**📄 Documento de Requisitos – Tienda E‑commerce con Stripe**

**1. Requisitos Funcionales**

Estos son los comportamientos que la aplicación **debe cumplir** para que el flujo de compra funcione correctamente:

* **Gestión del carrito**
  + Agregar productos al carrito desde la galería.
  + Persistencia del carrito en localStorage para mantener el estado entre páginas y sesiones.
  + Visualización dinámica del número de productos en el header (“numerito”).
  + Eliminación individual de productos desde el carrito.
  + Eliminación total de productos con el botón “Eliminar todo”.
  + Actualización automática del numerito a 0 cuando el carrito queda vacío.
* **Renderizado dinámico**
  + Renderizado del carrito con productos, cantidades, subtotales y total.
  + Renderizado del ranking de productos más comprados (“Top productos”), con carrusel dinámico.
  + Visualización de la cantidad comprada en el ranking para evidenciar popularidad.
* **Integración con Stripe**
  + Uso de Stripe.redirectToCheckout con integración **client-only**.
  + Redirección al checkout usando el **price ID** de Stripe (no el product ID).
  + Soporte para mode: "payment" o mode: "subscription", según el producto.
  + Definición de successUrl y cancelUrl para manejar el flujo de compra.
  + Manejo de errores de Stripe (parámetros inválidos, integración no habilitada, etc.).
* **Persistencia y navegación**
  + El numerito del carrito se actualiza automáticamente al volver al home, sin necesidad de agregar un nuevo producto.
  + El ranking de productos se alimenta de las compras realizadas y se guarda en localStorage.

**2. Requisitos No Funcionales**

Estos son criterios de calidad, arquitectura y buenas prácticas que garantizan la robustez del sistema:

* **Arquitectura y modularidad**
  + Separación del código en módulos (dom.js, api.js, carrito.js, etc.) para escalabilidad y mantenibilidad.
  + Uso de funciones reutilizables y defensivas para evitar duplicación de lógica.
* **Validaciones y robustez**
  + Validación defensiva: comprobar la existencia de elementos DOM antes de manipularlos (ej. if (DOM.numerito)).
  + Manejo de errores en peticiones fetch a la API de Stripe.
  + Mensajes de advertencia en consola cuando un elemento no está presente.
* **Seguridad**
  + Uso obligatorio de **HTTPS** en producción para cumplir con Stripe y proteger datos sensibles.
  + Uso de claves públicas y secretas de Stripe de forma diferenciada (publicK en cliente, secretK en servidor).
* **Persistencia**
  + Uso de localStorage para mantener el estado del carrito y el ranking de productos.
  + Recuperación automática del estado al cargar la página.
* **Rendimiento**
  + Uso de DocumentFragment para optimizar inserciones en el DOM.
  + Duplicación de ítems en el carrusel para lograr loop infinito sin recargar datos.
* **Adaptabilidad**
  + Renderizado adaptado según la página:
    - En el **home**, el ranking se inserta en el contenedor principal de la galería.
    - En el **carrito**, el ranking se inserta en wrapeC (contenedor principal del carrito).
  + Diseño responsivo y consistente en desktop y móvil.

**3. Conclusión**

La tienda e‑commerce implementa un flujo completo de compra con carrito persistente, ranking de productos más comprados y pasarela de pago con Stripe. La modularidad del código, las validaciones defensivas y la integración segura con Stripe garantizan que el sistema sea **escalable, mantenible y confiable**.