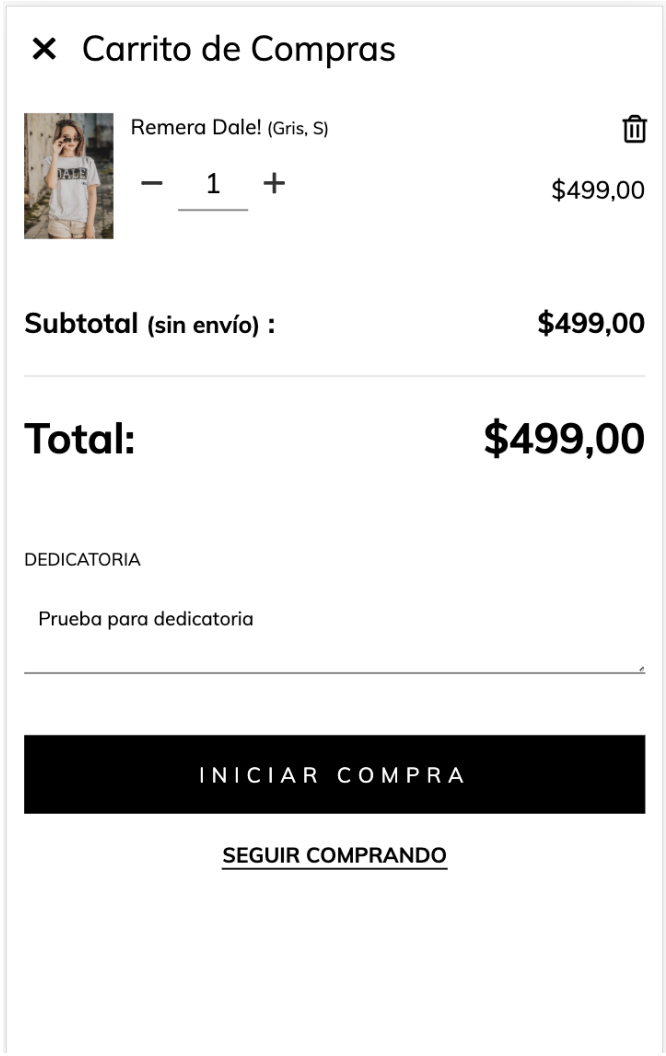
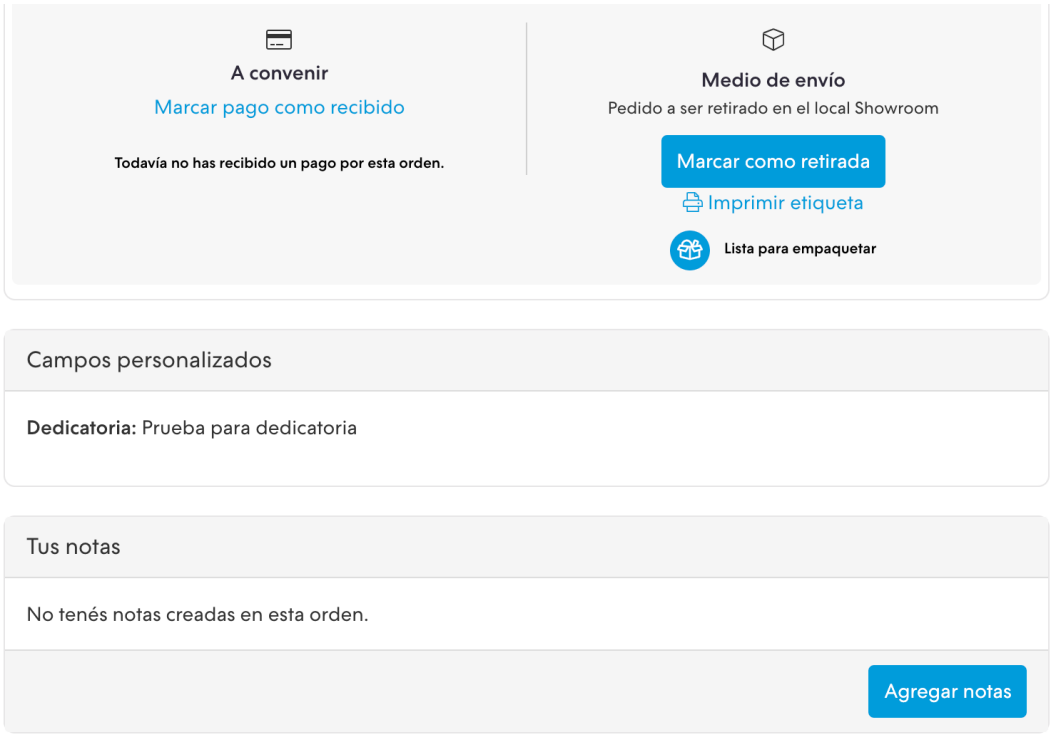
Campos personalizados en el carrito

En este tutorial vamos a ver como agregar campos personalizados dentro del carrito de compras. Puede servir por ejemplo para productos en los cuales se necesite una dedicatoria o aclarar información adicional respecto al envío.



Esta información luego se verá en la orden del Administrador nube bajo el título de “Campos personalizados”.



HTML

1. Creamos la carpeta forms dentro de la carpeta snipplets. Acá necesitamos sumar un archivo nuevo con el nombre form-input.tpl que vamos a usar para cada input en el detalle del producto.

```
{# /*============================================================================
==========
  #Form input
==============================================================================
=======*/

#Properties

#Group
    //input_group_custom_class for custom CSS classes
#Label
    // input_label_id for ID
    // input_for for label for
    // input_label_custom_class for custom CSS classes
    // input_label_text for label text
#Prepend
    // input_prepend_content to add content before input
#Container (Only if has prepend or append)
```

```
#container (div) if has prepend or append
    // form_control_container_custom_class for container custom class.
E.g: col
#Input
    // Can be text_area or input
    // input_type to define type (text, tel, number or passowrd)
    // input_id for id
    // input_name for name
    // input_value for val
    // input_placeholder for placeholder
    // input_custom_class for custom CSS classes
    // input_rows for textarea rows
    // input_data_attr for data attributes
    // input_data_val for input_data_attr value
    // input_aria_label for aria-label attribute
#Append
    // input_append_content to add content after input
#Alerts
    // input_form_alert to insert alerts
#}


    {% if input_label_text %}
        {{ input_label_text }}
    {% endif %}
    {% block input_prepend_content %}
    {% endblock input_prepend_content %}
    {% if input_append_content or input_prepend_content %}


    {% endif %}
    {% if text_area %}
```

```
    {% else %}
```

```
    {% endif %}
```

```
    {% if input_append_content or input_prepend_content %}


    {% endif %}
    {% block input_append_content %}
    {% endblock input_append_content %}
    {% if input_help %}


            {% block input_help_text %}{% endblock input_help_text %}


    {% endif %}
    {% block input_form_alert %}
    {% endblock input_form_alert %}
```

2. Necesitamos agregar un input dentro del carrito de compras en el archivo cart-totals.tpl dentro de la carpeta snipplets, puede ser que en tu diseño tengas que hacerlo en el template cart.tpl. Lo importante es que se encuentre dentro del carrito, y si estás usando la etiqueta de HTML

para el este, ubicarlo ahí dentro.

En el theme Base podemos llamarlo de la siguiente forma:

```
{# Custom cart inputs #}

{% embed "snipplets/forms/form-input.tpl" with{text_area: true, input_n
ame: 'custom[Dedicatoria]', input_label_text: 'Dedicatoria' | translate
 } %}
{% endembed %}
```

El atributo name con la propiedad “custom[X]”, hace que el campo sea enviado al checkout y luego aparezca en la orden. La X puede ser reemplazada por cualquier propiedad que necesites, en este caso “Dedicatoria”.

CSS

Requisito:

Tener agregados en tu diseño las clases helpers. Podés seguir este este pequeño tutorial para hacerlo (simplemente es copiar y pegar algunas clases, no toma más de 1 minuto).

1. Agregamos el siguiente SASS de colores en **style-colors.scss.tpl** (o la hoja de tu diseño que tenga los colores y tipografías de la tienda). Recordá que las variables de colores y tipografías pueden variar respecto a tu diseño:

```
{# This mixin adds browser prefixes to a CSS property #}


@mixin prefix($property, $value, $prefixes: ()) {
  @each $prefix in $prefixes {
      #{'-' + $prefix + '-' + $property}: $value;
  }
    #{$property}: $value;
}


{# /* // Forms */ #}


input,
textarea {
  font-family: $body-font;
}


.form-control {
  display: block;
  padding: 10px 8px;
  width: 100%;
  border: 0;
  border-bottom: 1px solid rgba($main-foreground, .5);
  -webkit-appearance: none;
  -moz-appearance: none;
  appearance: none;
  color: $main-foreground;
  background-color: $main-background;
  &:focus{
```

```
      outline: 0;
    }
    &-inline{
      display: inline;
    }
  }


.form-control::-webkit-input-placeholder {
  color: $main-foreground;
}
.form-control:-moz-placeholder {
  color: $main-foreground;
}
.form-control::-moz-placeholder {
  color: $main-foreground;
}
.form-control:-ms-input-placeholder {
  color: $main-foreground;
}
```

2. Agregar los estilos dentro del archivo **static/style-critical.tpl**

Si en tu diseño usas una hoja de estilos para el CSS crítico, vamos a necesitar agregar el siguiente código debajo dentro de la misma, pero si no es el caso podés unificar el CSS de los pasos 2 y 3 en un solo archivo.

```
{# /* // Forms */ #}


.form-group {
  position: relative;
  width: 100%;
}
.form-group .form-select-icon{
  position: absolute;
  bottom: 12px;
  right: 0;
  pointer-events: none;
}
.form-row {
  width: auto;
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -ms-flex-wrap: wrap;
  flex-wrap: wrap;
  margin-right: -5px;
  margin-left: -5px;
  clear: both;
}


.form-row > .col,
.form-row > [class*=col-]{
  padding-right: 5px;
  padding-left: 5px;
}


.form-label {
  display: block;
  font-size: 10px;
  text-transform: uppercase;
}
```

3. Agregar los estilos dentro del archivo **static/style-async.tpl**

```
{# /* // Margin and Padding */ #}


%element-margin {
  margin-bottom: 35px;
}


{# /* // Mixins */ #}


{# This mixin adds browser prefixes to a CSS property #}


@mixin prefix($property, $value, $prefixes: ()) {
  @each $prefix in $prefixes {
    #{'-' + $prefix + '-' + $property}: $value;
  }
  #{$property}: $value;
}


{# /* // Forms */ #}


.form-group{
  @extend %element-margin;
  .form-label{
    float: left;
    width: 100%;
    margin-bottom: 10px;
  }
  .alert{
    margin: 10px 0 0 0;
  }
}


{# /* Disabled controls */ #}


input,
select
```

```
  select,
  textarea{
    &[disabled],
    &[disabled]:hover,
    &[readonly],
    &[readonly]:hover{
      background-color: #DDD;
      cursor: not-allowed;
    }
  }
```

Listo, ¡Ya podés mostrar campos personalizados en el carrito!