Para facilitar el desarrollo de este proyecto, se utilizaron varias herramientas y tecnologías:

1. **Bootstrap**:
   * Se ha utilizado Bootstrap para aplicar estilos CSS y mejorar la apariencia de la interfaz de usuario. Las clases y componentes de Bootstrap se usan para crear una experiencia visualmente atractiva y funcional en la aplicación.
2. **FakestoreAPI**:
   * **FakestoreAPI** se utiliza para obtener la lista de productos y categorías. Esta API permite acceder a datos de productos de manera sencilla para poblar el contenido de la aplicación.
3. **SweetAlert2**:
   * **SweetAlert2** se emplea para mejorar la interacción con el usuario mediante la visualización de mensajes y alertas. Se utiliza para mostrar confirmaciones y notificaciones sobre las acciones realizadas, como agregar productos al carrito o completar una compra.
4. **Funciones Asíncronas**:
   * Para consumir la API de **FakestoreAPI**, se utilizan funciones asíncronas (async/await). Esto permite realizar solicitudes HTTP de manera eficiente y manejar las respuestas de forma clara y ordenada. Las funciones asíncronas permiten que el código continúe ejecutándose mientras espera que se complete la solicitud, mejorando la eficiencia y la capacidad de respuesta de la aplicación.

El proyecto está estructurado en carpetas separadas para HTML, estilos y scripts. La carpeta de scripts es clave, ya que contiene el archivo scripts.js, que maneja toda la funcionalidad de la tienda en línea.

Para organizar mejor el código y facilitar su mantenimiento, se han separado en archivos JavaScript diferentes la configuración y la conexión a la API. Se ha dedicado el archivo constantes.js para almacenar todas las configuraciones y valores fijos que se utilizan en la aplicación, como las URL de la API. El propósito de este archivo es que pueda cambiar estos valores en un solo lugar sin tener que buscar en todo el código para lograr una mayor claridad. La parte ridícula se encuentra en el archivo api\_connection.js. Se encarga de manejar las funciones usadas para la comunicación con la API. El propósito de esto también es que la lógica de acceso a los datos esté claramente separada de la configuración, lo que hace que sea más sencillo desarrollar y depurar. También hace que sea más fácil para los desarrolladores colaborar, ya que cada uno puede trabajar en un aspecto diferente del proyecto sin interferir con el desarrollo de los demás

**Método de Pago**

Para implementar el método de pago en nuestra web, utilizamos la API de PayPal. Esta API permite a los usuarios pagar con tarjeta de crédito, débito o mediante su cuenta de PayPal.

Primero, creamos una cuenta de desarrollador en PayPal para obtener nuestras credenciales, que incluyen el **Client ID** y el **Secret**. Estas credenciales son necesarias para realizar las transacciones de manera segura.

Luego, integramos el SDK de PayPal en nuestro sitio web utilizando el **Client ID**. El SDK nos permite mostrar botones de pago en nuestra página y gestionar las transacciones de forma segura. Añadimos el SDK en el archivo HTML de nuestra aplicación.

Para mostrar el botón de PayPal, simplemente lo insertamos en cualquier contenedor de nuestra página después de cargar el SDK. En nuestro caso, colocamos el botón dentro del modal del carrito de compras. De esta manera, cuando los usuarios revisan los productos en su carrito, pueden ver el botón de PayPal y proceder con el pago.

**Funcionamiento:** Cuando un usuario agrega productos al carrito y abre el modal, puede ver el total y hacer el pago usando el botón de PayPal. El botón procesa el pago, y una vez aprobado, captura la transacción y muestra una confirmación al usuario.