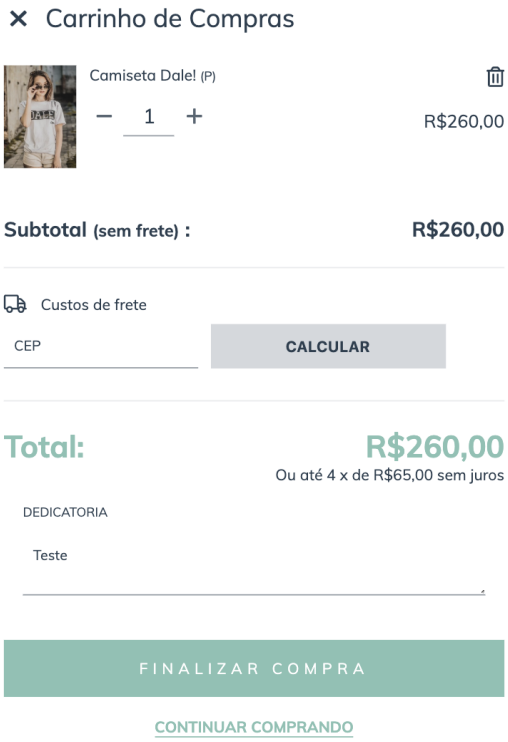
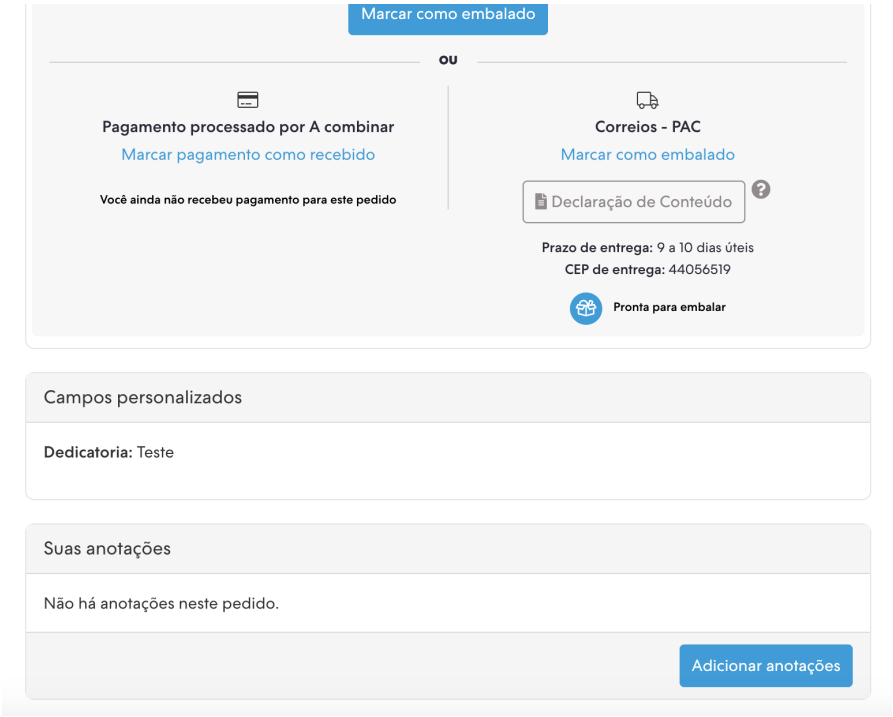
Campos personalizados no carrinho

Neste tutorial, veremos como adicionar campos personalizados dentro do carrinho de compras. Pode servir, por exemplo, para produtos em que é necessário uma dedicatória ou esclarecer informações adicionais sobre a remessa.



Essas informações são exibidas na página do pedido, no Administrador Nuvem, sob o título "Campos personalizados".



HTML

1. Criamos a pasta de forms dentro da pasta snipplets. Aqui, precisamos adicionar um novo arquivo com o nome form-input.tpl que usaremos para cada input nos detalhes do produto.

```
{# /*============================================================================
==========
  #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)
    // form_control_container_custom_class for container custom class
```

```
    // 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. Precisamos adicionar uma entrada ao carrinho de compras no arquivo cart-totals.tpl

dentro da pasta **snipplets**, pode ser que em seu design você tenha que fazer isso no template **cart.tpl**. O importante é que ele esteja dentro do carrinho e, se você estiver usando a tag HTML

, coloque-o lá.

No layout Base, podemos chamá-lo da seguinte 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 %}
```

O atributo name com a propriedade “custom [X]” faz com que o campo seja enviado para o checkout e, em seguida, aparece no pedido. O X pode ser substituído por qualquer propriedade que você precise, neste caso “Dedicatória”.

CSS

Requisito:

Ter adicionado helper classes em seu layout. Você pode seguir este pequeno tutorial para fazer isso (é só copiar e colar algumas classes, não leva mais que 1 minuto).

1. Adicionamos o seguinte SASS de cores em style-colors.scss.tpl (ou no stylesheet do seu layout que possui as cores e fontes da loja). Lembre-se de que as variáveis de cores e fontes podem variar em relação ao seu layout:

```
{# 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. Adicione os estilos no arquivo **static/style-critical.tpl**

Se em seu layout você usar um stylesheet para o CSS crítico, precisaremos adicionar o seguinte código abaixo, mas se não for o caso, você pode unificar o CSS dos passos 2 e 3 em um único arquivo.

```
{# /* // 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. Adicione os estilos no arquivo **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;
  }
}
```

Pronto, você já pode mostrar campos personalizados no carrinho!