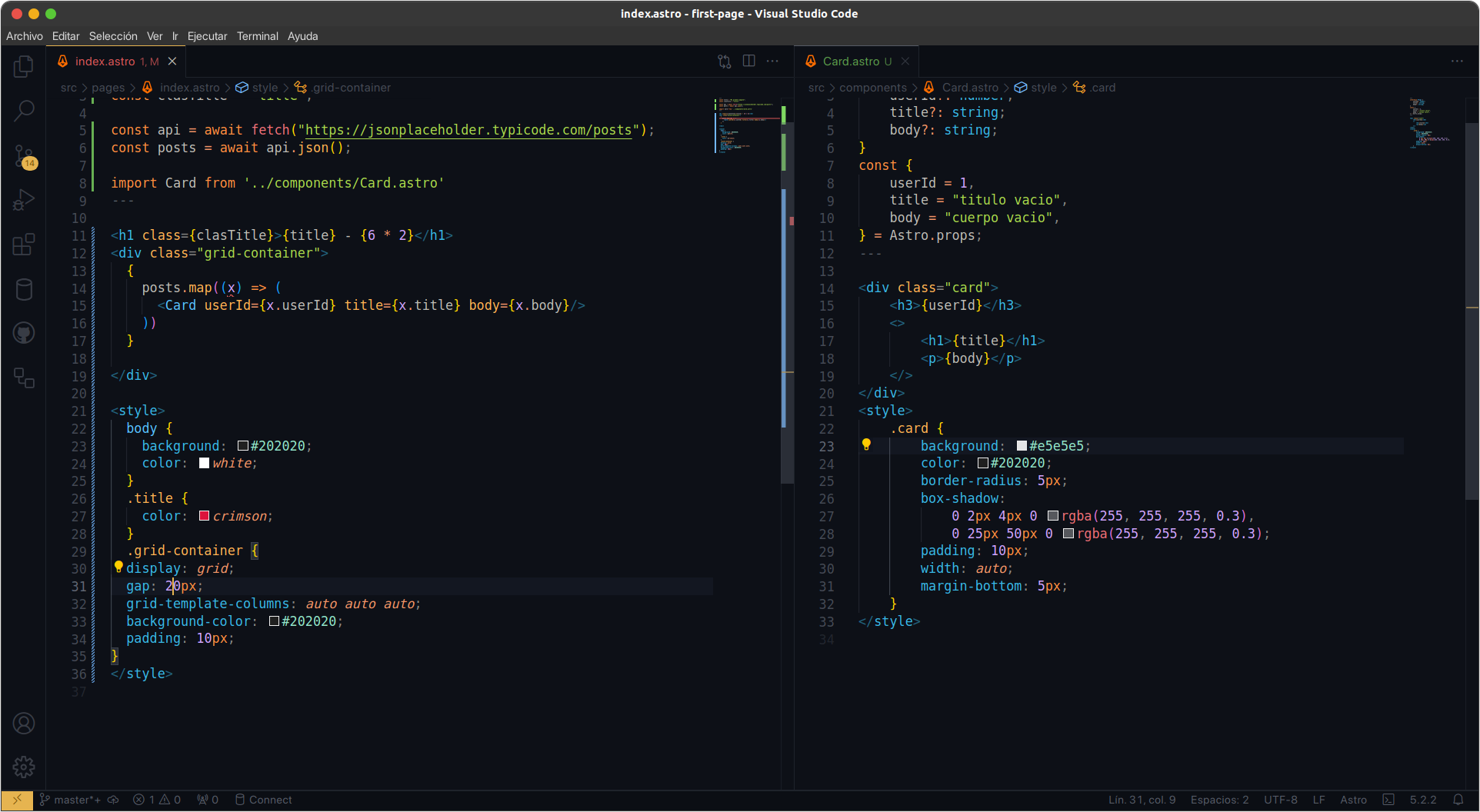
Primero que nada hay que localizarlos aparte de todo para tener un orden a la hora de crear nuevas porciones de código.

Así es como se agrega el componente a una nueva carpeta. Por conveniencia, solemos nombrar la carpeta con la primera letra en mayúscula para identificar que se trata de un componente y con la extensión ".astro". Desde aquí, podemos agregar diversos datos y llamarlo desde distintas páginas de la siguiente manera:

import Card from '../components/Card.astro'

En la página, utilizando la etiqueta "---", podemos llamar al componente. En primer lugar, le asignamos un nombre y luego especificamos su ubicación. En el componente, podemos definir el estilo que deseamos que se aplique a lo que vamos a mostrar. Además, es importante destacar que el componente será dinámico en cuanto a su contenido, lo que significa que la estructura visual será la misma o variará en algunos aspectos, además del contenido. Por lo tanto, podemos incorporar opciones dinámicas utilizando algo conocido como "props", lo cual nos ayudará a diseñar el código que utilizaremos.

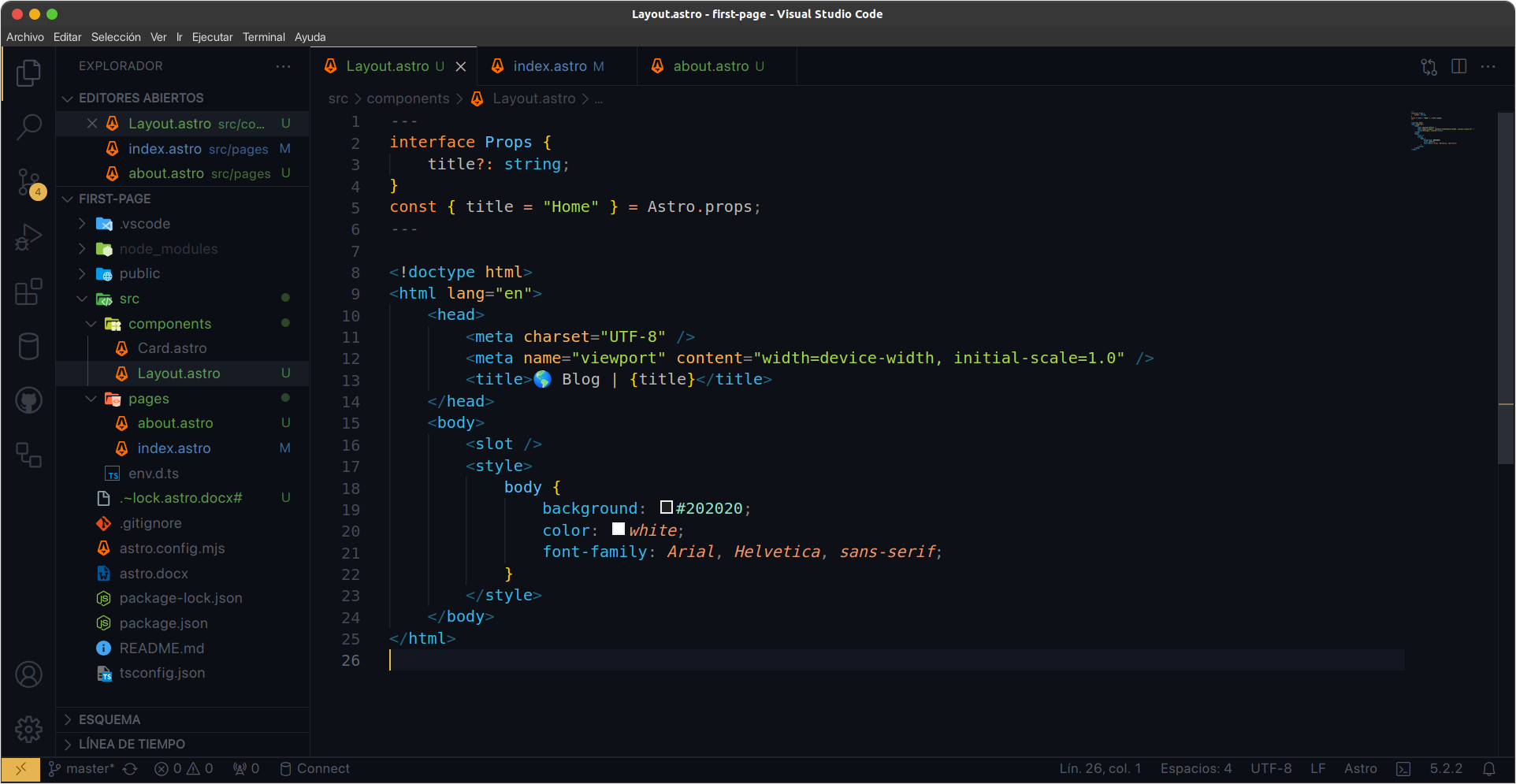
Primero, observemos:

  
Dentro del componente, se especifica el tipo de variables que se utilizarán mediante el uso de TypeScript. Al mismo tiempo, se les asigna la capacidad de admitir valores nulos, y se llenan estas constantes con los valores proporcionados por las propiedades (props) de Astro. Si no se asigna ningún valor, se les asigna un valor predeterminado. Una vez que estas constantes están definidas, se pueden incorporar en el componente "card" y aplicarles un estilo propio que se agregará posteriormente.

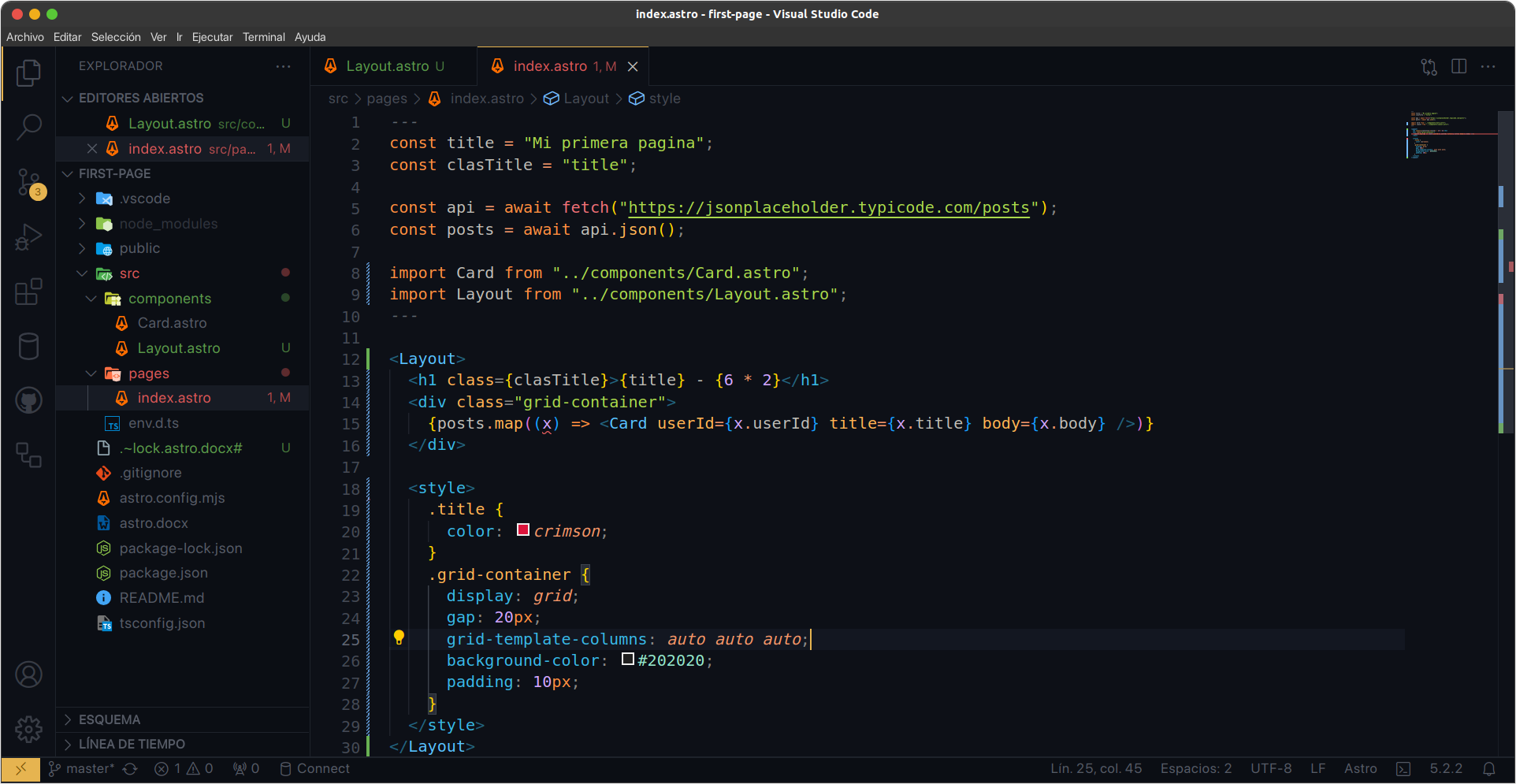
Después de diseñar el componente, podemos invocarlo en una página como se indicó anteriormente, y cargarlo en el "map" con los datos correspondientes provenientes de la API, utilizando los atributos que definimos en el archivo TypeScript del componente.

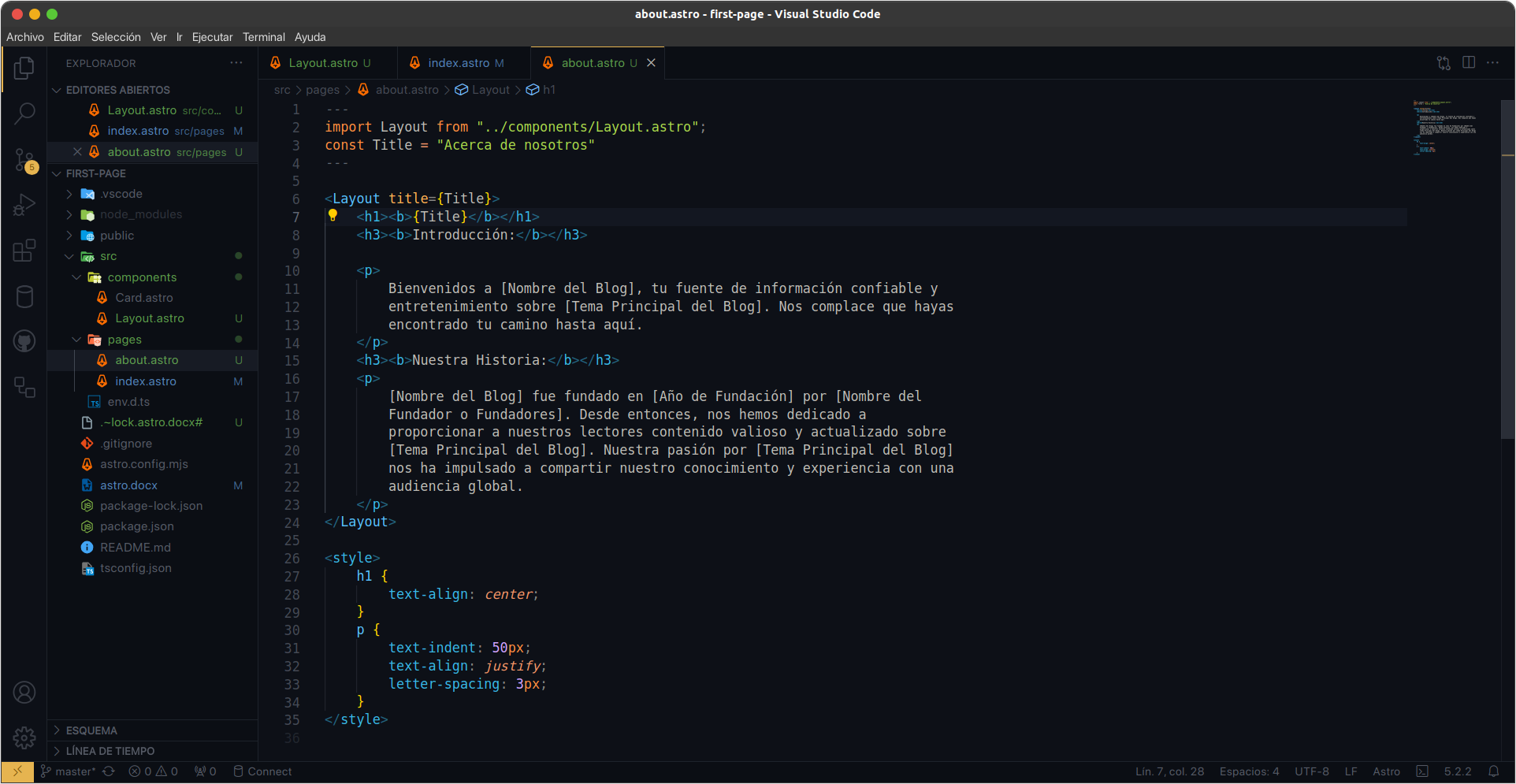
## Layouts

Uno de los desafíos a los que nos enfrentamos al programar en Astro, y específicamente al crear numerosas páginas nuevas, radica en la necesidad de repetir código de forma reiterada. En estos escenarios, Astro emplea una función denominada "layouts" que nos permite utilizar una base de código en la mayoría de las páginas, y que funciona de la siguiente manera.



Creamos en componentes un componente layout y lo dotamos con la base del html ademas de los estilos que podamos repetir.

Además, es posible cargar el diseño (layout) de manera similar a un componente, lo que implica que el contenido dentro de este estará ubicado en la posición definida como "<slot/>". Como es de esperar, también es factible transmitir datos a través de él mediante el uso de propiedades (props).

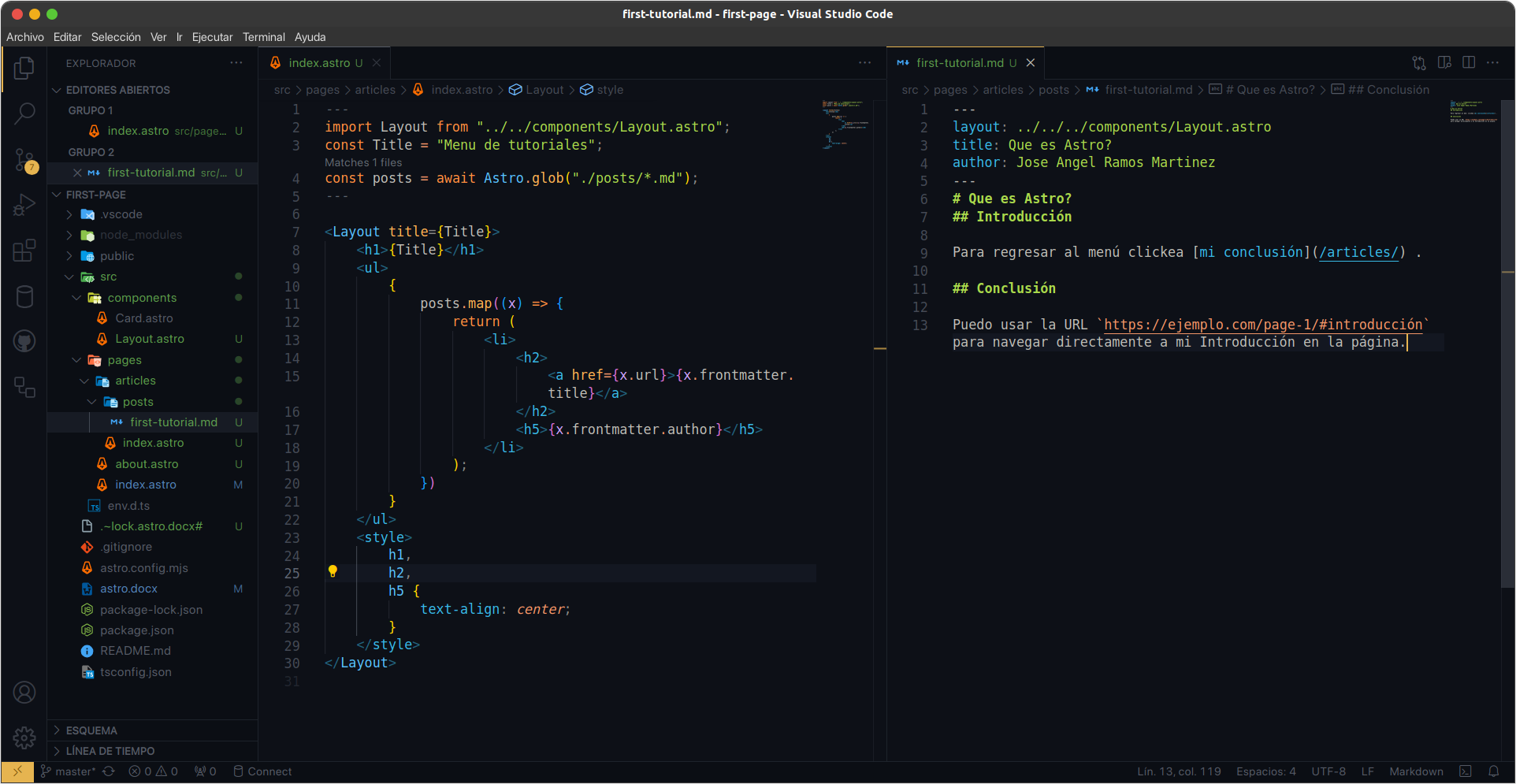
En este caso se pudo cargar el titulo de la pagina en el componente layout, que se puede verificar como aparece en el componente en la siguiente imagen.



## Markdown

Uno de los formatos que resulta de gran utilidad para la creación de notas de texto en la vida de un programador es el formato Markdown (MD). Es altamente apreciado por su simplicidad en la redacción. Es precisamente por esta razón que el framework de Astro aprovecha la oportunidad de incorporar compatibilidad con este formato, permitiendo que forme parte integral de sitios web tipo blog o de artículos que puedan ser redactados.

Para incluirlos en nuestro marco de trabajo, Astro los convierte directamente en la vista como un formato html para simular la accion de una pagina web.

Como podemos observar, para mantener una organización eficiente, crearemos una carpeta que contendrá los artículos, junto con un archivo Astro que funcionará como menú para que el autor pueda acceder a los posts de manera sencilla. En el archivo Markdown (MD), aprovecharemos la sintaxis de Astro que nos permite manipular código de JavaScript, para que en primer lugar, podamos importar el layout que sirve como base para la estructura del archivo principal. Además, tendremos la capacidad de definir otros atributos que podremos llamar desde el archivo Astro utilizado como menú. Esto se logra mediante el uso de "Astro.glob," que realizará un mapeo y guardará en un arreglo todos los archivos especificados, filtrando aquellos con extensión .md, y a través del "frontmatter" podremos acceder a los atributos que hemos definido en el archivo en formato Markdown.

