GUÍA PASO A PASO

Campo extra en el checkout de Shopify

Una checkout UI extension completa con un campo de referencia de compra: el archivo de configuración, el componente, el metafield del carrito que se copia al pedido y los pasos para desplegarla. Con la regla de Shopify Plus y la checklist.

PARA	Apps de Shopify que agregan un campo al checkout de una tienda Plus
PROBADO CON	Código validado con las herramientas de Shopify (@shopify/ui-extensions 2026.1.1 y el esquema de Admin GraphQL). No se probó en una tienda
ACTUALIZADA	12 de noviembre de 2026
DOCUMENTACIÓN	shopify.dev/docs/apps/build/checkout/fields-banners/add-field
ARTÍCULO	ecomlabs.dev/g/campo
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Qué vas a montar
Paso 01	Crear la extensión
Paso 02	shopify.extension.toml
Paso 03	El componente
Paso 04	La definición del pedido
Paso 05	Desplegar y activar
Al terminar	Checklist

ANTES DE EMPEZAR

Qué vas a montar

Una checkout UI extension que pinta un campo de texto, "referencia de compra", justo después del bloque de contacto del checkout, y guarda lo que escriba el cliente en un metafield del carrito. Con una definición de metafield de pedido, ese valor se copia al pedido. Necesitas una tienda en **Shopify Plus** (las extensiones en información, envío y pago solo existen ahí) y **Shopify CLI 3.85.1 o superior**, la versión que pide el tutorial oficial.

PASO 01

Crear la extensión

Dentro de la carpeta de tu app:

```
shopify app generate extension --template checkout_ui --name po-reference
```

El CLI crea `shopify.extension.toml`, `src/Checkout.jsx`, `locales/` y `package.json`. Para ver la extensión mientras desarrollas: `shopify app dev` y la tecla `p`. Si cambias el `.toml`, reinicia el comando.

PASO 02

shopify.extension.toml

El CLI genera el `uid` y la versión de API; deja la que escriba. Este es el contenido que importa:

```
api_version = "2026-10"

[[extensions]]
name = "po-reference"
handle = "po-reference"
type = "ui_extension"

[[extensions.targeting]]
module = "./src/Checkout.jsx"
target = "purchase.checkout.contact.render-after"

[[extensions.targeting.metafields]]
namespace = "my_app"
key = "po_reference"
```

El `target` decide el lugar: para un campo de envío usa `purchase.checkout.shipping-option-list.render-after`. `namespace` y `key` son tuyos, pero deben repetirse igual en el código y en la definición del pedido (Paso 04).

PASO 03

El componente

Reemplaza `src/Checkout.jsx`. Lo validamos con el validador de Shopify para extensiones de checkout, en el target `purchase.checkout.contact.render-after`:

```
import '@shopify/ui-extensions/preact';
import {render} from 'preact';
import {useState} from 'preact/hooks';

const NAMESPACE = 'my_app';
const KEY = 'po_reference';

export default function extension() {
  render(<Extension />, document.body);
}

function Extension() {
  const {appMetafields, applyMetafieldChange, instructions, i18n} = shopify;

  // The value saved earlier in this checkout, so the field survives a reload.
  const saved = appMetafields.value.find(
    (entry) =>
      entry.target.type === 'cart' &&
      entry.metafield.namespace === NAMESPACE &&
      entry.metafield.key === KEY,
  );
  const [value, setValue] = useState(String(saved?.metafield.value ?? ''));
  const [failed, setFailed] = useState(false);

  // Accelerated checkout (Apple Pay, Google Pay) can turn this off.
  if (!instructions.value.metafields.canSetCartMetafields) {
    return null;
  }

  async function save() {
    const text = value.trim();
    const result = await applyMetafieldChange(
      text === ''
        ? {type: 'removeCartMetafield', namespace: NAMESPACE, key: KEY}
        : {
            type: 'updateCartMetafield',
            metafield: {
              namespace: NAMESPACE,
              key: KEY,
              type: 'single_line_text_field',
              value: text,
            },
          },
    );
```

```
    setFailed(result.type === 'error');
  }

  function updateValue(event) {
    setValue(event.currentTarget.value);
  }

  return (
    <s-text-field
      label={i18n.translate('poLabel')}
      value={value}
      error={failed ? i18n.translate('poError') : undefined}
      onInput={updateValue}
      onBlur={save}
    />
  );
}
```

Los textos `poLabel` y `poError` van en `locales/en.default.json`, por ejemplo `{"poLabel": "Purchase order reference (optional)", "poError": "We couldn't save this. Try again."}`, y en un archivo por idioma, como `locales/es.json`. El campo guarda al salir (`onBlur`), no en cada tecla: demasiados cambios pueden limitar la extensión. No uses `applyAttributeChange`, que la referencia de la API marca como obsoleto. Con `shipping-option-list.render-after`, que se pinta una vez por tipo de compra, devuelve `null` salvo si `shopify.target.value.groupType` es `'oneTimePurchase'`, como el ejemplo oficial.

PASO 04

La definición del pedido

Shopify copia el metafield del carrito al pedido al crearse el pedido, solo si existe una definición de metafield de pedido con el mismo namespace y key y la capacidad *cart to order copyable*. Dos formas de crearla.

En el admin: crea una definición de metafield de **pedido**, tipo texto de una línea, namespace `my_app`, key `po_reference`, y activa *Copy values from cart metafields*.

Con la API (mutación validada contra el esquema de Admin GraphQL):

```
mutation {
  metafieldDefinitionCreate(
    definition: {
      name: "PO reference"
      namespace: "my_app"
      key: "po_reference"
      type: "single_line_text_field"
      ownerType: ORDER
      capabilities: { cartToOrderCopyable: { enabled: true } }
    }
  ) {
    createdDefinition { id namespace key }
    userErrors { field message code }
```

```
  }
}
```

Revisa siempre `userErrors`. La guía de migración de Shopify indica que la app necesita el scope `write_orders` para que los metafields lleguen al pedido; el tutorial no lo menciona, así que confírmalo en tu app.

PASO 05

Desplegar y activar

1. Despliega la versión: `shopify app deploy`. Con `--no-release` creas la versión sin publicarla.
2. En el admin de la tienda, en inglés: *Settings > Checkout > Customize checkout*. Elige el paso, pulsa *Add app block* y selecciona la extensión. Sin este paso, la extensión no se muestra.
3. Haz un pedido de prueba: llena el campo, completa la compra y abre el detalle del pedido. El valor aparece en la sección Metafields.

> **CUIDADO**
> No probamos esta extensión en una tienda: validamos el código y la mutación con las herramientas de Shopify, y la configuración sigue su repositorio de ejemplo. Haz el pedido de prueba antes de activarla para clientes. En pago acelerado (Apple Pay, Google Pay) el cambio puede fallar; por eso el código comprueba `canSetCartMetafields`.

AL TERMINAR

Checklist

- [ ] La tienda es Shopify Plus si el campo va en información, envío o pago.
- [ ] `namespace` y `key` coinciden en el `.toml`, en el código y en la definición del pedido.
- [ ] La definición del pedido tiene activa la copia desde el carrito, y la app tiene `write_orders`.
- [ ] El código comprueba `canSetCartMetafields` y guarda al salir del campo.
- [ ] El comerciante agregó el bloque de la app en el editor de checkout.
- [ ] Un pedido de prueba muestra el valor en Metafields.
- [ ] Sin Plus: pide el dato en el carrito del tema con un `input` llamado `attributes[nombre]`.