# Interfaz Grafica

Requerimientos:

* Editor de código (Visual studio code) <https://code.visualstudio.com/>

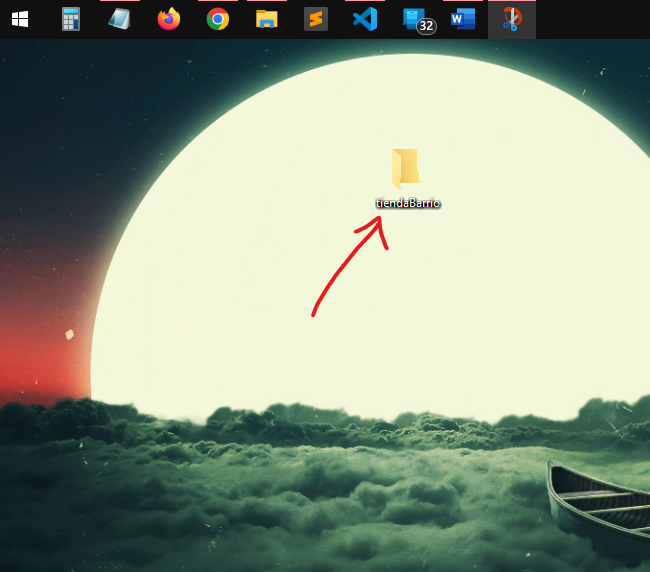
Tecnologías:

* JavaScript
* CSS
* HTML5
* Bootstrap 5
* JQuery 3
* DataTable

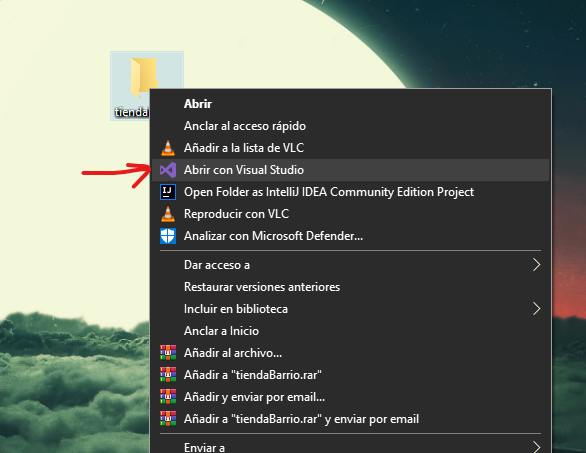
## Estructura de carpetas

Primero que todo necesitamos un patrón de diseño el cual nos sirve para estructurar nuestro código, utilizaremos MVC (Modelo Vista Controlador).

Crearemos nuestras carpetas y archivos, ubicamos la raíz de nuestro proyecto donde vamos almacenar el código. yo voy a realizar en el escritorio tu lo puedes guardar donde quieras ej:

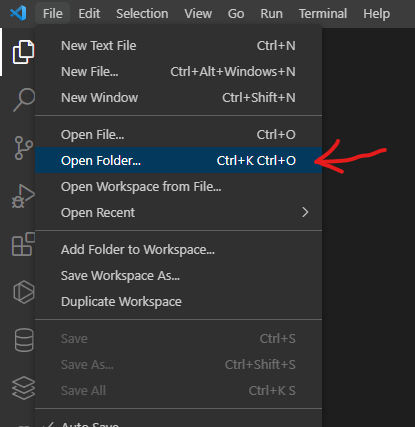


Listo ahora abrimos, le damos clic derecho y la abrimos con visual studio code ej:

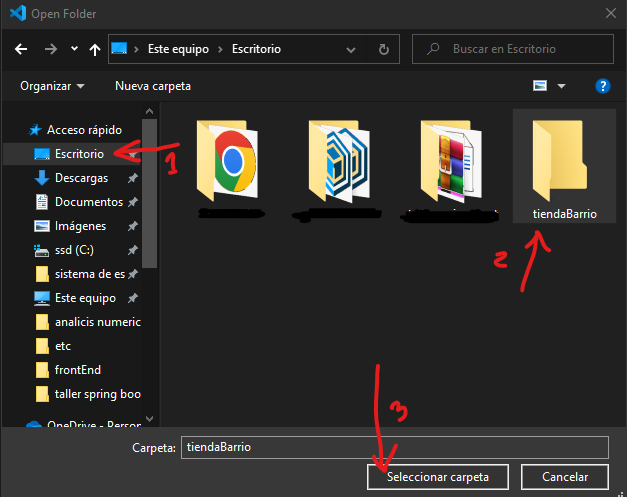


En caso tal no tengas esta opción, solo abres visual studio code normal, y le das en

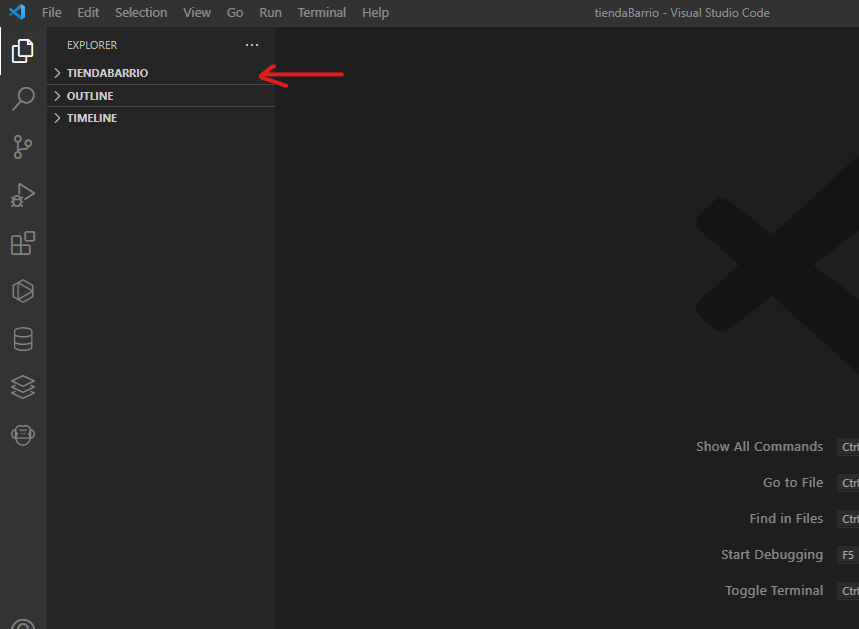
File🡪 **abrir carpeta u Open folder** e ingresas a tu carpeta ej:



Luego seleccionas la ruta de tu proyecto, yo lo tengo en escritorio ej:



listo ahora solo queda crear nuestra estructura:

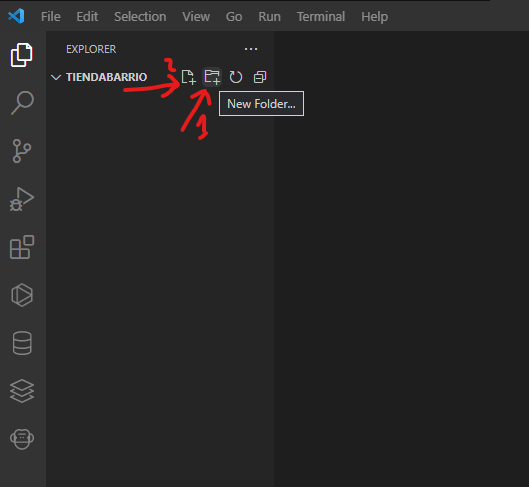


Para crear carpetas es super sencillo, le damos clic en el nombre del proyecto,

Y nos aparece las opciones de crear carpeta y archivos así:

1 crear carpetas

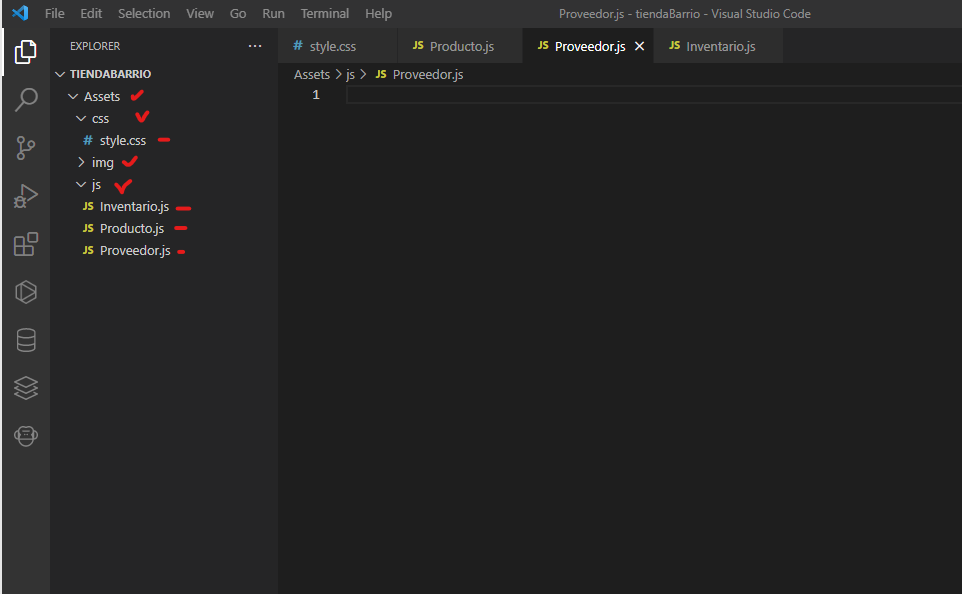
2 crear archivos



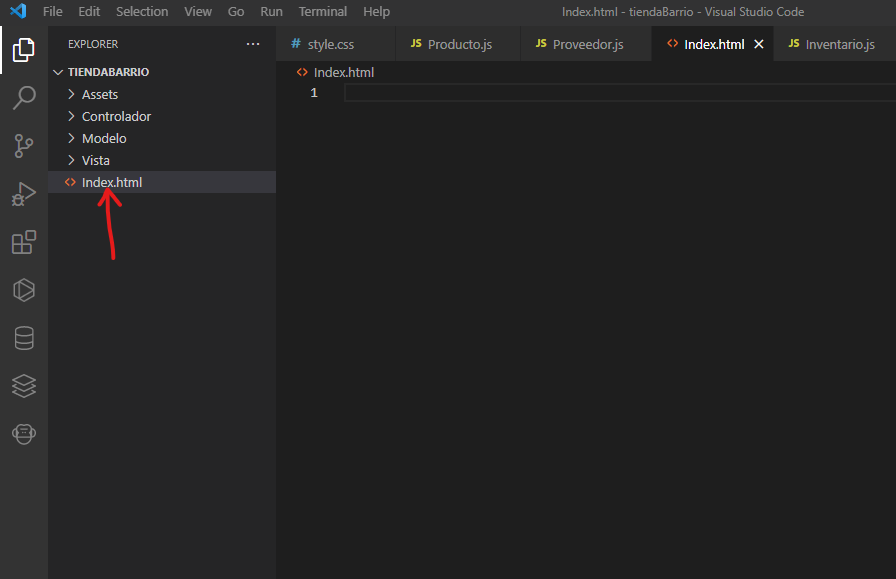
Ahora si armemos nuestro esquema, **primero necesitamos una carpeta de utilidades que nos sirva para guardar los archivos activos css, js, imágenes librería entre otros.**

Nos quedaría así la primera carpeta:

1. Assets/
   * 1. Css/
        + style.css
     2. Js/
        + Productos.js
        + Proveedor.js
        + Inventario.js
     3. Img/



Listo, ahora seguimos con la estructura de modelo vista controlador, y de igual forma necesitas un archivo HTML que nos muestre el inicio de nuestra aplicación. Que este ubicado en la raíz del proyecto, necesaria e importante que se llame index.html para que se reconozca como nuestro archivo principal al iniciar la aplicación así:



Listo ya tenemos nuestro esquema completo ahora sí, iniciemos con el diseño.

Primero iniciemos con nuestro archivo principal index.hmtl.

Utilizaremos la sintaxis de html5 que la podemos identificar al ver la antinomia del proyecto web ej:

<!DOCTYPE html> :

La etiqueta doctype declara el tipo de documento, por lo que esta está señalando que este es un documento HTML.

Etiquetas principales de html5

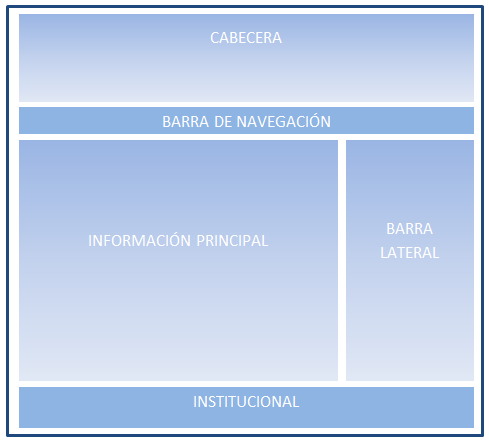
<header> ... </header>

<section> ... </section>

<nav> ... </nav>

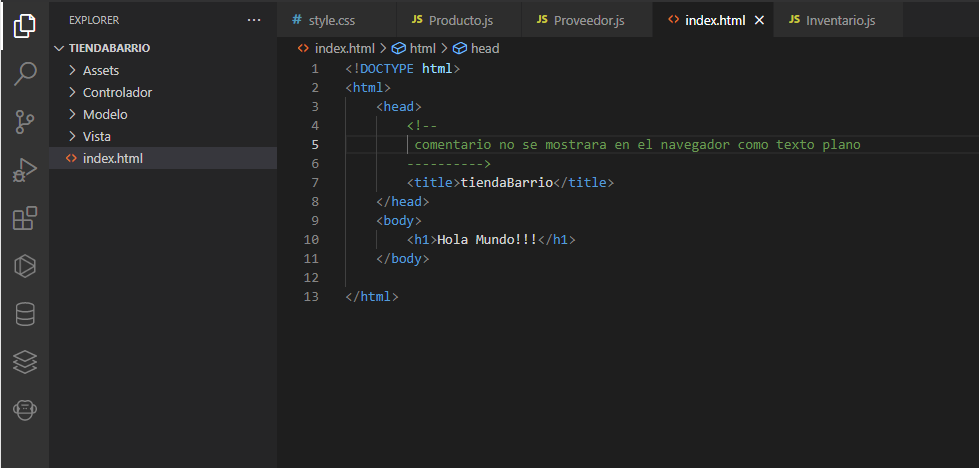
<footer> ... </footer>

Para conocer mas sobre las etiquetas y sus funciones aquí <https://lenguajehtml.com/html/>



Sabiendo esto podemos iniciar:

Mostremos nuestro **hola mundo** en el navegador, para probar sus funciones



Ahora podemos integrar un plugin que nos ayuda a visualizar en tiempo real lo que vamos programando en HTML, nos vamos a extensiones, en algunos casos puede estar del otro lado del editor.

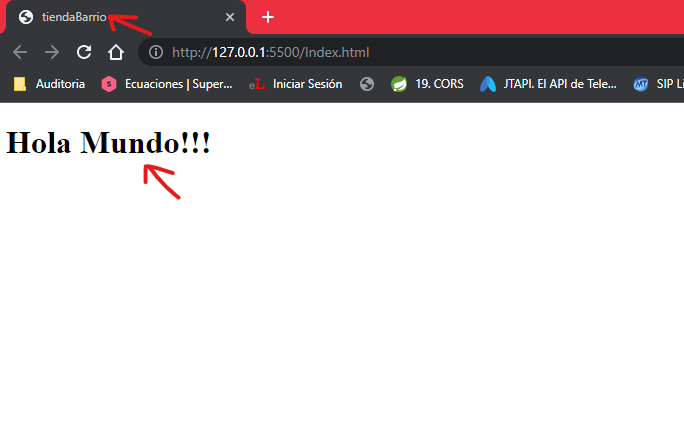


Buscamos el plugin **Live Server** y le damos instalar, esperamos que se instale y cerramos esa pestaña, Vamos a nuestro archivo HTML.



En la parte de abajo en la línea azul veremos un ítem que dice **Go Live** le damos clic y esperamos que nos muestre el navegador.

Así se vería en el navegador:



Nos muestra el titulo la pagina y el h1 que le pasamos con el texto, muy bien ahora si integremos todas las funciones. Css, js a ver cómo funcionan.

A este elemento le llamamos metadatos el cual nos sirve para hacer enlaces, funciones específicas que debería aceptar nuestro sistema, los principales.

* <head> </head>, que indican los metadatos respecto a un documento como los enlaces. Esta clase de información brindada sólo es visible para los navegadores.
* <title> </title>, la etiqueta que define el título de una web.
* <link>, utilizada para afiliar recursos que son externos a dicho documento.
* <meta>, utilizada para poder colocar metadatos como el autor de la web o hasta su descripción. Siempre es usada para datos que no poseen una etiqueta específica.
* <style></style>, funciona como medio para ingresar código CSS en el documento.

**Configuraciones de la pagina <head>**

* **Especifica la codificación de caracteres para el documento HTML**

<meta charset="UTF-8">

* **Declarar el idioma de la página web**

<html lang="en">

* **Definir palabras clave para motores de búsqueda:**

<meta name="keywords" content="tienda virtual,tiendabarrio,ventaproductos">

* **Define una descripción de tu página web:**

<meta name="description" content="tienda virtual de barrio">

* **Definir el autor de una página:**

<meta name="author" content="John Doe">

* **Actualizar documento cada 30 segundos:**

<meta http-equiv="refresh" content="30">

* **Configuración de la ventana gráfica para que su sitio web se vea bien en todos los dispositivos:**

<meta name="viewport" content="width=device-width, initial-scale=1.0">

La <link> etiqueta define la relación entre el documento actual y un recurso externo.

La <script> etiqueta se utiliza para incrustar un script del lado del cliente (JavaScript).

Elementos de secciones

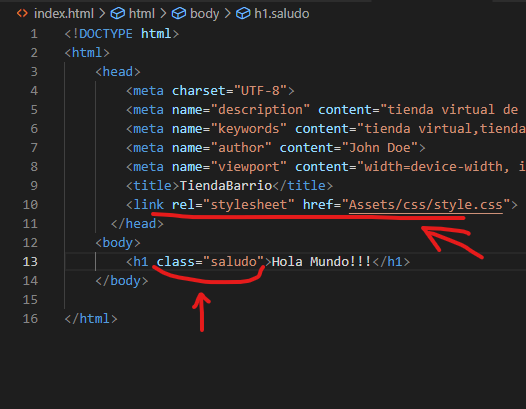
Dividir una página por secciones es más que necesario para que sean visualmente atractivas y posean una estructura que los navegadores comprendan. Es por ello que, mencionamos las versiones más comunes de etiquetas para poder crear secciones dentro de una web:

* <body> </body>. Incluye todo el contenido que se desee mostrar dentro de la página web.
* <nav> </nav>. Define el contenido que estará ubicado dentro de la sección de navegación de la página.
* <main> </main>. Permite elegir el contenido principal que posee un documento, siendo único.
* <h1>, <h2>, <h3>, <h4>, <h5>, <h6>. Permiten crear una estructura para el contenido de una página web. Son esenciales para cumplir con la correcta organización de dicho contenido.
* <section> </section>. Indica la sección de un documento HTML.
* <article> </article>. Incluye el contenido de una página que es independiente.
* <aside> </aside>. Todo el contenido extra de una página web se incluye en estas etiquetas, pues posee poca relevancia, pero coincide con la web.
* <footer> </footer>. Incluye el contenido que irá en el pie de página.
* <header> </header>. Define la parte superior o cabecera de una web. En ella mayormente se encuentra el logotipo y nombre de la web, así como el menú.

Bueno, después de repasar todo para entender un poco mejor lo que vamos a realizar

**Le vamos a dar estilo a nuestro titulo hola mundo**

Lo primero es importar o conectar nuestro archivo externo style.css y después asignar una clase a nuestro <h1 class=” saludo”> quedaría así.

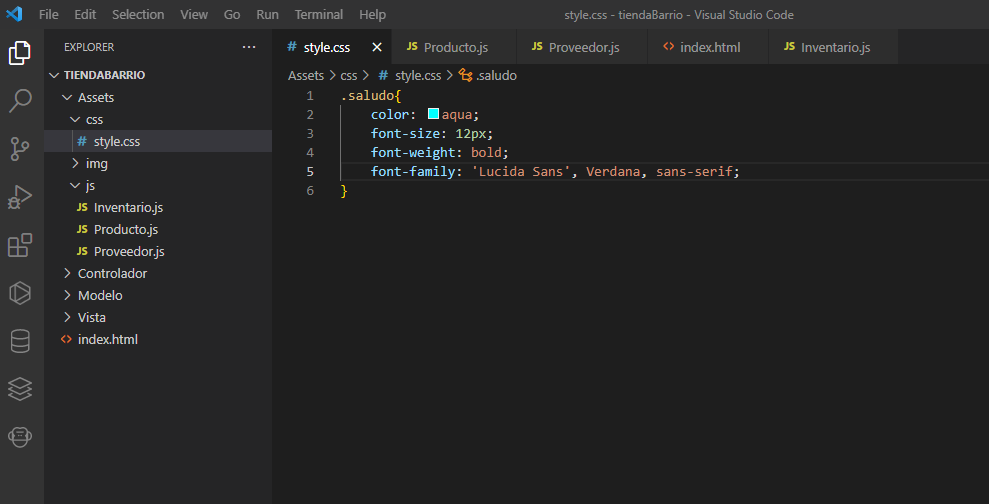


Luego abrimos nuestro archivo css para dar el estilo que necesitamos.

Aquí en este link encontraras todas las sintaxis de html y css que te ayudaran mucho:

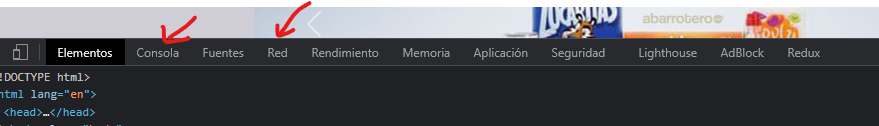
<https://www.w3schools.com/css/default.asp>

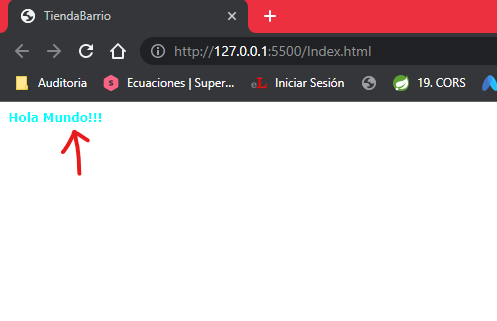
así quedara nuestro CSS con, los estilos asignados que son **color del texto, tamaño, negrita y tipo de letra.**



Ahora guardemos y miremos nuestro navegador a ver si se aplicaron los cambios.

De Igual forma nos podemos apoyar de la consola del navegador nos muestra error y las peticiones que realizaremos mas adelante.





Muy bien, ya hicimos nuestro primer enlace css y sabemos como funciona ahora para evitarnos esto **utilizaremos un framework css** que nos facilitara el uso de etiquetas y diseños mas complejos y para una mejor experiencia de usuario.

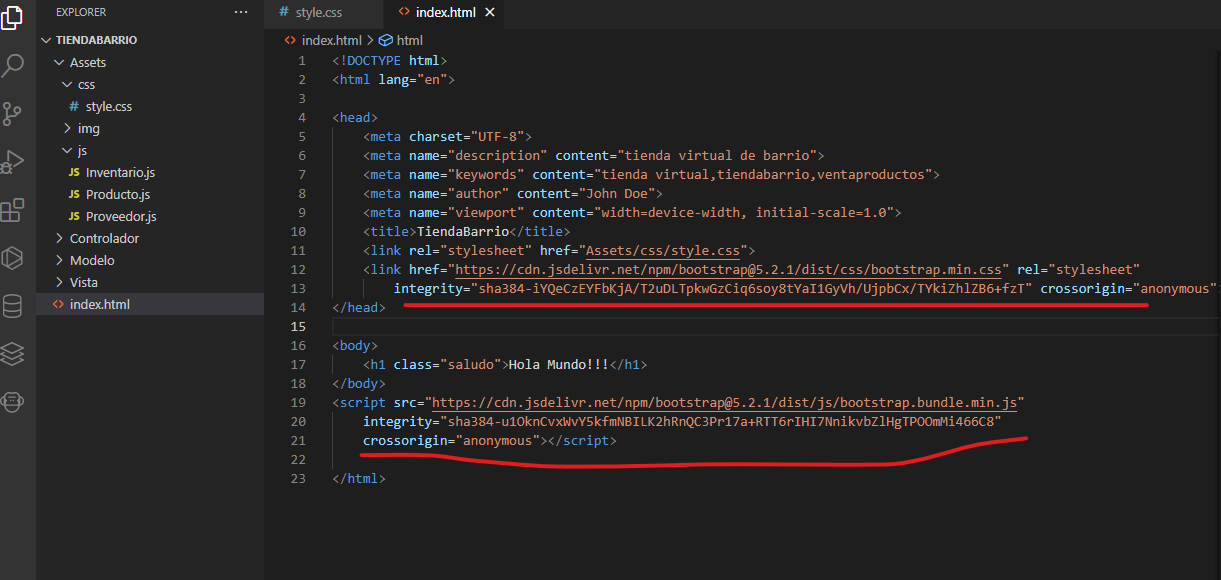
Primero es integrar sus cdn o Link de css , vamos a su pagina oficial.

<https://getbootstrap.com/docs/5.2/getting-started/introduction/>

aquí encontraremos ejemplo muy prácticos de la integración del framework.



Y así se vería en nuestra aplicación:

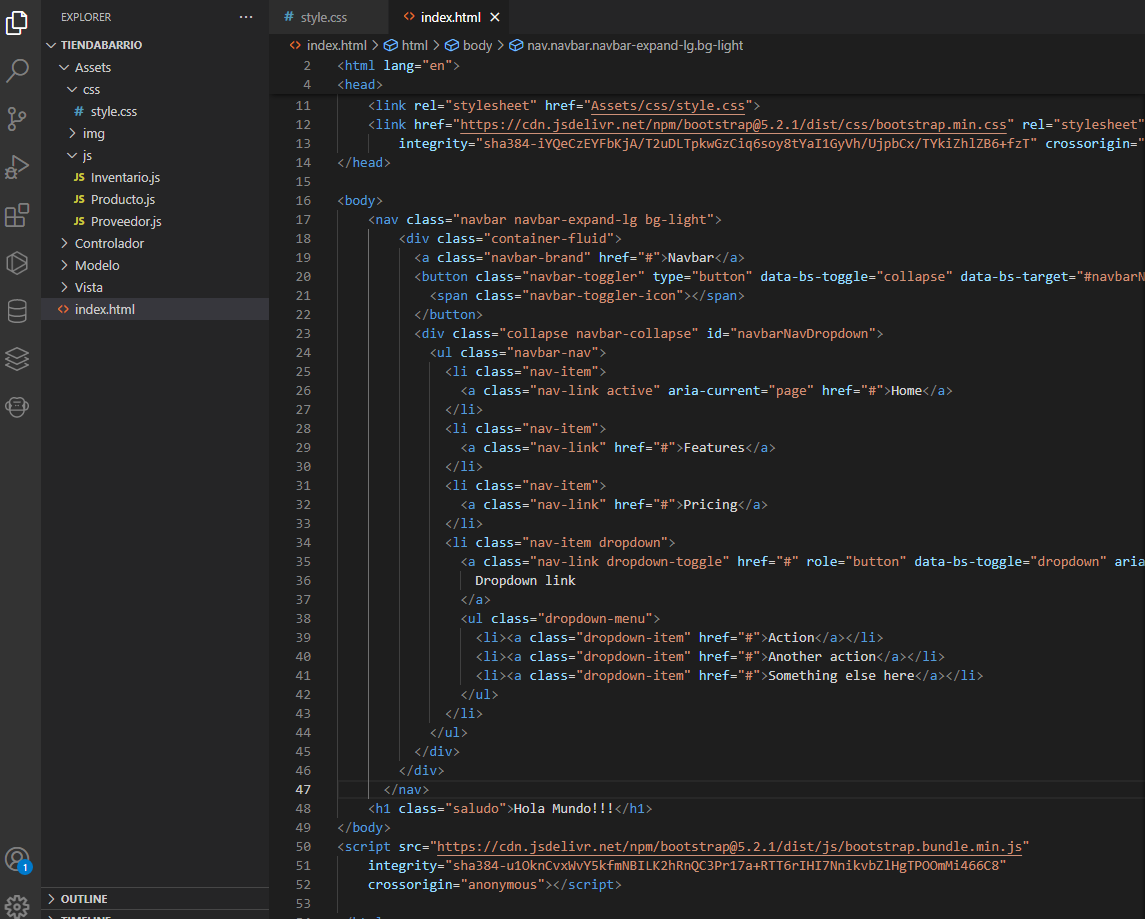


Listo probemos si es real o no. Volvamos a su documentación y busquemos Nav, para que nos muestras las barras de navegación disponibles, con estilos y todo.

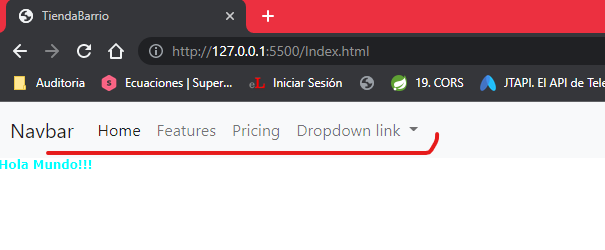


Elijamos esta, copiemos el código debajo y pequemos en nuestro código.

Quedaría de esta forma.



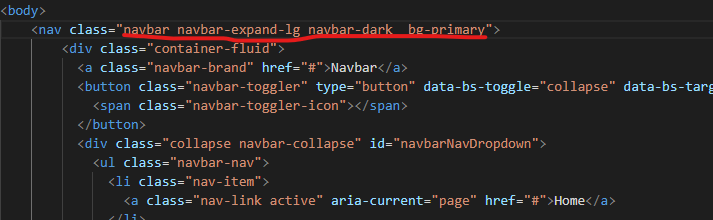
Guardamos y vamos a nuestro navegador a ver como quedo.



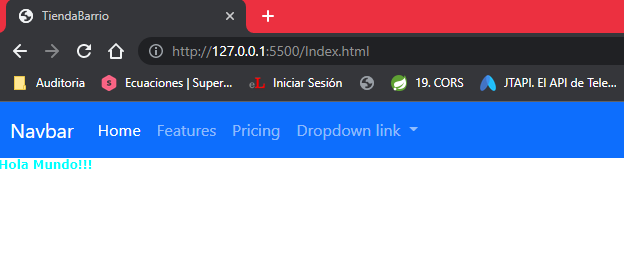
Perfecto, vamos bien ahora cambiemos ese color de navegador vamos de nuevo a Bootstrap y busquemos Color **schemes o esquema de colores.** Copiemos el estilo que tiene la clase de color azul o el negro si prefieres. Solo copiemos nav-dar bg-prymary es la clase del color y el texto.

****

Asi se veria en nuestro codigo



Vamos al navegador y verificamos. Que todo este bien.

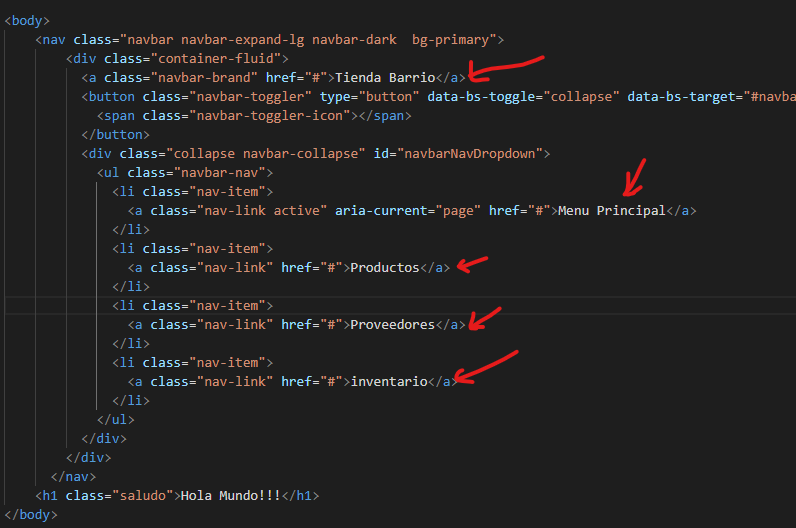


Listo todo está bien, personalicemos los botones, cambiemos los nombres y eliminemos lo que no necesitamos.

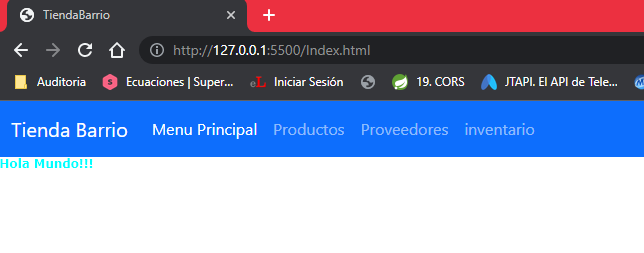
Necesitamos el siguiente botones.

**Menú principal, producto, proveedor, inventario**

Así quedaría en nuestro código:

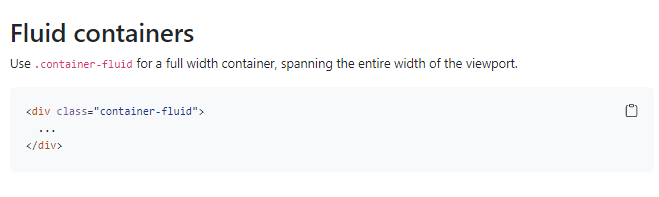


Así queda en el navegador:

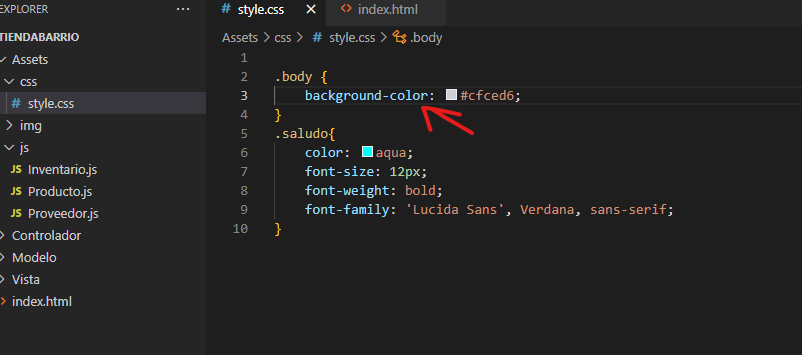


Vamos bien, pongamos unas imágenes de nuestros productos en el menú principal para que sea vea mejor. Vamos a Bootstrap y busquemos, **container-fluid** para crear un contenedor central en la página.

**Este será el contenedor de todo el proyecto lo copiamos y agregamos todo el código html dentro.**

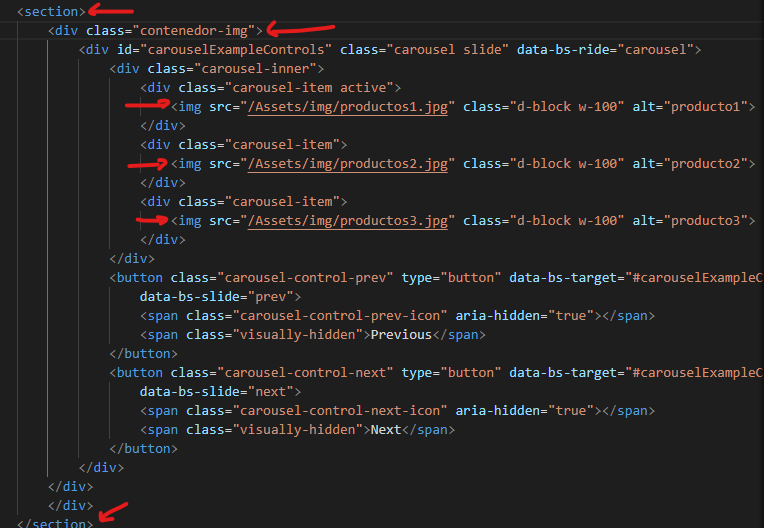
****

Luego agregamos una clase al body de nombre body para dar color al fondo desde nuestro css.



Podemos eliminar el css .saludo

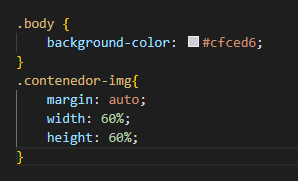
Ahora creamos la etiqueta <header> y dentro metemos todo el <Nav> asi:



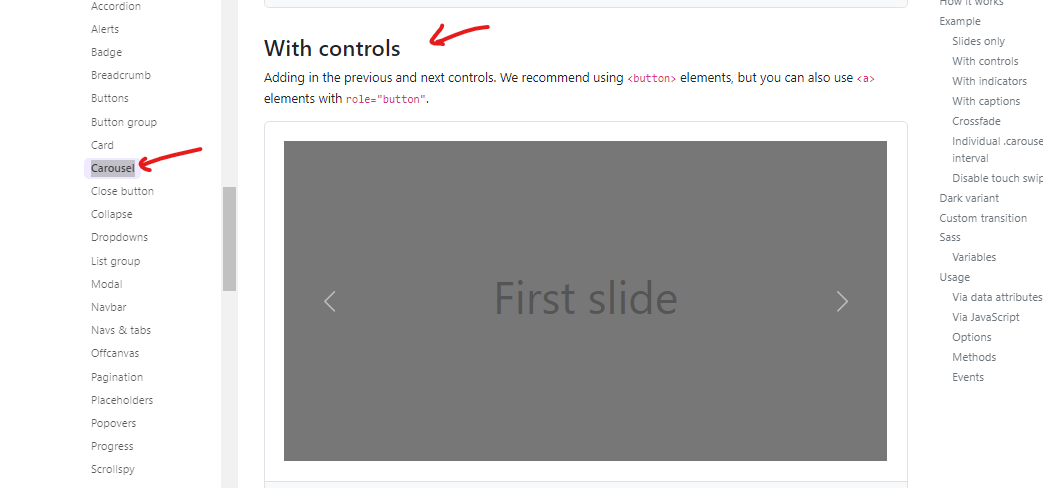
Luego, creamos una etiqueta de <section> y un <div> con la **clase contenedor-img** y metemos todo el código **Carousel. En Bootstrap buscamos Carousel y Copiemos su código y pegamos en nuestro proyecto.** Cuando ya tengamos el código necesitaremos tres imágenes para la muestra

Puedes poner las que quieras, pero esta vez solo usaremos tres.

**En el archivo css , ponemos las clase . contenedor-img**

****

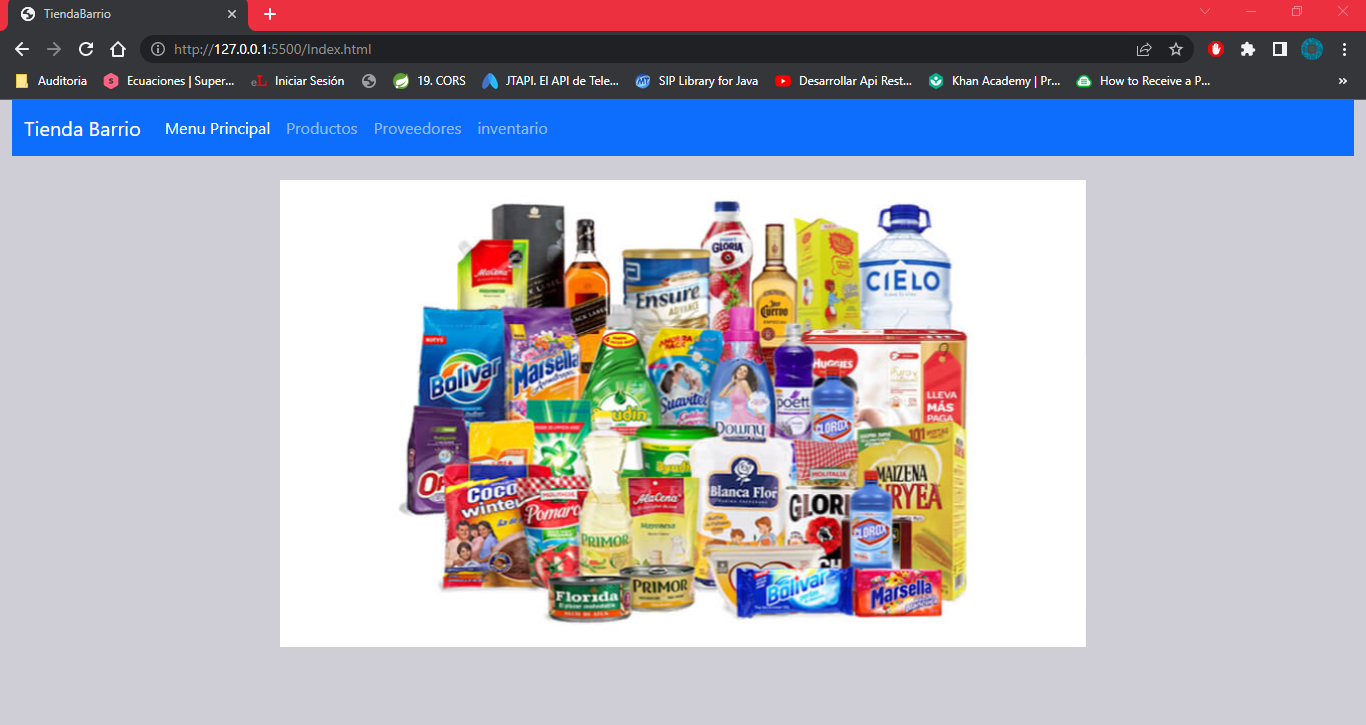
**Luego pasamos a nuestro índex y ponemos el código de Bootstrap.**



Asi quería el código, podemos eliminar el h1 que creamos al inicio en index.html:

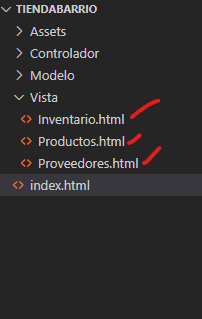


Ahora si solo vamos a nuestro navegador y vería así:

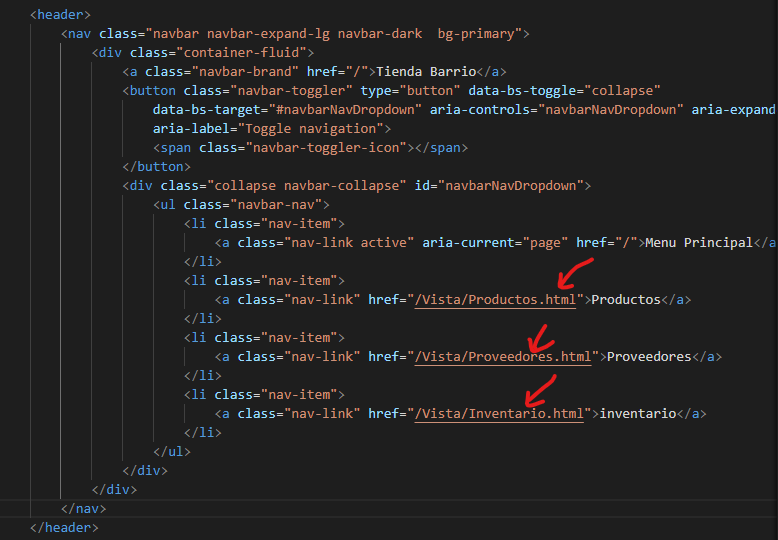


Listo tenemos nuestra vista ahora crearemos las redirecciones para cada módulo,

Vamos a la carpeta vistas y crearemos las vistas para cada módulo así:



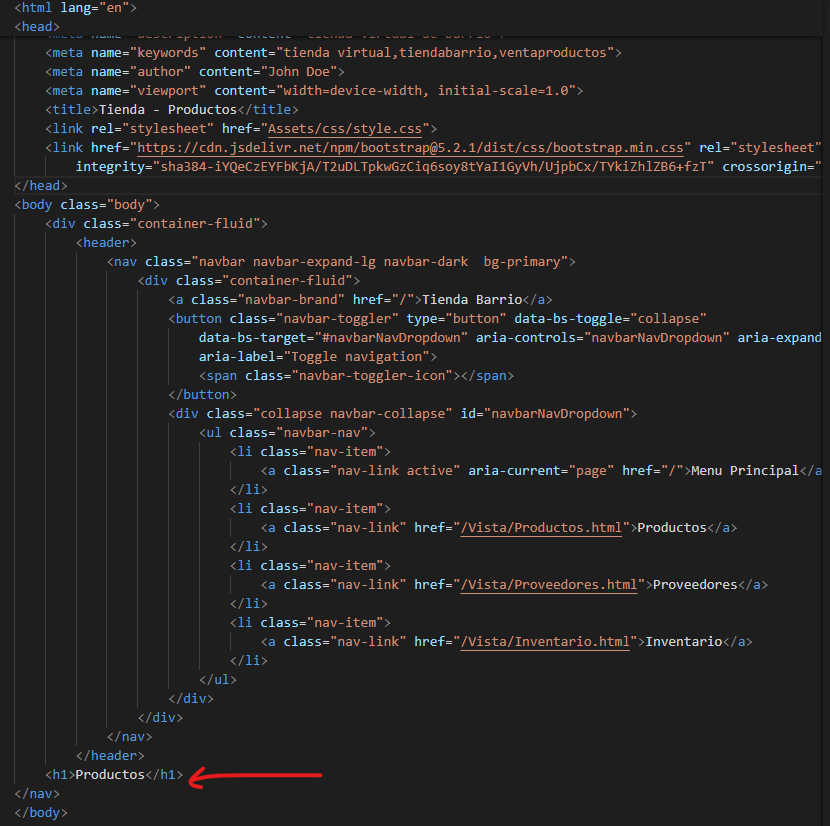
Ahora miramos la redirección en nuestro código:



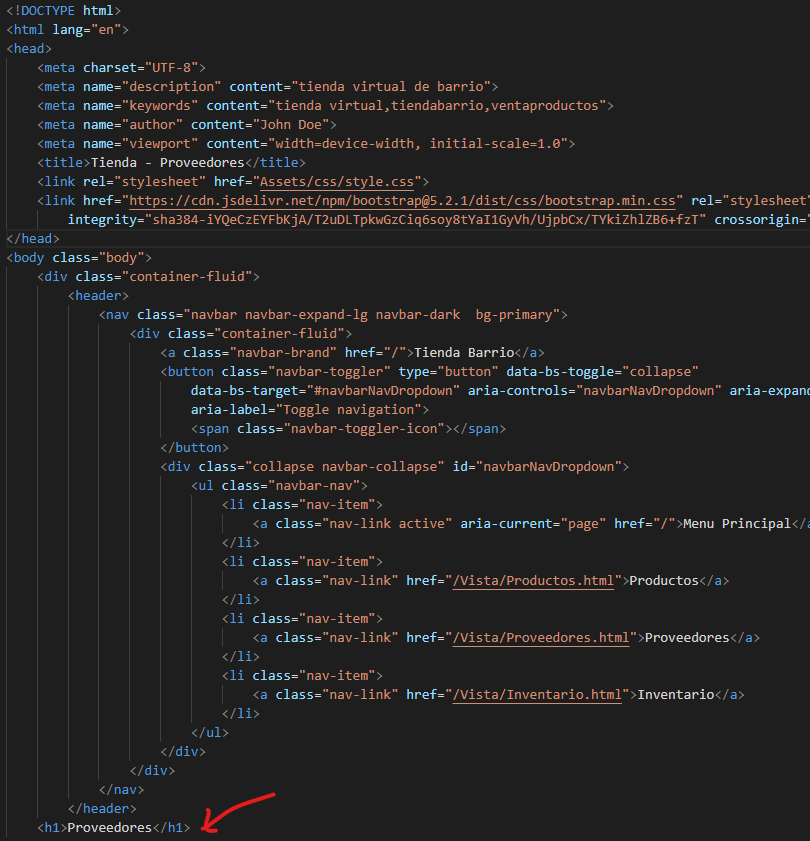
En cada uno de los modulo ponemos código HTML con sus respetivos cuerpo ejemplo:

Pasamos la etiqueta <header> a cada uno de los módulos.y cambiamos sus title y cuerpo.

Vista - Producto



Vista - Proveedores

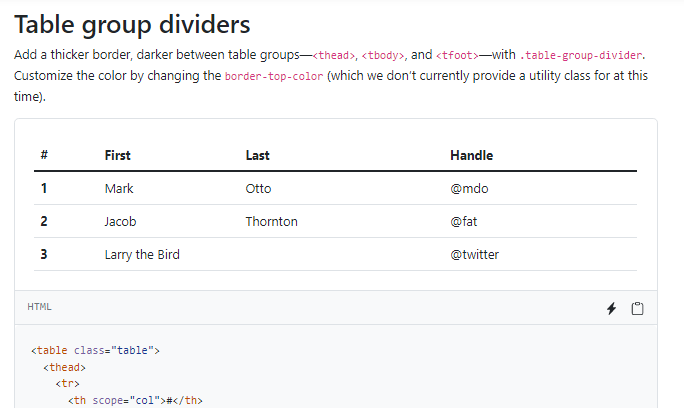


Ahora podemos ir al navegador y presionar cualquier botón y nos llevara a cada módulo.

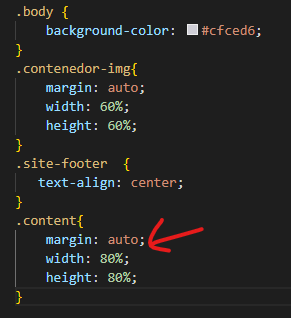
Ahora bien, podemos iniciar con el módulo productos el cual recibe una lista de productos, donde podamos crear, eliminar y actualizar un producto.

Lo principal es ir a Bootstrap y buscar **Table** la tabla que va mostrar los productos y encima un botón para ingresar los productos.

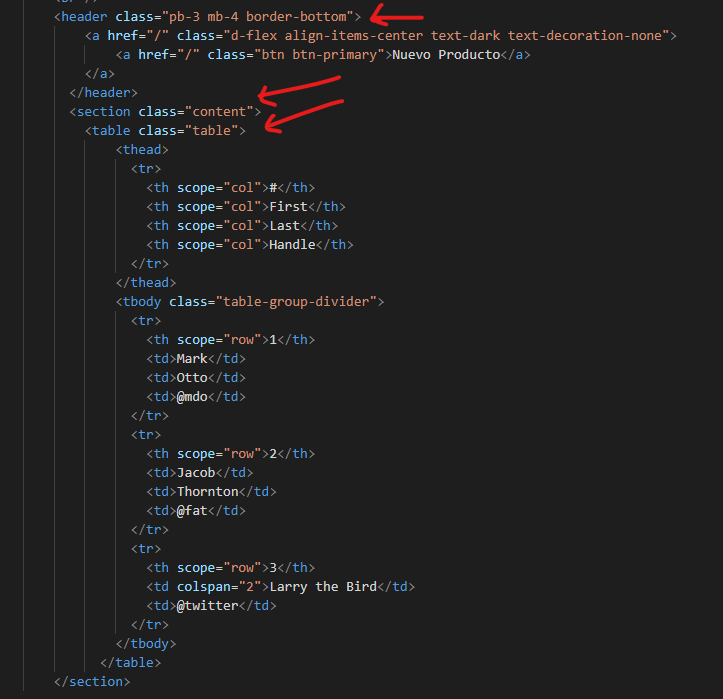
Esta seria nuestra tabla, copiamos el código lo ponemos de las etiquetas <sección>



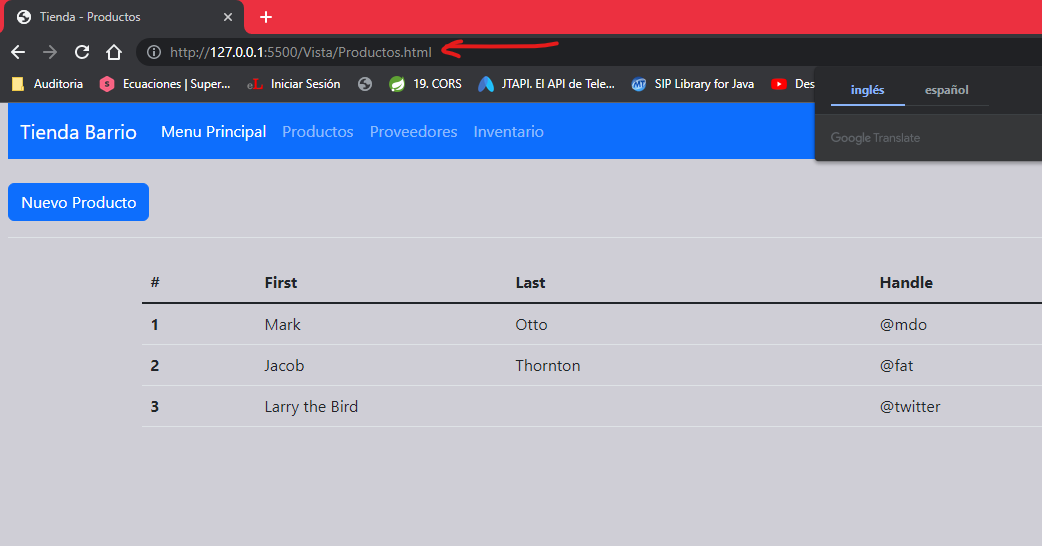
En la etiqueta <section> agregamos una clase **content, para dar posición a nuestra tabla, luego vamos al archivo css y ponemos lo siguiente, en caso no actualice verificar las rutas o enlaces css.**

****

Así sería nuestro código de la tabla en productos:



En el navegador:



Entonces tenemos listos nuestros módulos, ahora sigue traer los datos de nuestro api y pintarlos en nuestra tabla productos.

Para esto utilizaremos **javascript** y dos librerías llamadas **DataTable y JQuery**

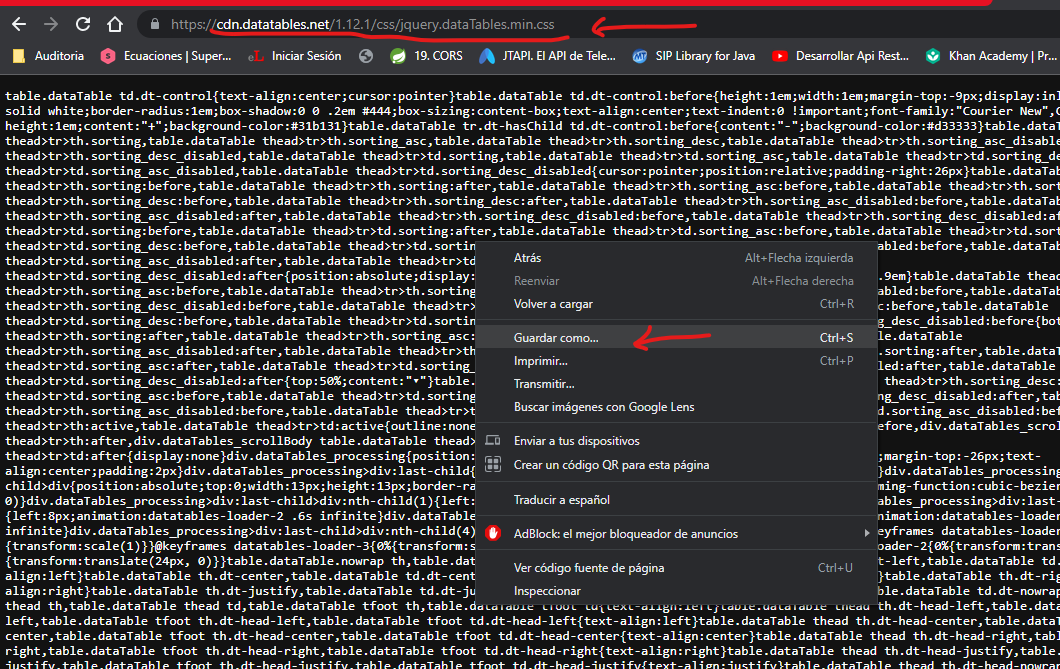
**Aquí te dejo la documentación si quieres conocer más sobre esta librería.**

[**https://datatables.net/**](https://datatables.net/)

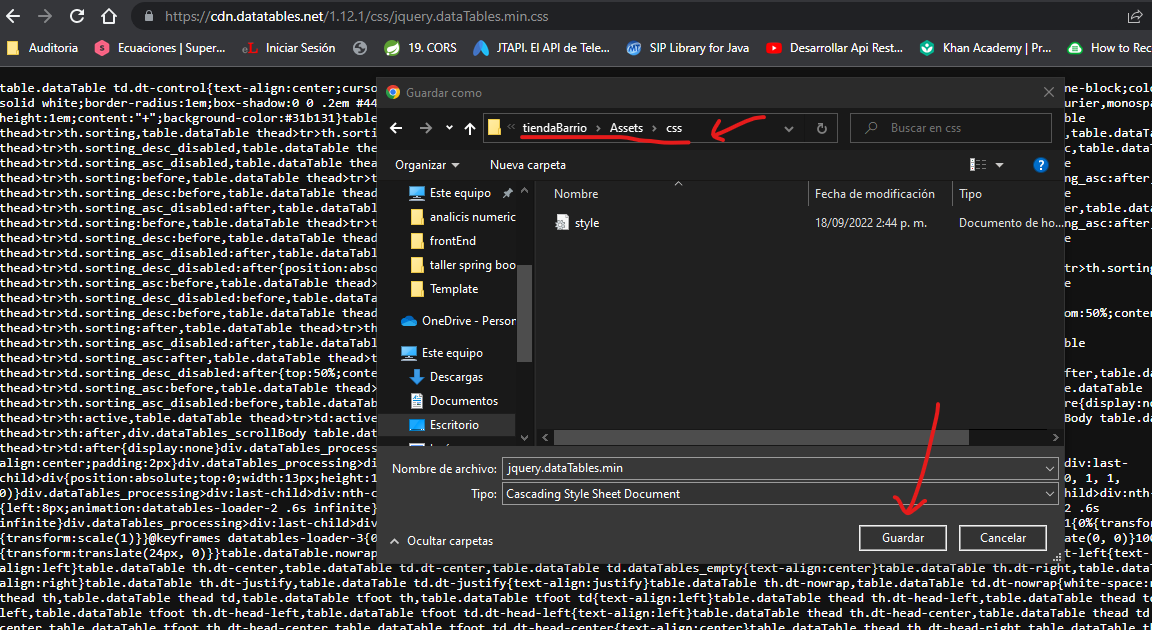
Lo primero es importar las librerías a nuestro proyecto lo primero es ir a su pagina oficial y ubicar sus librerías. Seguimos los pasos primero guardamos el **css** luego el **js** listo podemos usar sus funciones.



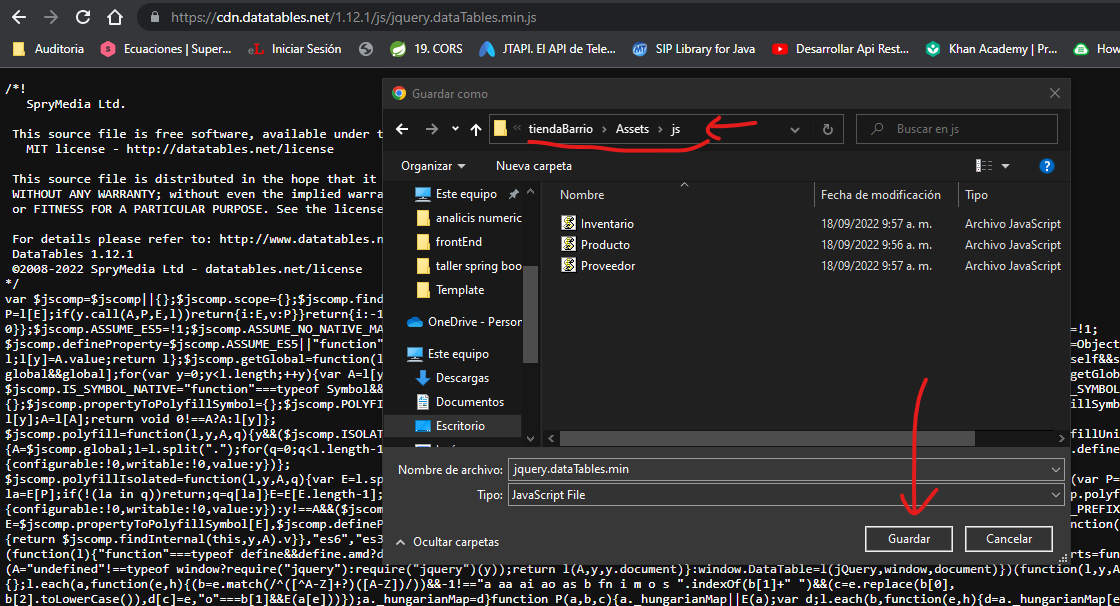
Como lo hacemos copia el link que esta en el css y lo pegamos en el navegador nos aparece la librería le damos guardar como y lo ubicamos en nuestro proyecto de esta forma:



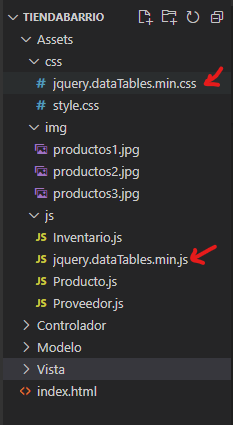
Lo guardamos en nuestro proyecto en la carpeta css, así:



Listo hacemos el mismo proceso para el siguiente archivo js



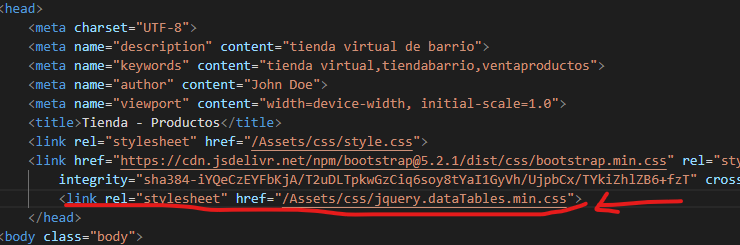
Verifiquemos nuestras carpetas de archivos, tendríamos algo parecido.



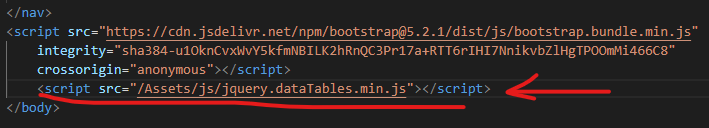
Bien ahora ya podemos trabajar con **dataTable**

Importemos su librería en las tres vistas donde usaremos las tablas, **producto.html, proveedor.html, inventario.html** así.

Datatable.css

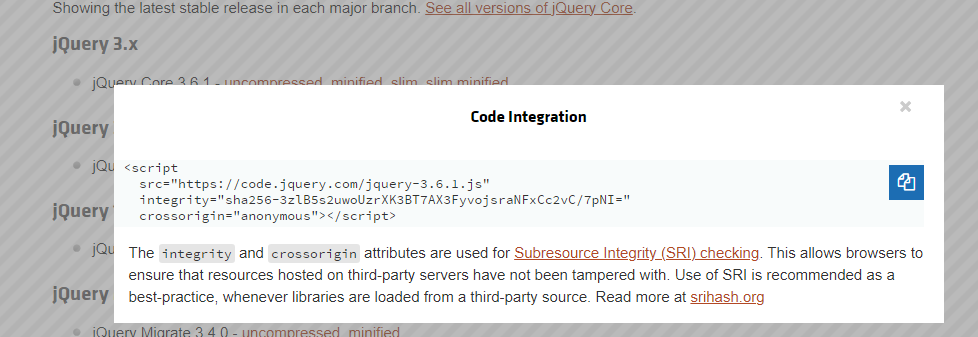


Datatable.js se pone en la parte de abajo del archivo

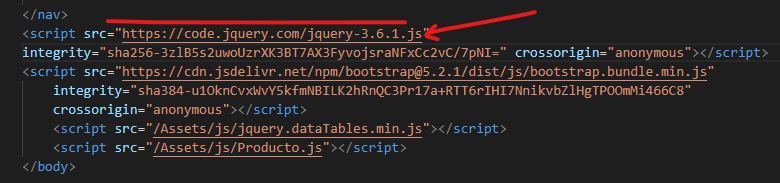


Listo solo falta importar la librearía de JQuery la podemos encontrar en

<https://releases.jquery.com/> le das clic en **– uncompressed** y copias el cdn

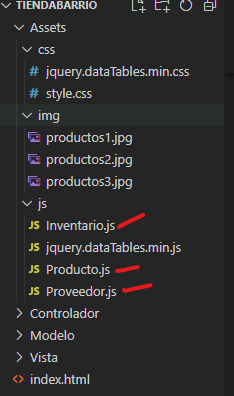


así se vería nuestro código

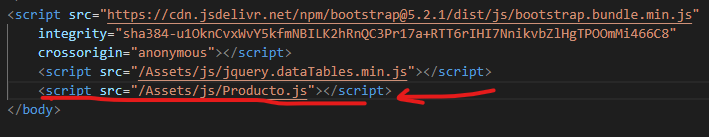


Tenemos listo, probemos que esté funcionando correctamente,

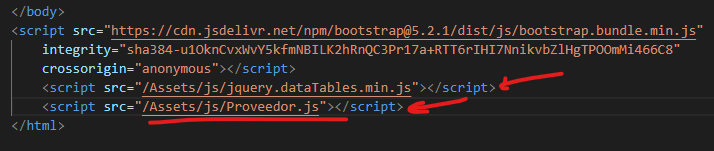
Enlacemos el archivo js que creamos para producto proveedor, inventario. **cada un en su archivo html.** Iniciemos con producto



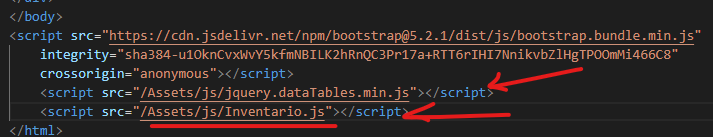
Asi quedaría la integración de nuestro archivo js a producto.html



Lo mismo para proveedor e inventario

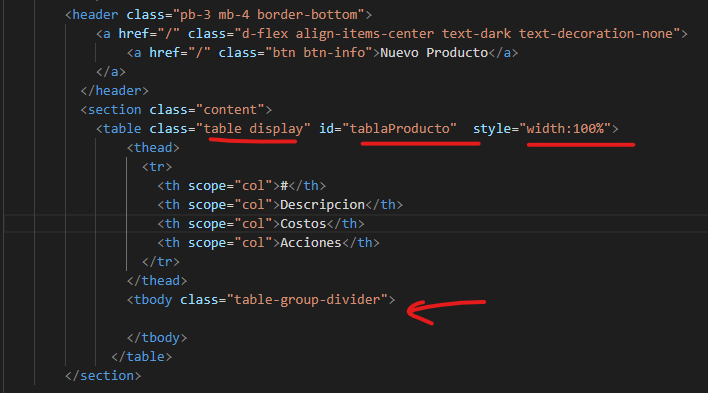


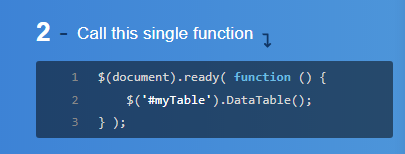
Inventario



Uva vez tengamos ya enlazado todas librería y nuestros archivos podemos continuar probando el código.

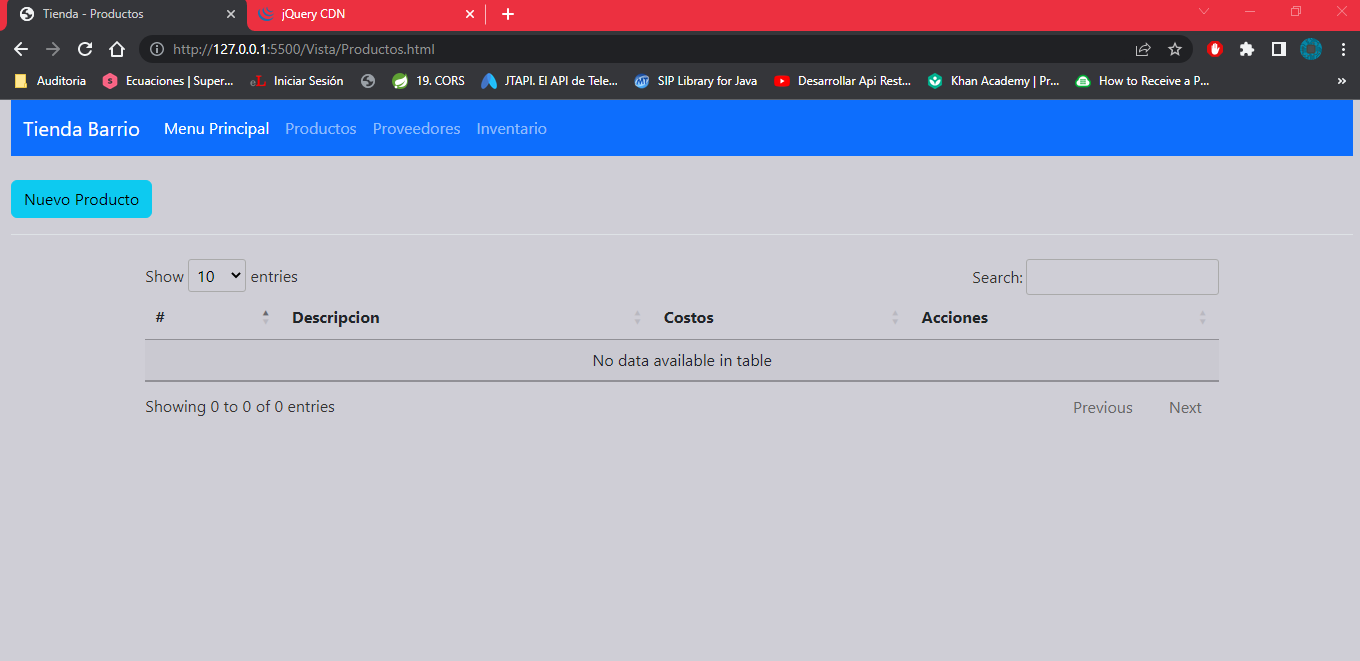
Ahora en nuestro archivo **producto.html** le asignamos un estilo y nombre de clase y un id a nuestra tabla, de igual forma limpiamos nuestra tabla quedaría así:



Abrimos nuestro archivo **producto.js**  y ponemos el siguiente código. (**JavaScript**)

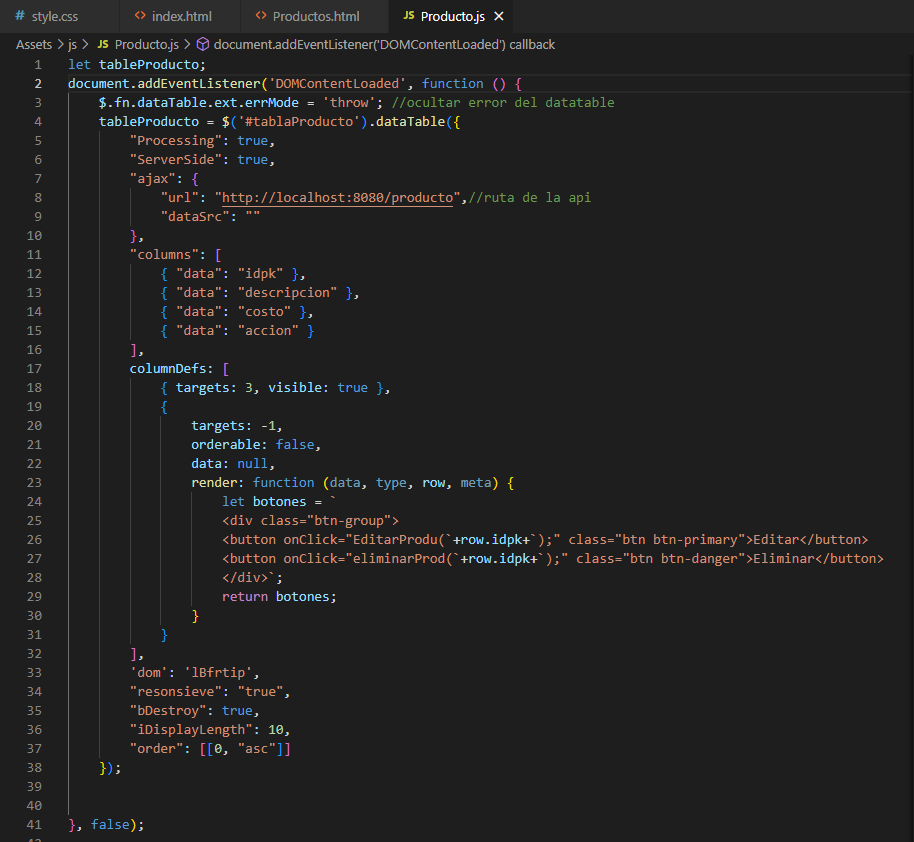
Donde dice **#mytable** ponemos **#tablaProducto** listo probemos a ver qué pasa.

Así se vería nuestra página con los plugin cargados.



En nuestro archivo penemos el siguiente código.

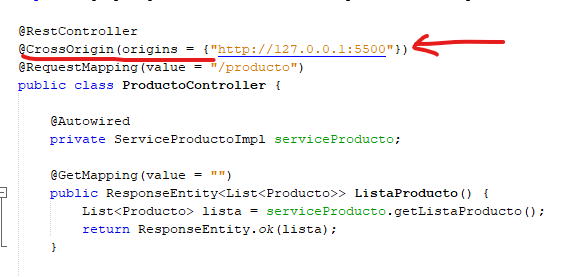
Agregamos un evento en el documento que cuando inicie el archivo JS llame al **datatable** y consuma nuestro api. Mas abajo se crean las dos funciones de editar y eliminar



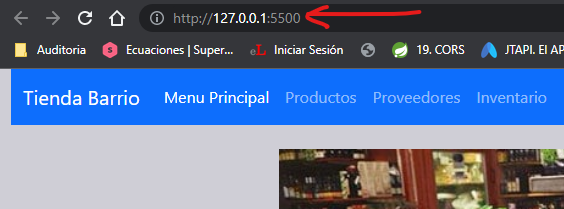
Si por casualidad tienes un error de CORS como este, no te preocupes es fácil de solucionar, ve a tu api en java en clase controller pones el siguiente código así:

**@CrossOrigin(origins = {"aquí va la ruta desde donde se ejecuta de aplicación FrontEnd"})**

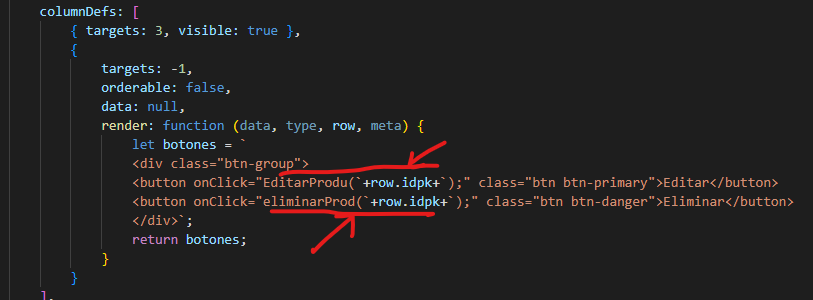
**Aquí te dejo mas información sobre los CORS.** [**https://docs.spring.io/spring-security/site/docs/4.2.x/reference/html/cors.html**](https://docs.spring.io/spring-security/site/docs/4.2.x/reference/html/cors.html)



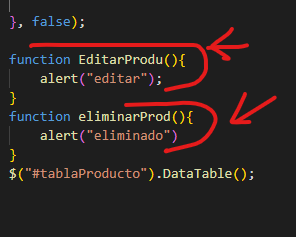
Aquí puedes ver la ruta de tu aplicación:



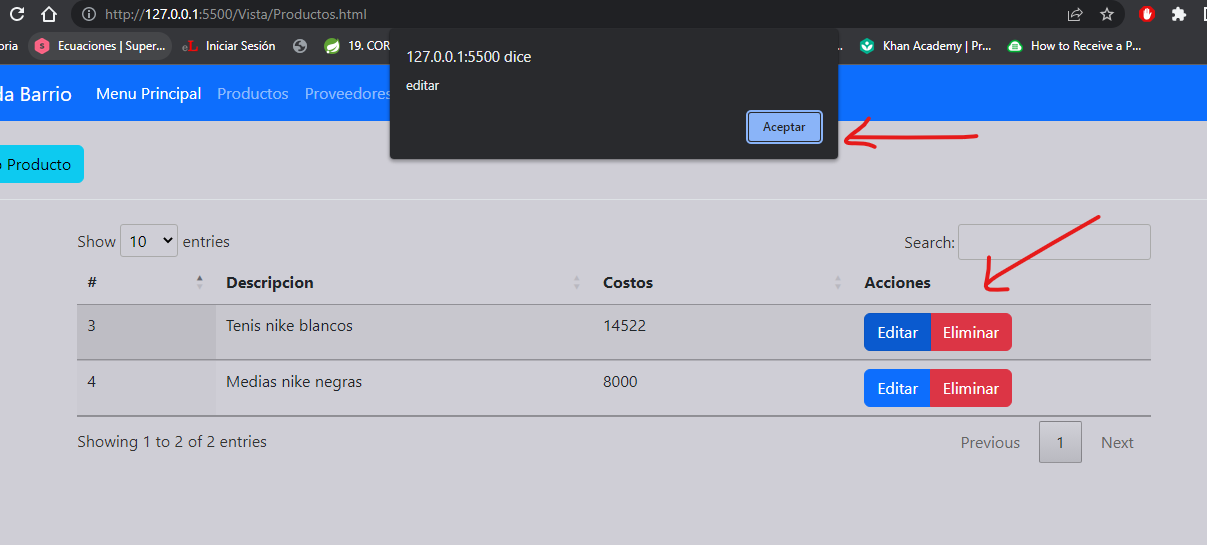
Listo sigamos con el código, ahora en el código aparecen dos funciones muy importantes que son el botón de eliminar y el de editar, la cual tenemos que definir abajo estas son las funciones.



**Así quedarían definidas las funciones con una alerta de prueba**

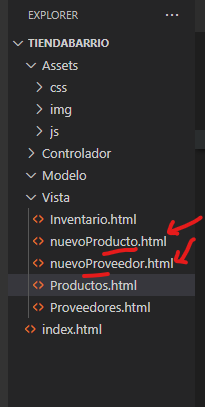


Listo así se vería nuestra pagina con los botones y la alerta.

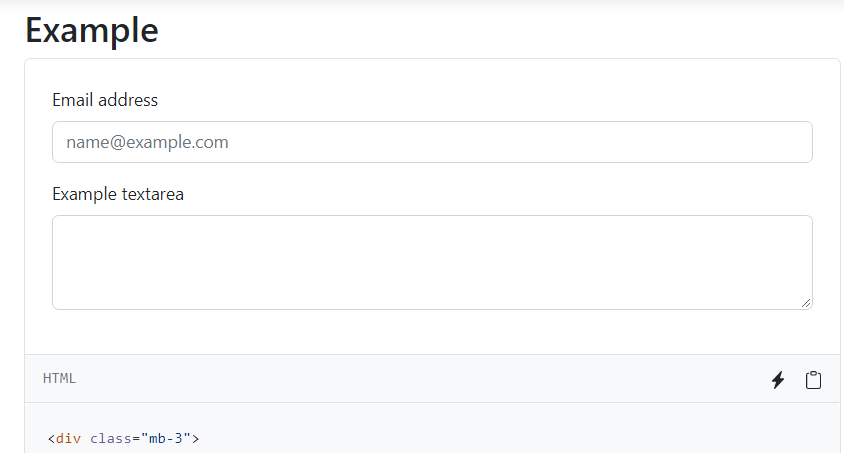


Ahora si demos las funcionalidades a los botones, iniciemos con crear nuevo producto.

Vamos a la carpeta vistas y creamos dos vistas una para proveedor y otra para producto con el nombre de **nuevoProducto y nuevoProveedor.**

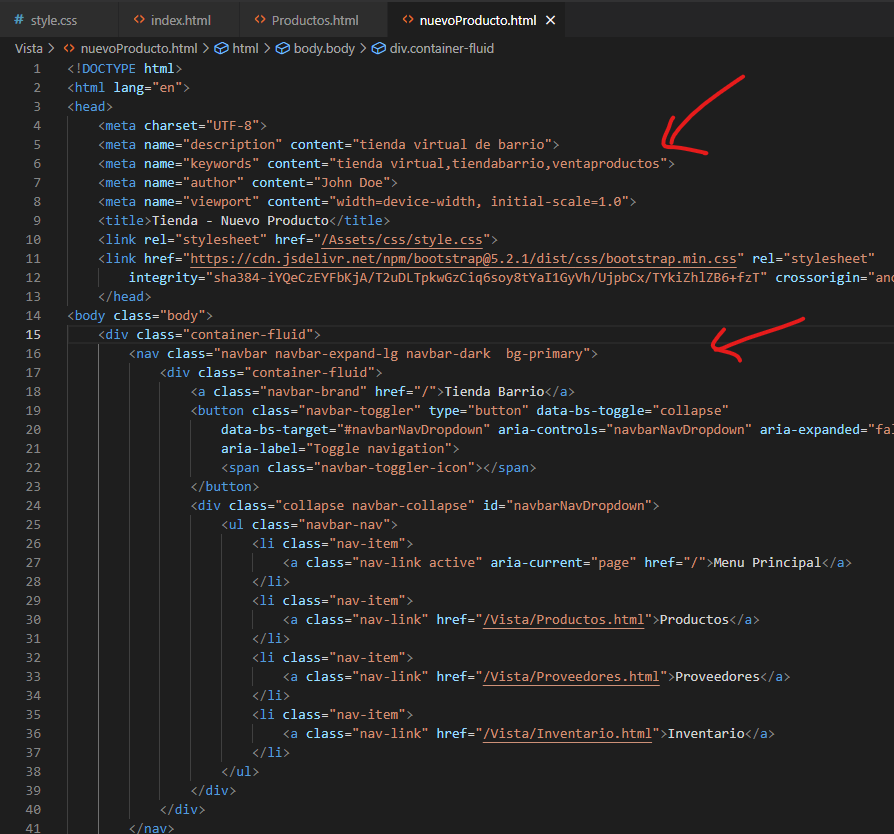


Abrimos nuevoProducto y creamos un formulario para ingresar los nuevos productos, vamos a **Bootstrap** y podemos **Form** y seleccionamos el que mas les guste en mi caso seleccione este.

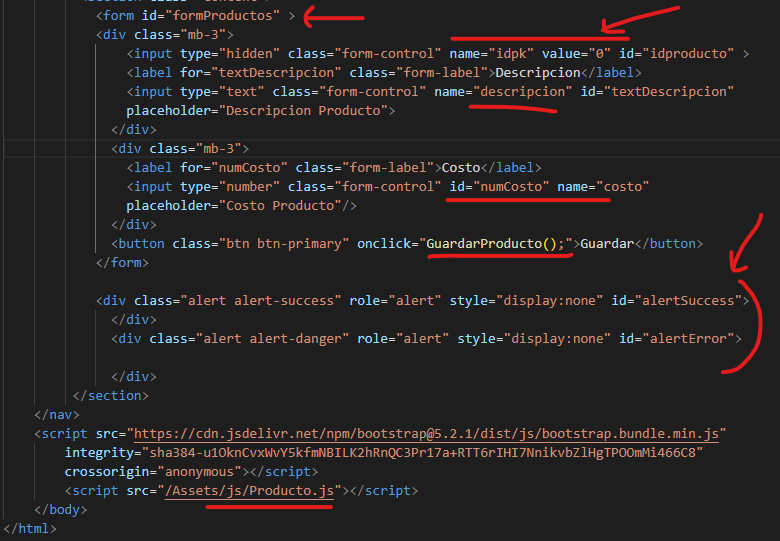


Copiamos el código y lo pegamos en nuestro archivo nuevoProducto nos quedaría así.

Tener en cuenta que hay que replicar el código **html de head y nav y section. Ej:**

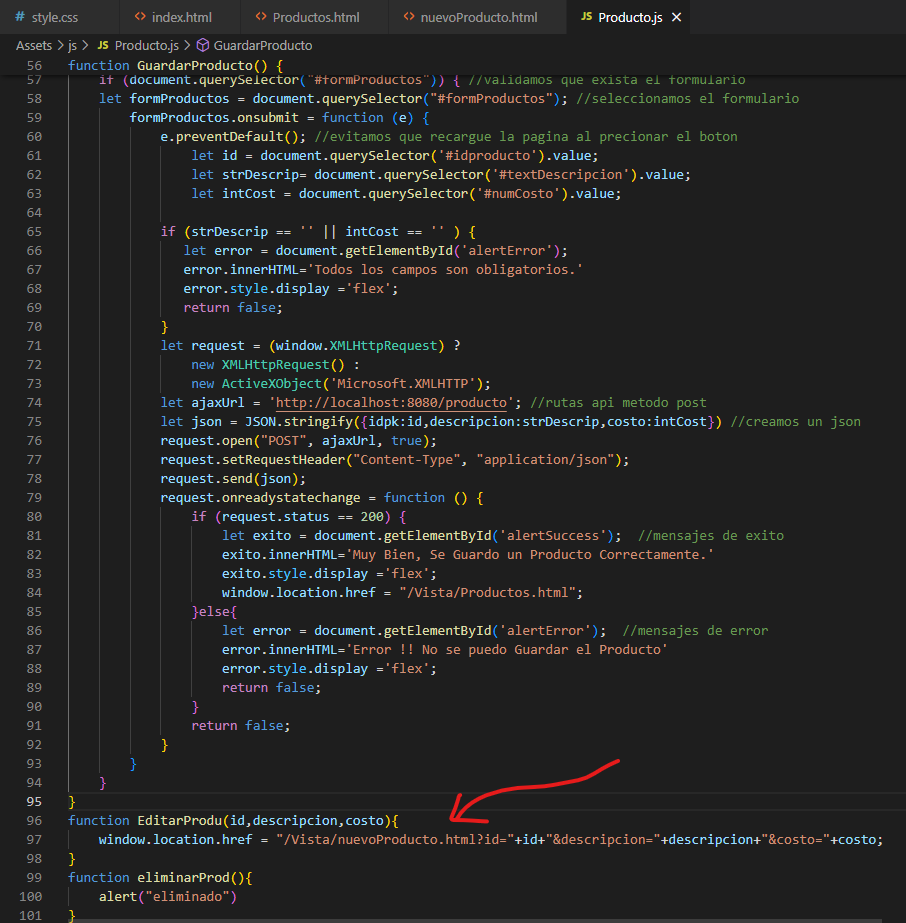
****

Ahora si nuestro formulario: También vamos a Bootstrap y buscamos **Alerts** para mostrar unos mensajes en caso tengamos error o éxito así se vería nuestro código.



Listo ahora que ya tenemos el formulario capturemos los datos para enviarlos a nuestra Api nos vamos a nuestro archivo producto.js ponemos el siguiente código en las funciones así se verían nuestras funciones de **guardar, eliminar y actualizar.**

**Función guardar producto y actualizar**

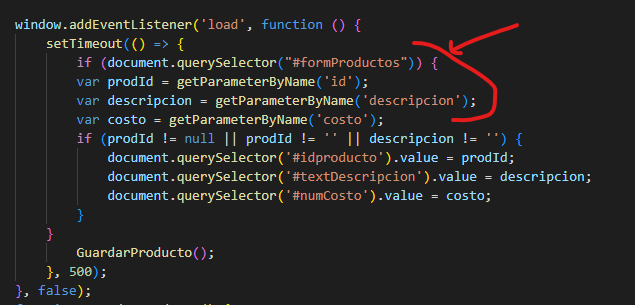


Este método funciona de la siguiente manera cuando presionamos nuevo producto no enviamos Id del producto así que ya sabemos que es un producto nuevo, pero si enviamos un id es una actualización. También tenemos que agregar este método que nos recuperar los datos para actualizar. Te señalo la ruta con los parámetros para actualizar un producto.

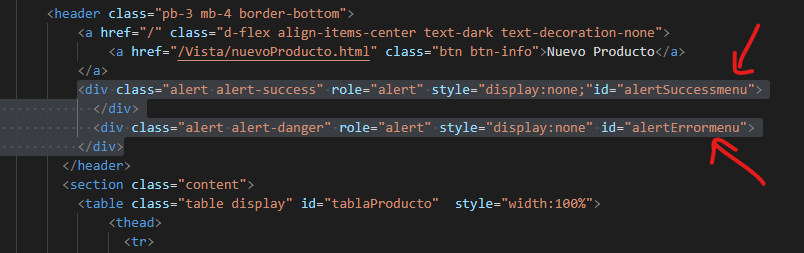


Entonces al agregarlo nos quedaría así en nuestro código.

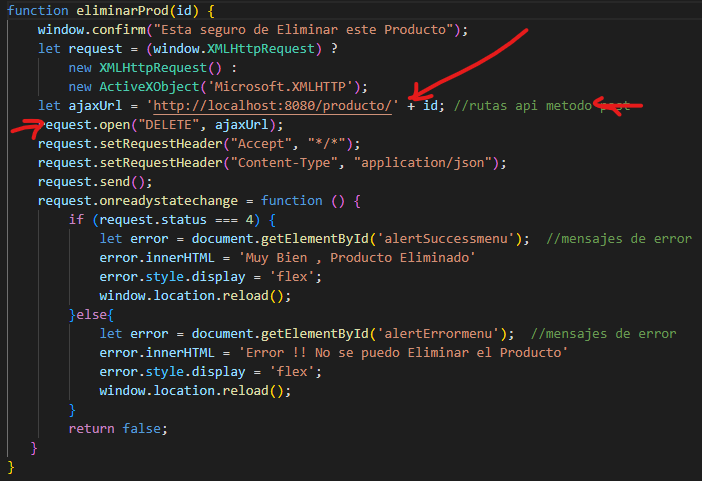
Creamos un evento a parte en el documento para que inicie retrasado y verifique si hay parámetros, si los hay los pone en el formulario para actualizar. Y si no, no pasa nada mas



Listo ahora seguimos con el método de eliminar el producto, es casi el mismo código solo cambia el endpoint de conexión por DELETE así se vería nuestro código. en nuestro archivo producto.html agregamos lo siguiente alertas.

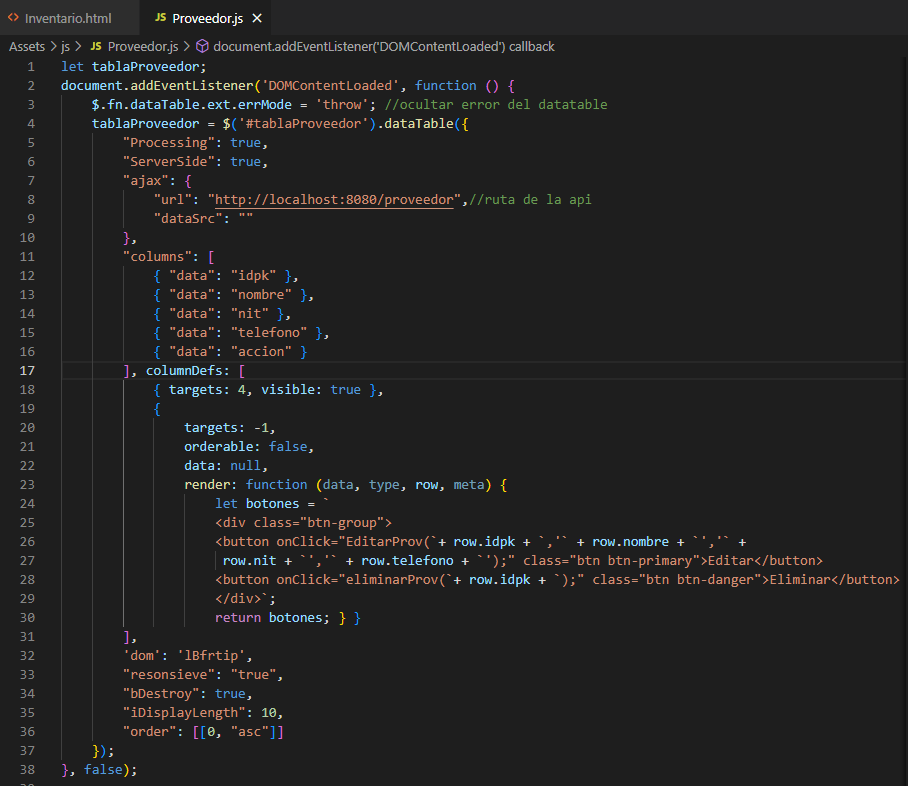


Bueno después de esto si procedemos con el método para eliminar el producto



Listo tenemos completo el **CRUD** para producto a partir de aquí te pongo como quedaron cada una de los demás métodos, siguiendo el mismo proceso.

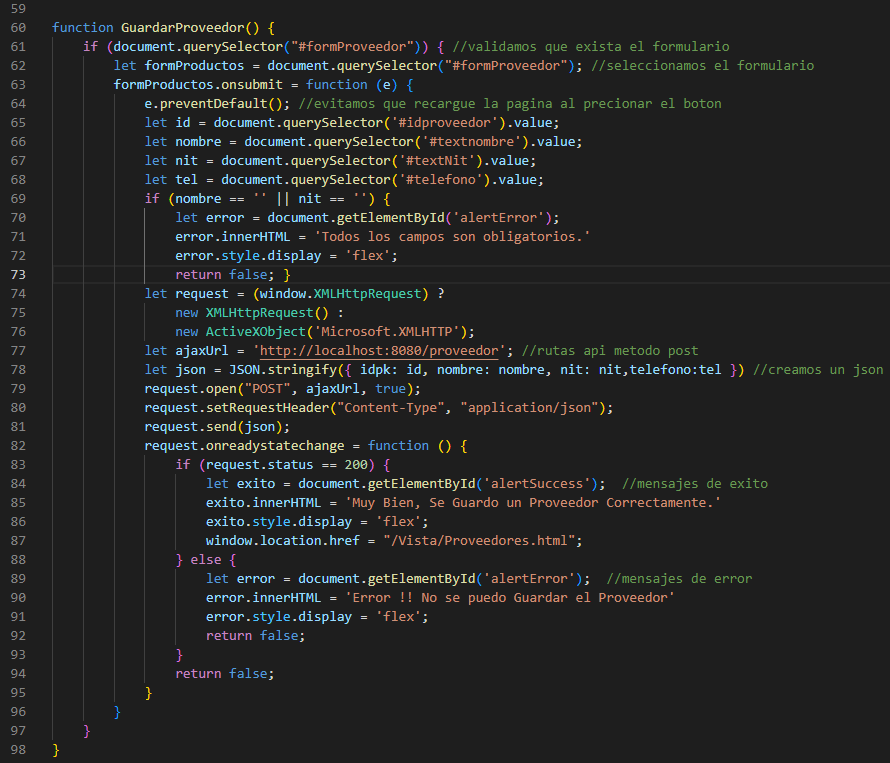
**Iniciemos con proveedor este es el método que carga la tabla proveedor:**

****

**Esta función inicia después de cierto tiempo en DOM y carga la función guardar.**

****

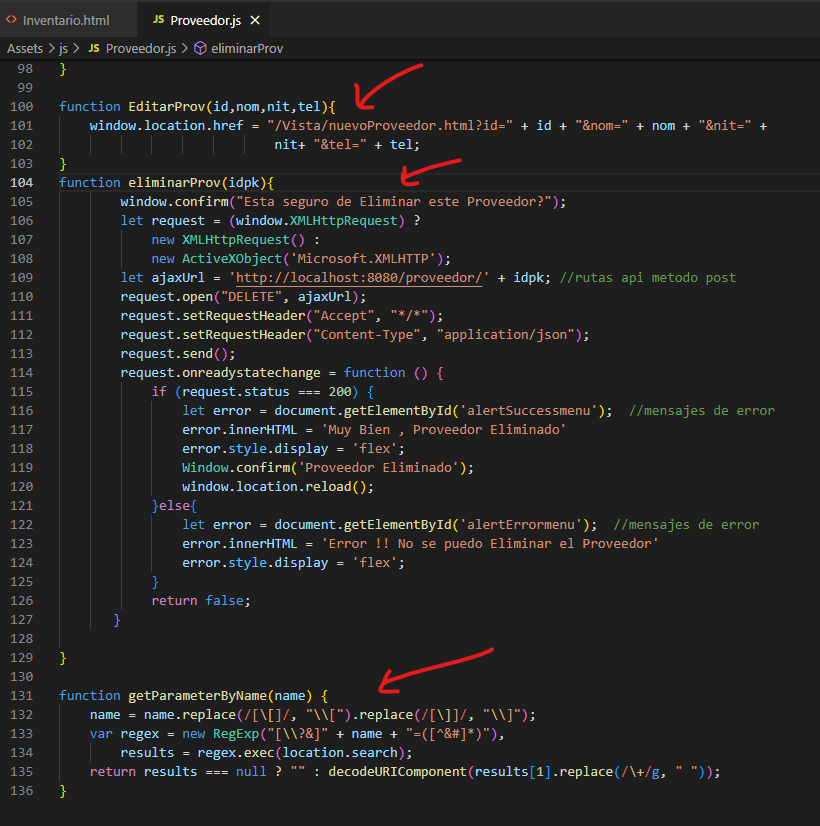
**Método Guardar Proveedor**

****

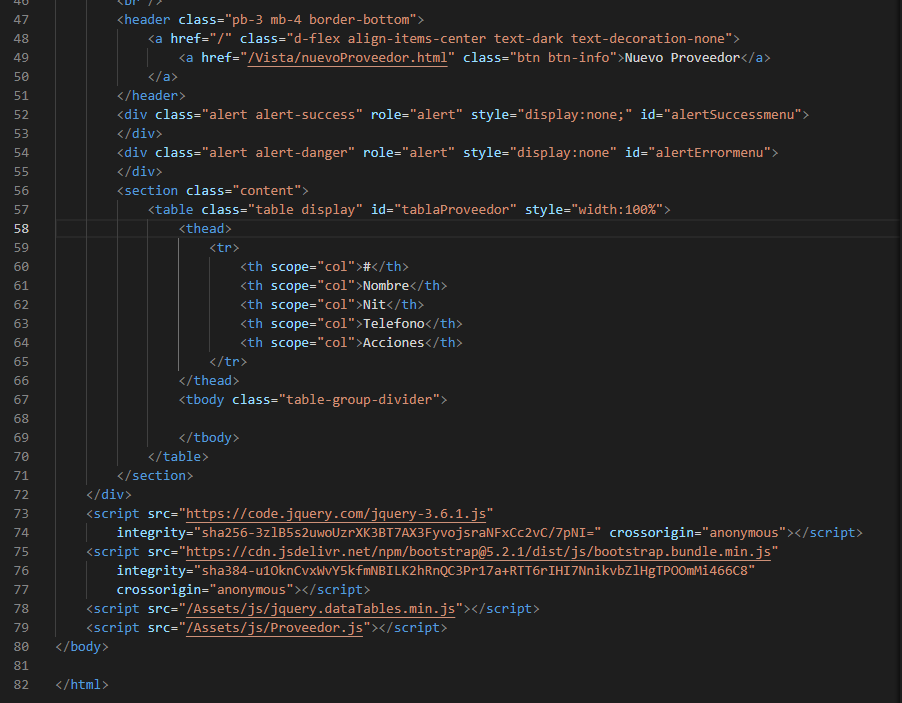
**Métodos EDITAR (reenvía los datos al formulario)**

**Método Eliminar**

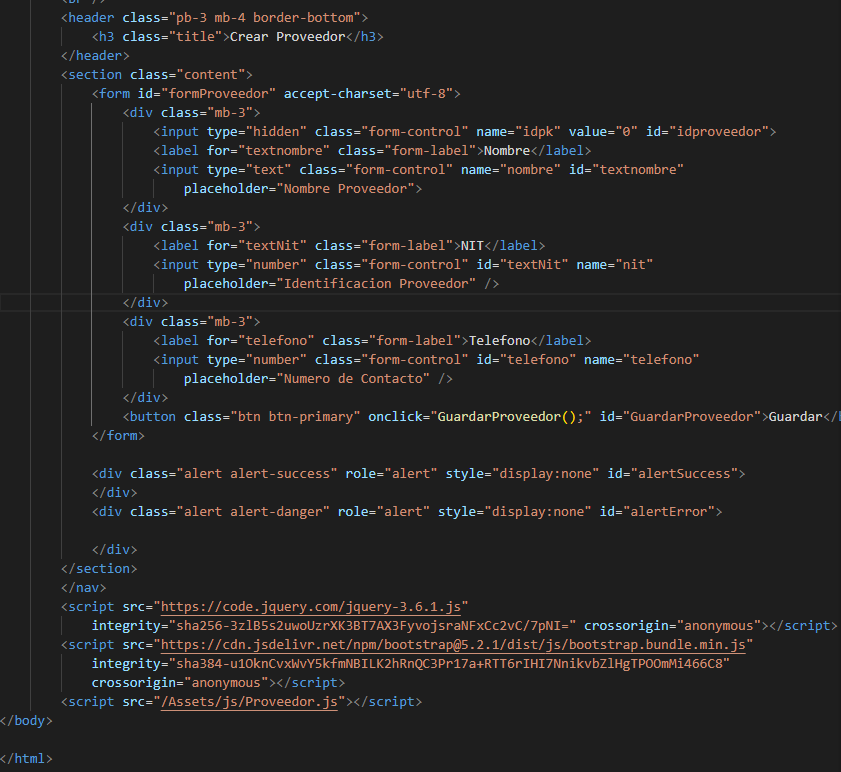
**Y el método que recura la información por medio get url**

****

**Listo ahora les mostrare el provedor.html, solo el header y el cuerpo lo de arriba es lo mismo para todos.**

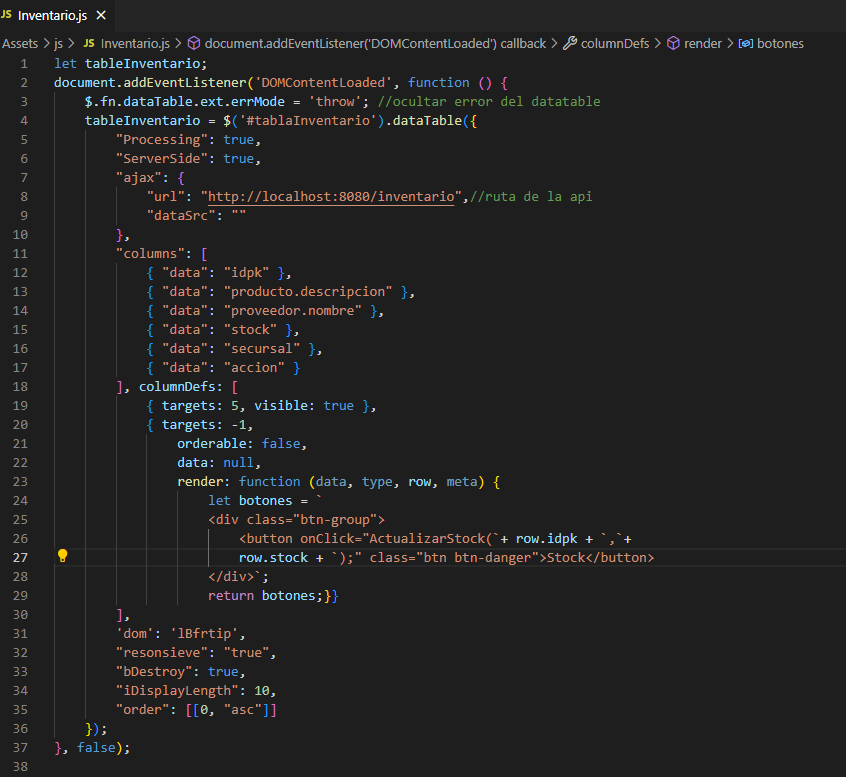
****

**Vista de nuevo proveedor:**

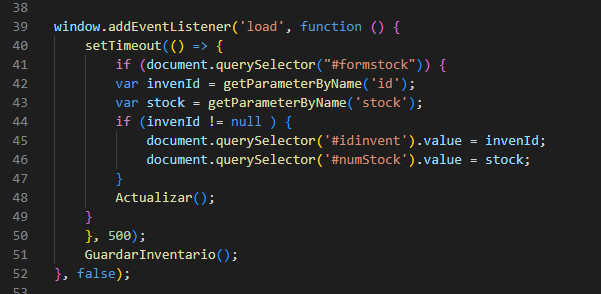
****

**Listo esto es todo por parte de Proveedor ahora seguimos con inventario la cual se encarga de recuperar los productos y proveedores para asignarse a un inventario.**

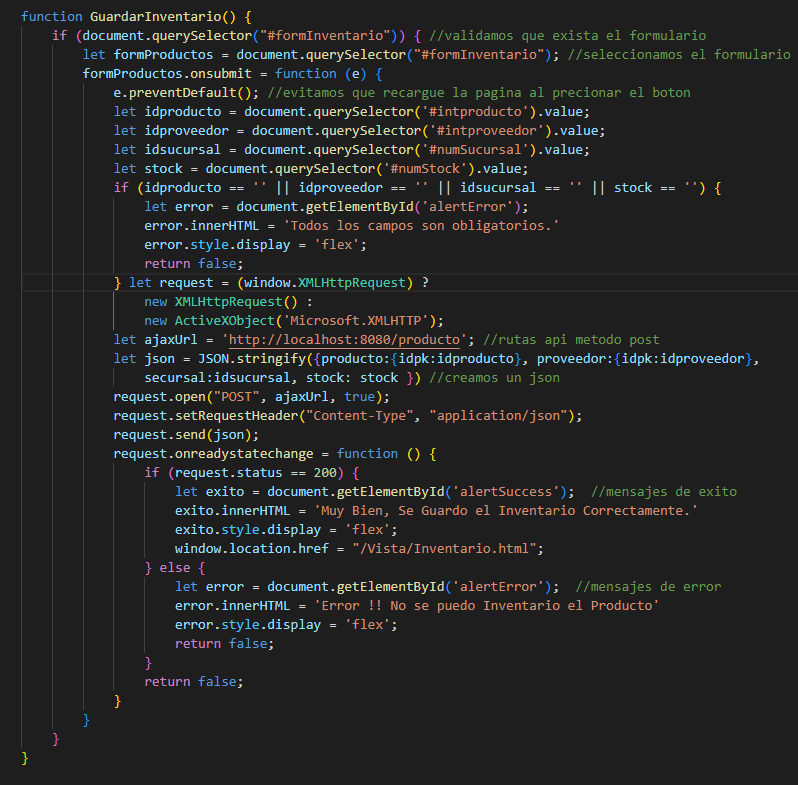
**Método que carga la tabla:**

****

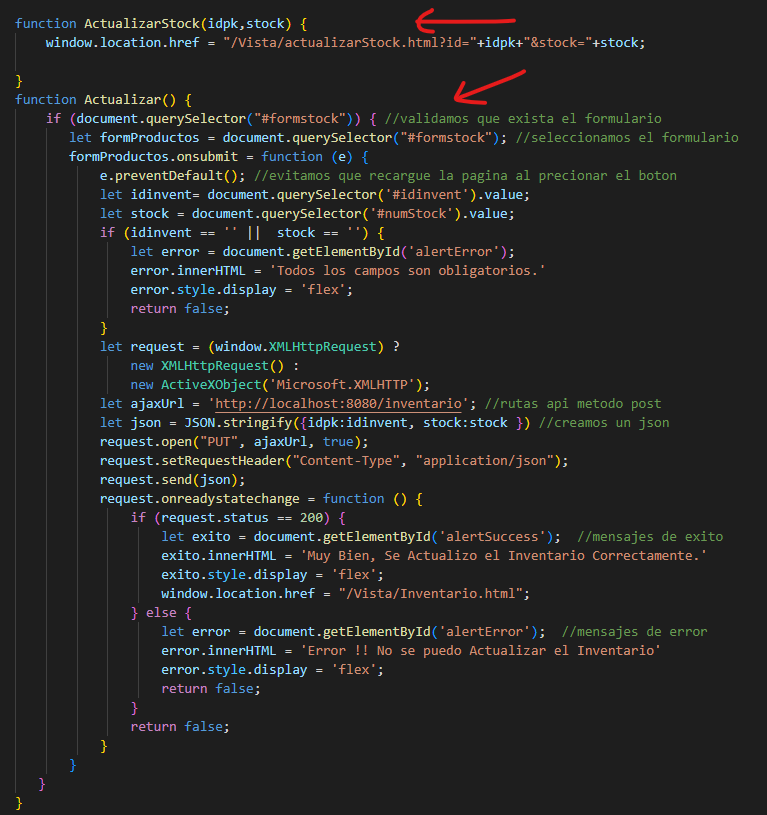
**Función que nicia después de cargar el DOM y llama a dos métodos que es actualizar y guardar.**

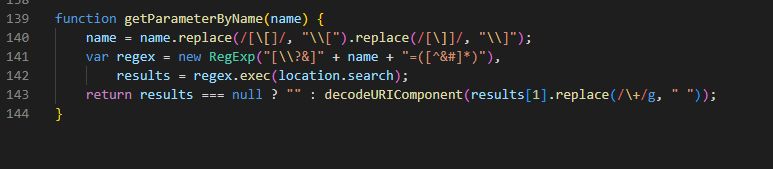
****

**Método guardar**

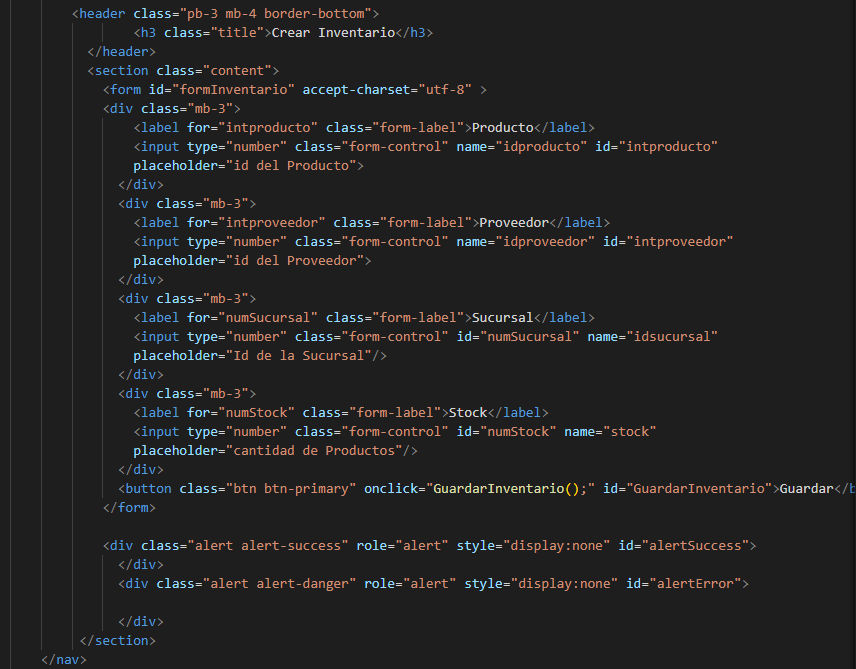
****

**Método actualizar y el la función que redirige los datos para actualizar.**

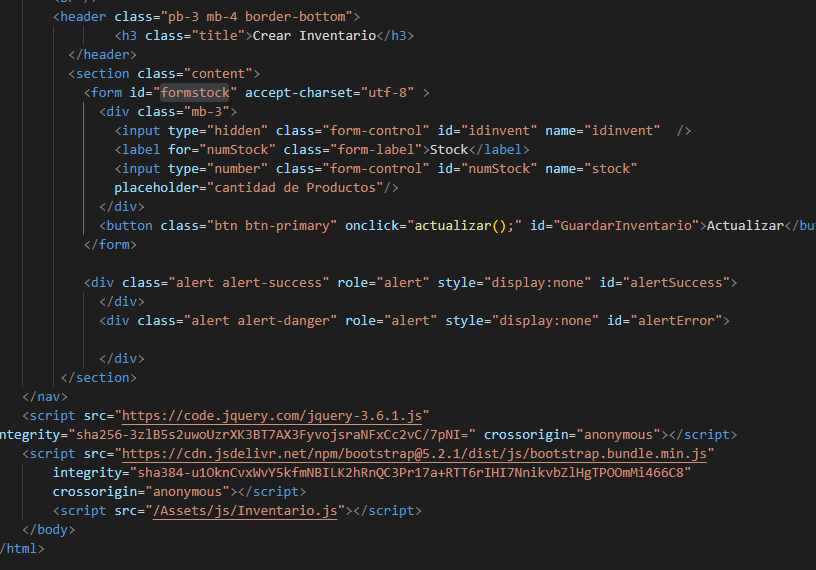
****

****

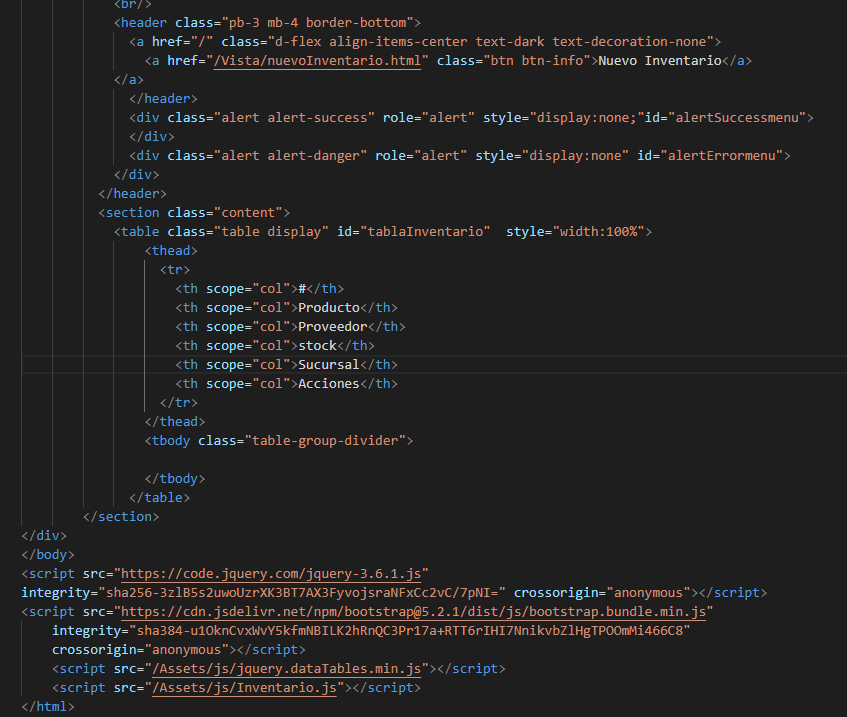
**Vistas nuevo inventario**

****

Vista actualizar stock



Vista inventario.html

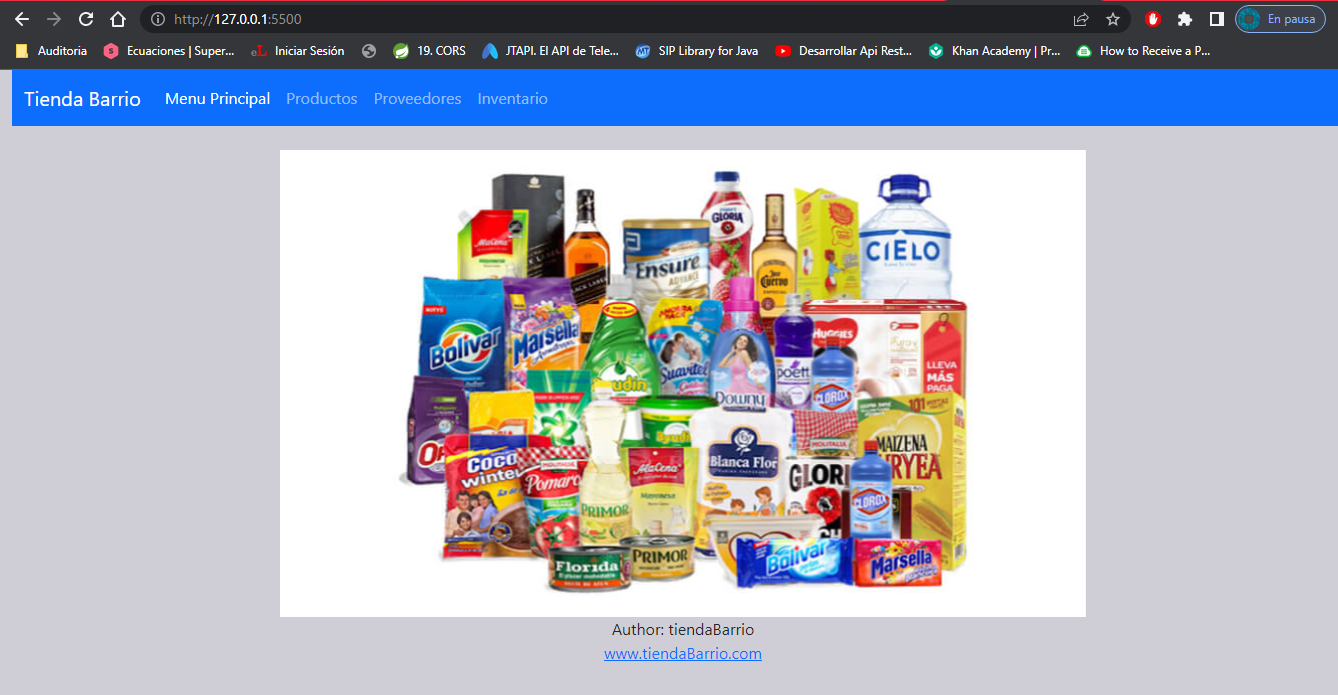


Así quedaría nuestro proyecto final, aunque no utilizamos lógica para controlador y tampoco lógica para modelo ya que toda venia desde el API. Aunque se pueden realizar.

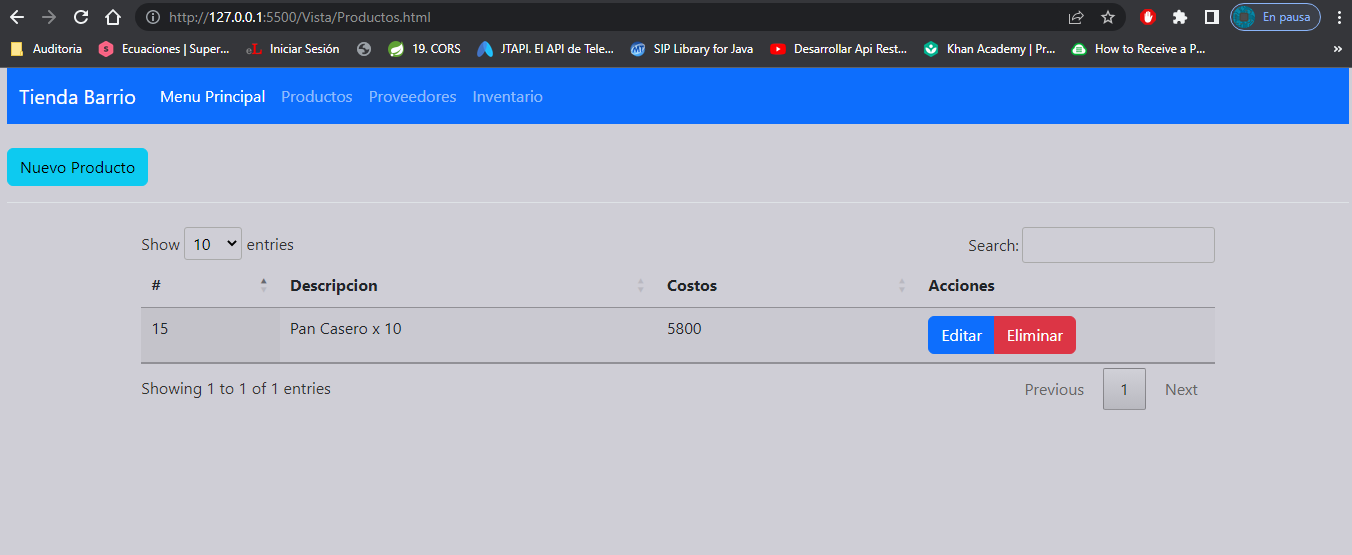
Pero en este caso no se utilizó para ser más práctico.



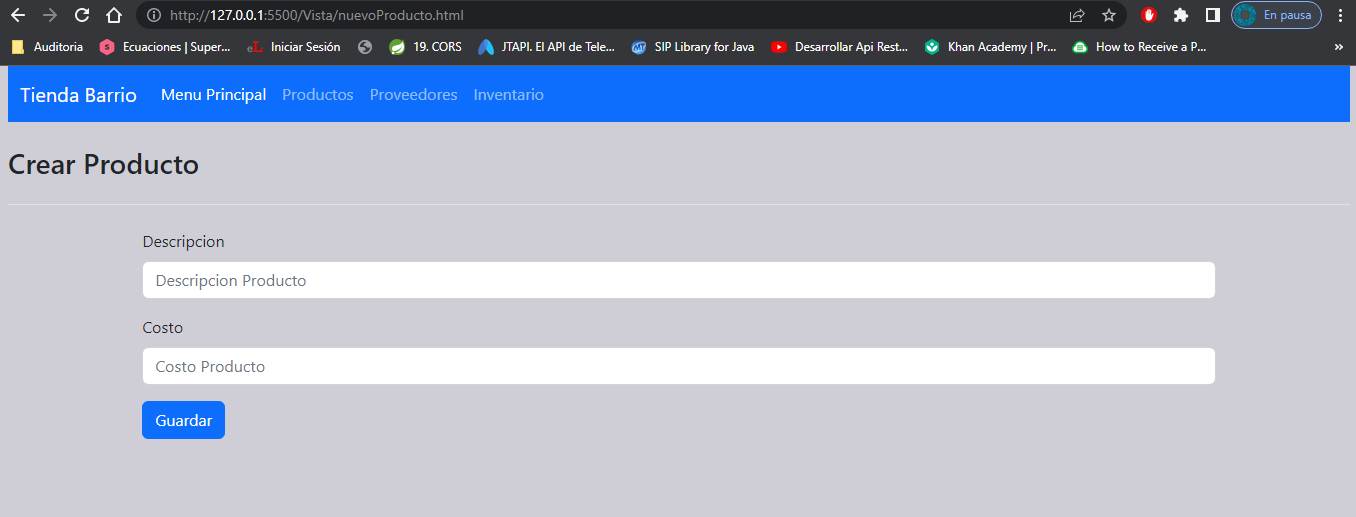
Menu Princpal :



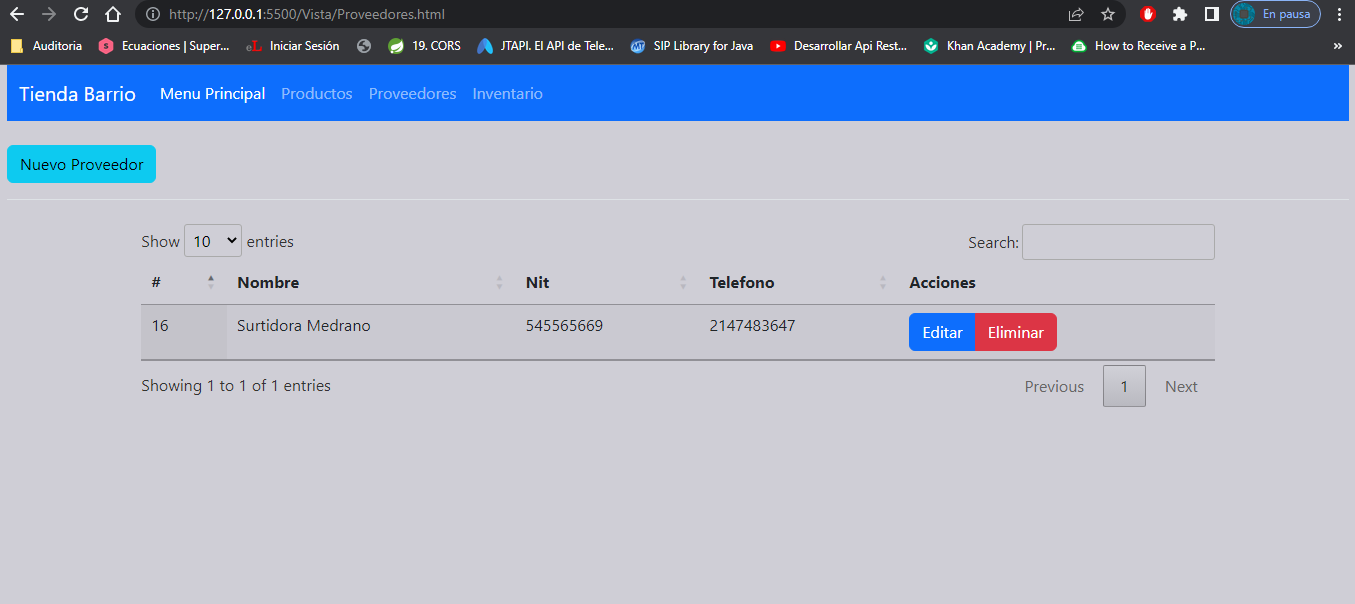
Productos :



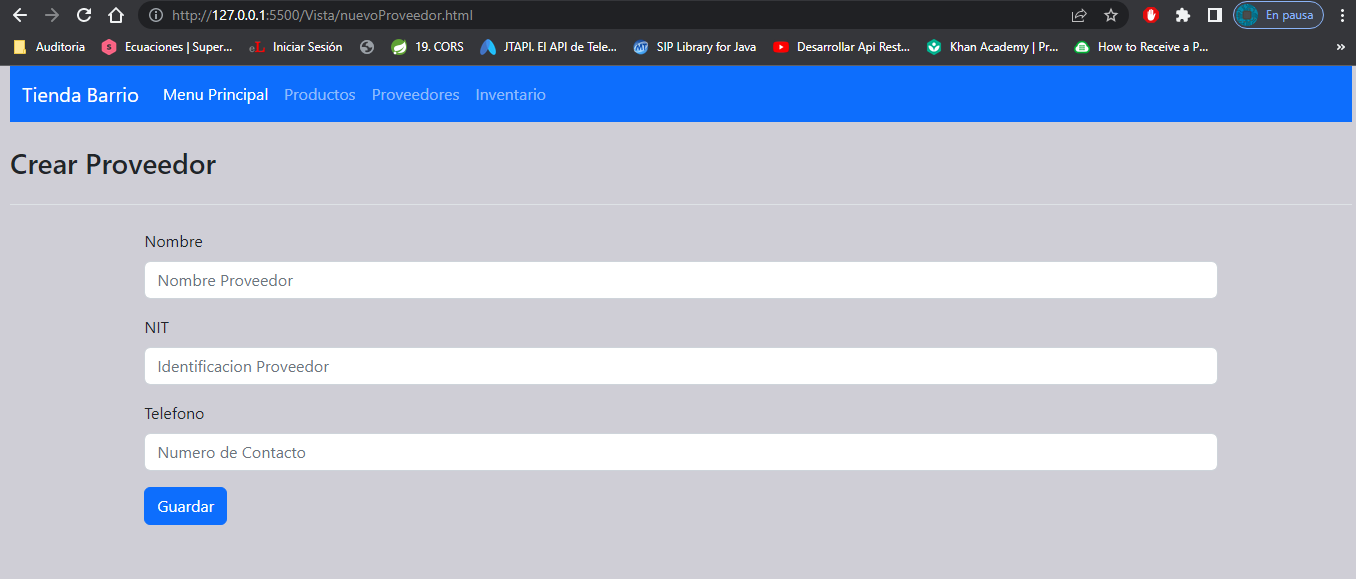
Nuevo Producto:

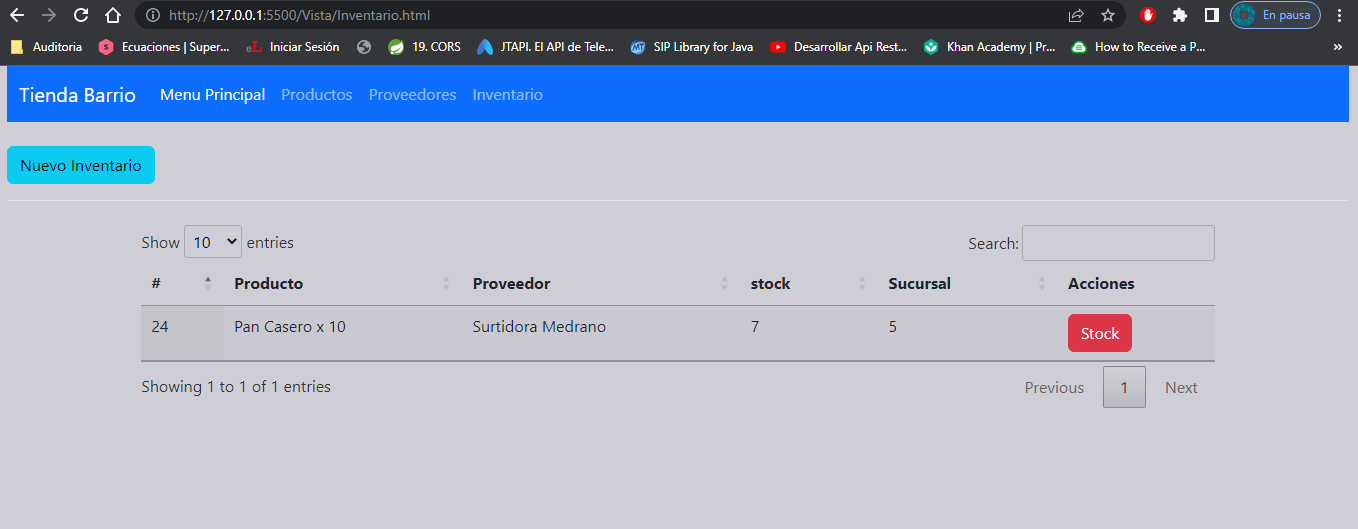


Proveedor:

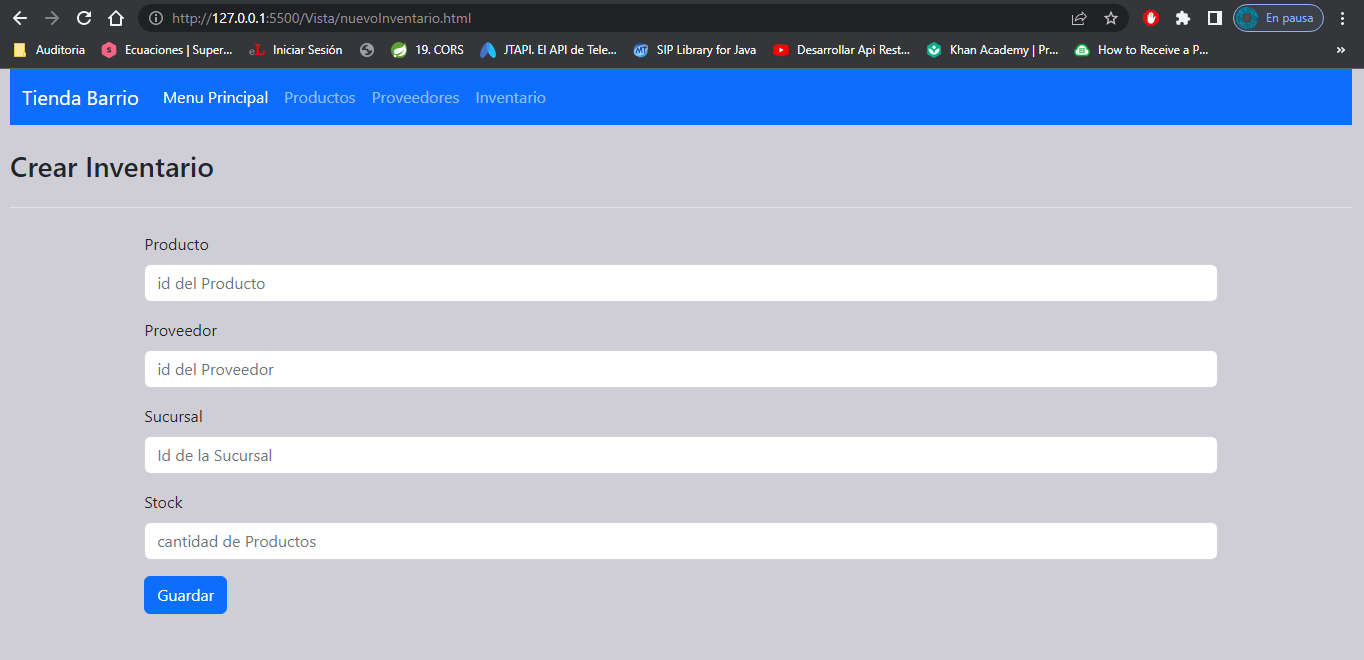


Nuevo proveedor:

Inventario :



Nuevo inventario:



Actualizar Stock:



Listo esto sería todo, ya tendríamos funcionando nuestro proyecto.