← Volver a la lista de capítulos

Cómo añadir un total de cotización con Turbo Frames

Publicado el 15 de marzo de 2022

En este capítulo, agregaremos una barra adhesiva que contenga el total de la cotización. Este total se actualizará cada vez que creemos, actualicemos o eliminemos una partida.

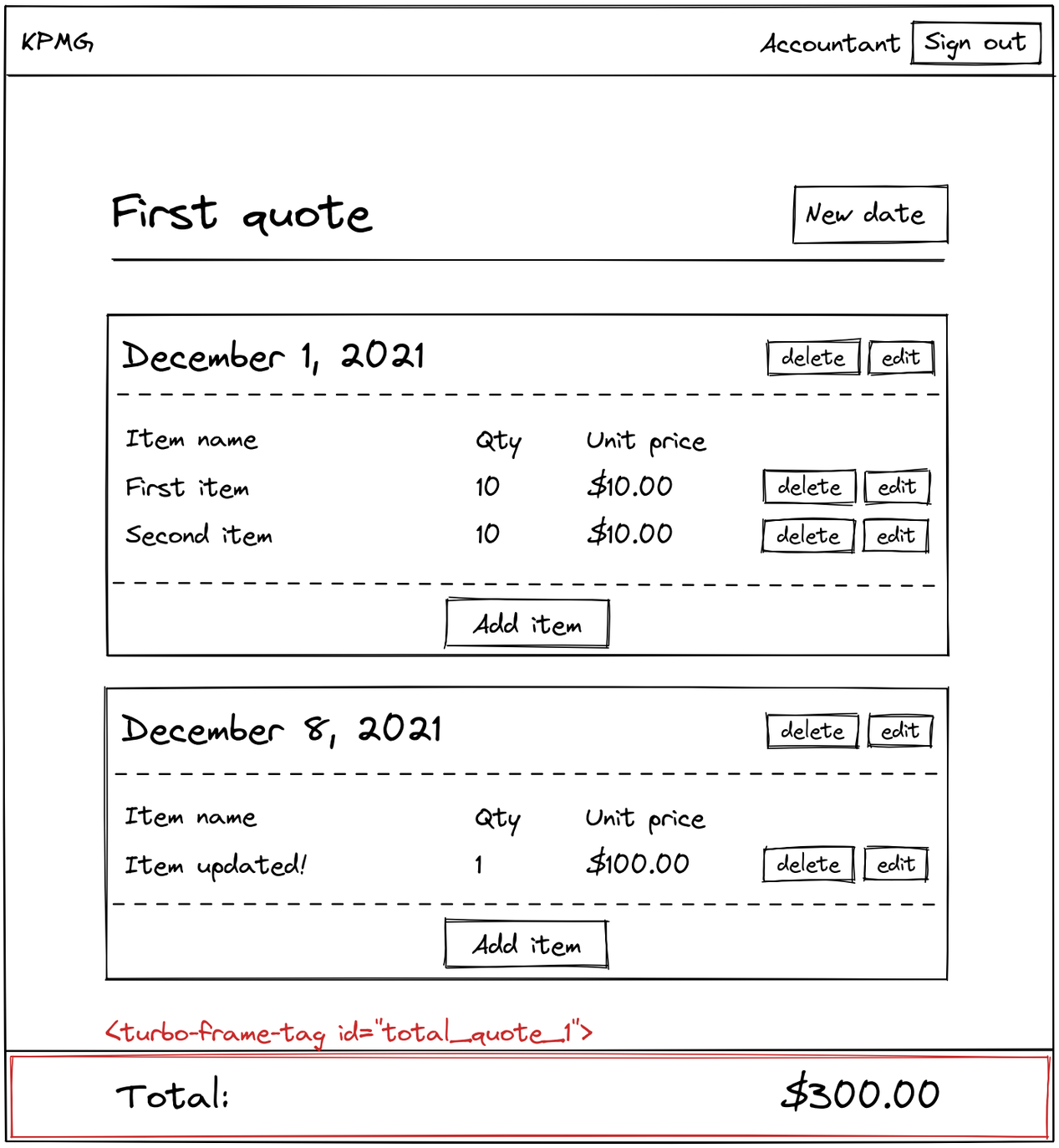
Lo que construiremos en este capítulo

En este último capítulo, finalizaremos nuestro editor de cotizaciones y este tutorial de Turbo Rails. Agregaremos una barra adhesiva a la `Quotes#show` página que contiene el monto total de la cotización.

Este importe total se actualizará cada vez:

- Se crea, actualiza o elimina una **partida**
- Se elimina la **fecha de un elemento de línea** , ya que la fecha del elemento de línea puede contener algunos elementos

Aquí hay un boceto de cómo se verá nuestro editor de cotizaciones:



Boceto de la cotización#mostrar página con el monto total de la cotización

Ahora que los requisitos están claros, ¡es hora de comenzar a codificar!

Nota : Ahora que tenemos más experiencia con Turbo Frames, analicemos cuándo usar una `<turbo-frame>` etiqueta o un `<div>` .

Debemos usar un Turbo Frame cuando necesitamos que Turbo intercepte clics en enlaces y envíe formularios por nosotros . Por ejemplo, el contenido del `line_items/_line_item.html.erb` parcial debe estar dentro de un Turbo Frame porque cuando hacemos clic en el botón "Editar" de un **elemento de línea** , queremos que Turbo reemplace el contenido de este Turbo Frame con el formulario extraído de la `LineItems#edit` página. Esta característica no funcionaría con un simple `div` .

Por otro lado, **no *necesitamos* un Turbo Frame cuando solo apuntamos a un id del DOM en una vista Turbo Stream** . Por ejemplo, no usamos un Turbo Frame para mensajes flash. En su lugar, usamos un `div` con un id de `flash` para *anteponer* nuevos mensajes. Podríamos haber usado un `div` con el id de `quotes` en la `Quotes#index` página en lugar de un Turbo Frame, porque el `quotes` id solo se usa en las vistas Turbo Stream para *anteponer* comillas a la lista. Nunca sirve para interceptar clics en enlaces y envíos de formularios.

Mi *preferencia personal* aquí es siempre usar etiquetas Turbo Frame, ya que hace evidente que la identificación se usa en algún lugar de una vista Turbo Stream.

💎 Trenes calientes

embargo, está perfectamente bien si no estás de acuerdo y prefieres usar una `div` cuando no necesitas un Turbo Frame.

Con esa sutil diferencia explicada, ¡continuemos con el tutorial!

Diseño de la barra de navegación fija

Antes de trabajar con Turbo, agreguemos la barra de navegación fija a nuestra `Quotes#show` página y tomémonos un tiempo para diseñarla:

```
<%# app/views/quotes/show.html.erb %>

<main class="container">
  <!-- All the previous code -->
</main>

<%= render "quotes/total", quote: @quote %>
```

Agreguemos el marcado para el parcial que contiene el total de la cotización:

```
<%# app/views/quotes/_total.html.erb %>

<footer class="quote-total">
  <div class="quote-total__inner container">
    <div>Total:</div>
    <div><%= number_to_currency quote.total_price %></div>
  </div>
</footer>
```

Necesitamos agregar el `#total_price` método en el `Quote` modelo para poder mostrar el monto total de una cotización:

```
# app/models/quote.rb

class Quote < ApplicationRecord
  # All the previous associations
  has_many :line_items, through: :line_item_dates

  # All the previous code

  def total_price
    line_items.sum(&:total_price)
  end
end
```

Como podemos ver en el código que acabamos de agregar, el precio total de una cotización es la suma de los precios totales de cada **línea de artículo** de esta cotización. Asegurémonos de implementar el `#total_price` método para una sola **línea de artículo** :

```
# app/models/line_item.rb

class LineItem < ApplicationRecord
  # All the previous code

  def total_price
    quantity * unit_price
  end
end
```

El precio total de un **artículo de una sola línea** es su cantidad multiplicada por su precio unitario. Agreguemos rápidamente algunas pruebas para los dos métodos que acabamos de implementar:

```
# test/models/line_item_test.rb

require "test_helper"

class LineItemTest < ActiveSupport::TestCase
  test "#total_price returns the total price of the line item" do
    assert_equal 250, line_items(:catering_today).total_price
```

```
  end
end
```

```
# test/models/quote_test.rb

require "test_helper"

class QuoteTest < ActiveSupport::TestCase
  test "#total_price returns the sum of the total price of all line item
    assert_equal 2500, quotes(:first).total_price
  end
end
```

Probemos en el navegador y ahora deberíamos ver el precio total de la cotización en nuestra página. Apliquemos estilo a nuestra barra fija con CSS antes de pasar a la siguiente sección:

```
// app/assets/stylesheets/components/_quote_total.scss

.quote-total {
  position: fixed;
  bottom: 0;
  width: 100%;

  font-size: var(--font-size-xl);
  font-weight: bold;
  background-color: var(--color-white);
  box-shadow: var(--shadow-large);

  padding-top: var(--space-xs);
  padding-bottom: var(--space-xs);

  @include media(tabletAndUp) {
    padding-top: var(--space-m);
    padding-bottom: var(--space-m);
  }

  &__inner {
    display: flex;
    align-items: center;
    justify-content: space-between;
  }
}
```

No olvidemos importar este archivo a nuestro archivo de manifiesto:

```
// app/assets/stylesheets/application.sass.scss

@import "components/quote_total";
```

Probémoslo en nuestro navegador y todo debería funcionar como se espera, excepto que necesitamos un poco más de margen en la parte inferior de nuestra `Quotes#show` página para que la **fecha del último ítem de línea** no se superponga con la barra fija que contiene el total de la cotización. Agreguemos una clase de utilidad para agregar un margen inferior cuando sea necesario:

```
// app/assets/stylesheets/utilities/_margins.scss

.mb-xxxxl {
  margin-bottom: var(--space-xxxxl);
}
```

No olvidemos importar este archivo dentro de nuestro archivo de manifiesto:

```
// app/assets/stylesheets/application.sass.scss
// All the previous code

// Utilities
@import "utilities/margins";
```

Ahora podemos usar esta clase de utilidad en nuestra `Quotes#show` vista:

```
<%# app/views/quotes/show.html.erb %>

<main class="container mb-xxxxl">
  <!-- All the previous code -->
</main>

<%= render "quotes/total", quote: @quote %>
```

Probemos en el navegador. El importe total de la cotización se muestra como se esperaba. Sin embargo, el total no se actualiza cuando creamos, actualizamos o eliminamos un **artículo de línea** . Tenemos el mismo problema cuando eliminamos una **fecha de artículo de línea** que contiene algunos **artículos de**

línea . Resolveremos estos problemas con las vistas de Turbo Stream en la siguiente sección.

Actualizando nuestra visión con Turbo Streams

Como se mencionó en la sección anterior, necesitamos actualizar el total de la cotización cuando:

- Creamos, actualizamos o eliminamos una **partida**
- Destruimos una **fecha de artículo de línea** ya que puede contener algunos **artículos de línea**

Actualizar partes independientes de la página es fácil con Turbo Streams. Simplemente, volvamos a renderizar la `quotes/_total.html.erb` parte cada vez que realicemos una de esas operaciones:

```
<%# app/views/line_items/create.turbo_stream.erb %>

<%# All the previous code %>

<%= turbo_stream.update dom_id(@quote, :total) do %>
  <%= render "quotes/total", quote: @quote %>
<% end %>
```

```
<%# app/views/line_items/update.turbo_stream.erb %>

<%# All the previous code %>

<%= turbo_stream.update dom_id(@quote, :total) do %>
  <%= render "quotes/total", quote: @quote %>
<% end %>
```

```
<%# app/views/line_items/destroy.turbo_stream.erb %>

<%# All the previous code %>

<%= turbo_stream.update dom_id(@quote, :total) do %>
  <%= render "quotes/total", quote: @quote %>
<% end %>
```

```
<%# app/views/line_item_dates/destroy.turbo_stream.erb %>

<%# All the previous code %>

<%= turbo_stream.update dom_id(@quote, :total) do %>
  <%= render "quotes/total", quote: @quote %>
<% end %>
```

No olvidemos agregar una etiqueta Turbo Frame o una `div` para envolver la barra de navegación fija que contiene el total de la cotización:

```
<%# app/views/quotes/show.html.erb %>

<main class="container mb-xxxxl">
  <!-- All the previous code -->
</main>

<%= turbo_frame_tag dom_id(@quote, :total) do %>
  <%= render "quotes/total", quote: @quote %>
<% end %>
```

Ahora podemos probar en el navegador y todo debería funcionar como se espera. Dividir nuestra página en partes independientes es muy fácil, ¡gracias a Turbo!

Añadiendo pruebas del sistema a nuestra aplicación

No olvidemos agregar algunas líneas más a nuestras pruebas del sistema para asegurarnos de que nuestra aplicación se comporte como se espera. Queremos asegurarnos de que el monto total de la cotización se actualice correctamente justo después de crear, actualizar o eliminar una **partida** :

```
# test/system/line_items_test.rb

require "application_system_test_case"

class LineItemsTest < ApplicationSystemTestCase
  # All the previous code

  test "Creating a new line item" do
```

```
    # All the previous code
    assert_text number_to_currency(@quote.total_price)
  end

  test "Updating a line item" do
    # All the previous code
    assert_text number_to_currency(@quote.total_price)
  end

  test "Destroying a line item" do
    # All the previous code
    assert_text number_to_currency(@quote.total_price)
  end
end
```

También queremos asegurarnos de que el total de la cotización se actualice cuando destruimos una **fecha de artículo de línea** que contiene **artículos de línea** :

```
# test/system/line_item_dates_test.rb

require "application_system_test_case"

class LineItemDatesTest < ApplicationSystemTestCase
  # We must include this module to be able to use the
  # `number_to_currency` method in our test
  include ActionView::Helpers::NumberHelper

  # All the previous code

  test "Destroying a line item date" do
    # All the previous code
    assert_text number_to_currency(@quote.total_price)
  end
end
```

Ejecutemos todas nuestras pruebas una vez más con el `bin/rails test:all` comando. ¡Todas deberían estar en verde!

Conclusión

¡Este fue el último capítulo de este tutorial de Turbo Rails inspirado en un proyecto del mundo real en el que tuve que trabajar! ¡Espero que hayas disfrutado leyéndolo tanto como yo disfruté escribiéndolo! ¡También espero que hayas aprendido mucho sobre Turbo!

Si desea implementar su aplicación en Heroku, aquí hay una demostración de YouTube de DHH con una marca de tiempo donde habla sobre la implementación en Heroku .

Si quieres aprender más sobre Hotwire y Turbo, te recomiendo que eches un vistazo al repositorio hotwire-example-template de Sean Doyle en Thoughtbot. Todos los ejemplos del repositorio son ejemplos prácticos que encontrarás en casi todas las aplicaciones Ruby on Rails.

Si disfrutaste de este tutorial y te resultó muy útil, puedes patrocinarlo directamente desde mi perfil de Github . ¡Usaré tu dinero para mantener este sitio web y, tal vez, crear más tutoriales en el futuro!

¡Sientete libre de compartir este tutorial con todos aquellos que puedan estar interesados!

← anterior