← Volver a la lista de capítulos

Dos formas de gestionar estados vacíos con Hotwire

Publicado el 1 de marzo de 2022

En este capítulo, aprenderemos dos formas de manejar estados vacíos con Turbo. La primera utiliza Turbo Frames y Turbo Streams, y la segunda utiliza la pseudoclase CSS de hijo único.

Añadiendo estados vacíos a nuestras aplicaciones Ruby on Rails

Los estados vacíos son una parte importante de nuestras aplicaciones . Cuando llegamos a una página web por primera vez, no tendremos ningún dato que nos ayude a adivinar para qué se utiliza la página. Como nuevo usuario de una aplicación, es bueno tener una imagen o algunas frases que expliquen las acciones que podemos realizar en la página.

Si destruimos todas las citas en nuestro editor de citas, actualmente tenemos una página en blanco con solo un título y un botón. Sería bueno mostrar un estado vacío cuando no haya citas para ayudar a nuestros usuarios.

En este breve capítulo, aprenderemos dos formas de agregar estados vacíos a nuestras aplicaciones Ruby on Rails 7:

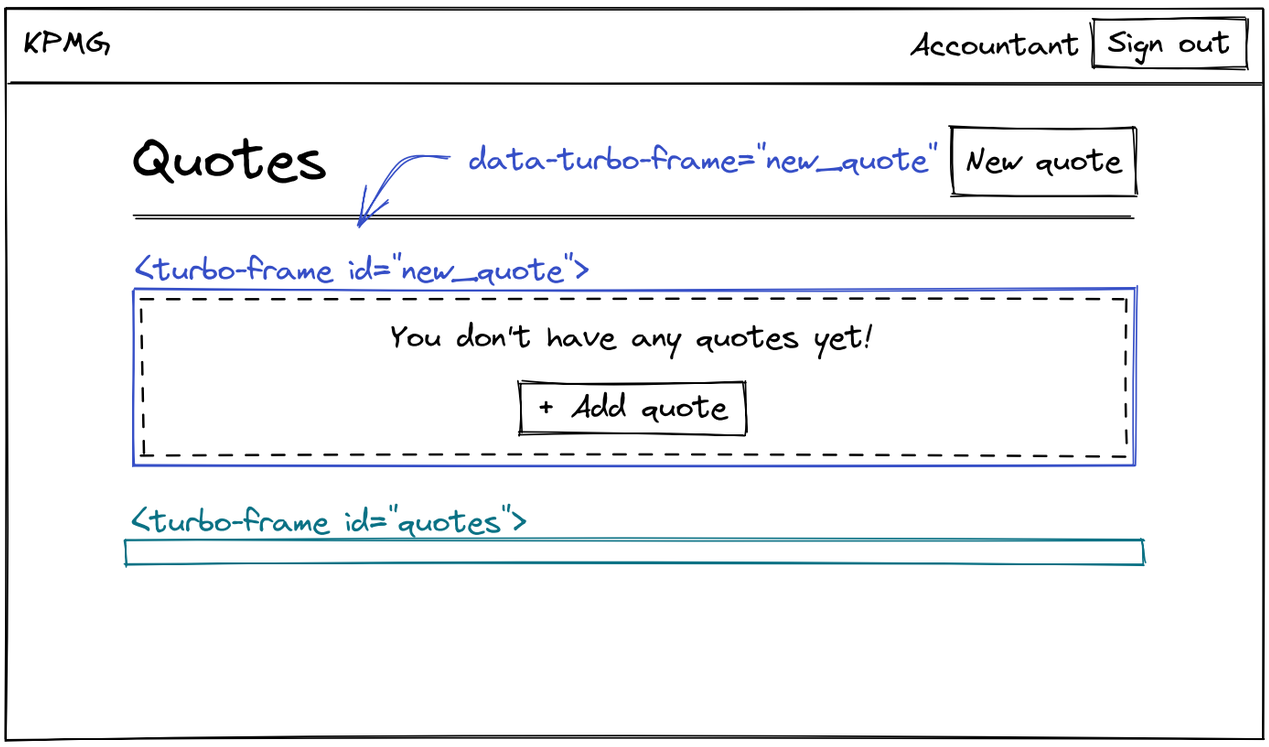
1. El primero utilizará Turbo Frames y Turbo Streams

2. El segundo utilizará la `:only-child` pseudoclase CSS

¡Empecemos!

Estados vacíos con Turbo Frames y Turbo Streams

Antes de sumergirnos en el código, tomémonos un tiempo para esbozar lo que crearemos. Cuando un usuario no tiene ninguna cita, queremos mostrar un estado vacío que contenga un "mensaje útil" y una llamada a la acción:



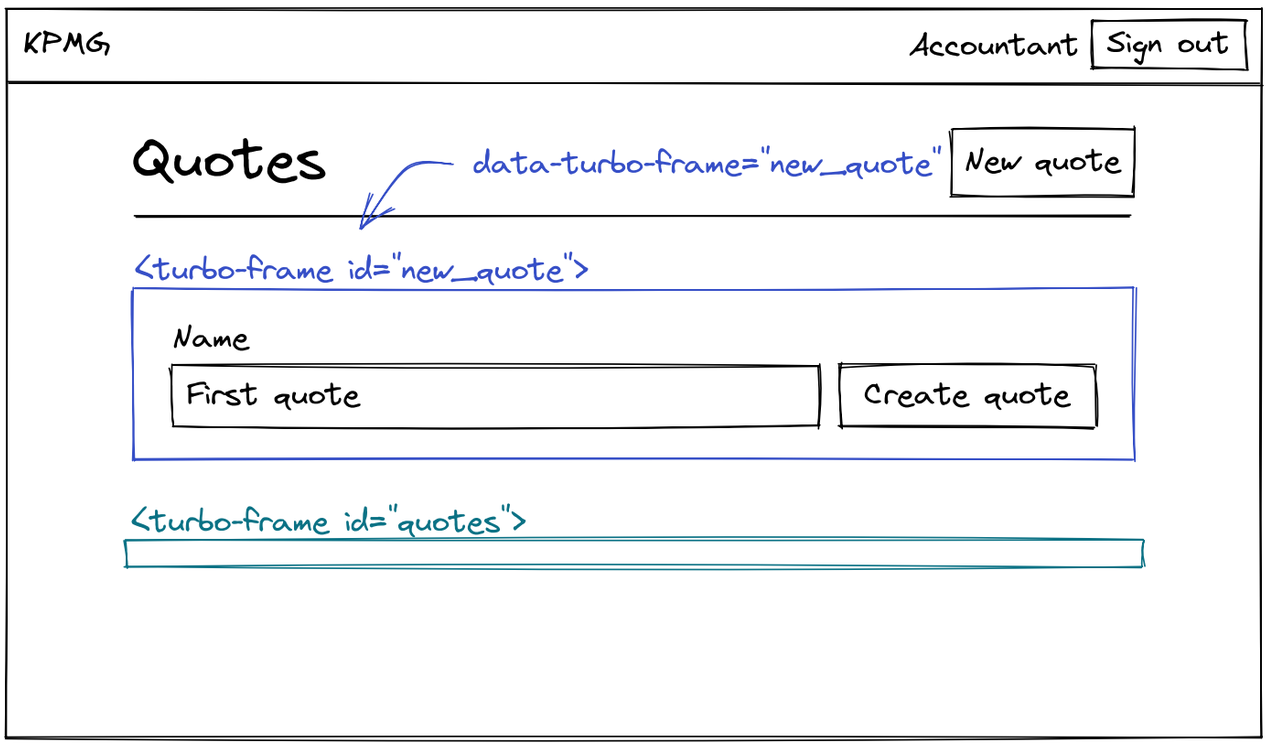
Boceto de la página de índice de citas con el estado vacío

Agregaremos el estado vacío dentro del Turbo Frame con id `new_quote` , de modo que:

- Si el usuario hace clic en el botón “Nueva cotización” dentro del encabezado, Turbo reemplazará el Turbo Frame con id `new_quote` en la `Quotes#index` página por el extraído de la `Quotes#new` página, gracias al `data-turbo-frame="new_quote"` atributo data.
- Si el usuario hace clic en el botón "Agregar cotización" dentro del estado vacío, Turbo reemplazará el Turbo Frame con id `new_quote` de la `Quotes#index` página con el extraído de la `Quotes#new` página, ya que el enlace está dentro de este Turbo Frame.

Como se ha descrito anteriormente, cuando el usuario hace clic en cualquiera de los dos enlaces, el estado vacío se sustituirá por el nuevo formulario de

cotización. El estado de la página cuando el usuario hace clic en el botón "Nueva cotización" o "Añadir cotización" se describe en el siguiente esquema:

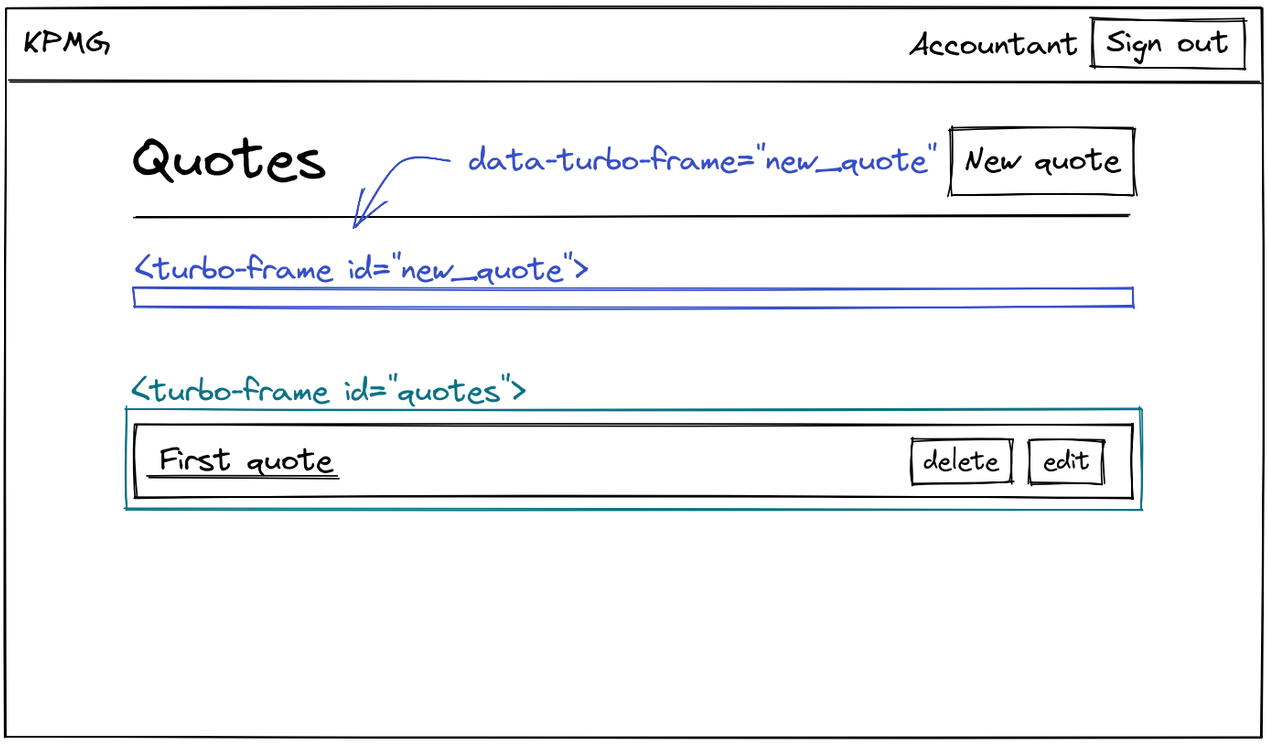


Boceto de la página Quotes#index con el estado vacío reemplazado

Cuando el usuario envía un formulario válido, el comportamiento no cambia:

1. La cita creada se *antepone* a la lista de citas.
2. `new_quote` Se elimina el HTML contenido en el Turbo Frame con id

Este comportamiento se describe en el siguiente esquema:



Boceto de la página Quotes#index sin el estado vacío

Si actualizamos la página, **el estado vacío ya no debería ser visible ya que hay al menos una cita en la página** .

Ahora que los requisitos están claros, ¡comencemos a codificar! Lo primero que tenemos que hacer es mostrar el estado vacío solo cuando no haya comillas en la página. Para ello, crearemos un **estado vacío parcial** que luego podremos usar en la `Quotes#index` página:

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

<div class="empty-state">
  <p class="empty-state__text">
    You don't have any quotes yet!
  </p>

  <%= link_to "Add quote", new_quote_path, class: "btn btn--primary" %>
</div>
```

Ahora podemos representar este estado vacío solo cuando el usuario actual no tiene citas en la `Quotes#index` página:

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

<%= turbo_stream_from current_company, "quotes" %>

<div class="container">
  <div class="header">
    <h1>Quotes</h1>
    <%= link_to "New quote",
                new_quote_path,
                class: "btn btn--primary",
                data: { turbo_frame: dom_id(Quote.new) } %>
  </div>

  <%= turbo_frame_tag Quote.new do %>
    <% if @quotes.none? %>
      <%= render "quotes/empty_state" %>
    <% end %>
  <% end %>

  <%= turbo_frame_tag "quotes" do %>
    <%= render @quotes %>
```

```
  <% end %>
</div>
```

Antes de probar en el navegador, también diseñemos nuestro estado vacío para que quede un poco más agradable:

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

.empty-state {
  padding: var(--space-m);
  border: var(--border);
  border-style: dashed;
  text-align: center;

  &__text {
    font-size: var(--font-size-l);
    color: var(--color-text-header);
    margin-bottom: var(--space-l);
    font-weight: bold;
  }
}
```

No olvidemos importar este archivo a nuestro archivo de manifiesto para aplicar los estilos:

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

// All the previous code
@import "components/empty_state";
```

Ahora estamos listos para experimentar en el navegador.

En la `Quotes#index` página, eliminemos todas las citas. Luego podemos hacer clic en el botón "Nueva cita" o "Agregar cita". Deberíamos ver que el formulario para crear una nueva cita reemplaza el estado vacío. Si enviamos un formulario válido, la cita creada se *antepone* a la lista de citas y el estado vacío ya no es visible.

Sin embargo, hay una pequeña mejora que podríamos hacer. Si eliminamos la cita que acabamos de crear, el estado vacío no vuelve a aparecer en la pantalla. **Queremos que el estado vacío esté siempre presente cuando no haya ninguna cita en la página** . Para ello, debemos indicar a la

`destroy.turbo_stream.erb` vista que actualice el contenido del Turbo Frame con id `new_quote` con el contenido del `quotes/empty_state` parcial cuando no haya citas en la página:

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

<%= turbo_stream.remove @quote %>
<%= render_turbo_stream_flash_messages %>

<% unless current_company.quotes.exists? %>
  <%= turbo_stream.update Quote.new do %>
    <%= render "quotes/empty_state" %>
  <% end %>
<% end %>
```

Con esas líneas de código agregadas, ¡nuestro estado vacío ahora se comporta como queremos! ¡Perfecto!

Sin embargo, hay un pequeño problema con la implementación actual que podríamos pasar por alto fácilmente . Desde el Capítulo 5 y el Capítulo 6 , estamos transmitiendo las creaciones, actualizaciones y eliminaciones de cotizaciones realizadas por todos los usuarios de nuestra empresa en la `Quotes#index` página. Por lo tanto, **si alguien crea una cotización mientras estamos en la página vacía, la cotización se agregará a la lista y el estado vacío seguirá siendo visible en la página** .

Analicemos por qué este caso de uso es importante y cómo podemos resolverlo en la siguiente sección.

Estados vacíos con la pseudoclase CSS de hijo único

Antes de hablar sobre la segunda forma de manejar estados vacíos con Turbo, intentemos reproducir el problema en el navegador. Naveguemos hasta la `Quotes#index` página y eliminemos todas las comillas. Luego, creemos una nueva comilla **desde la consola** . La comilla creada se transmite a nuestra vista y vemos que **tanto la comilla creada como el estado vacío son visibles en la página** .

Nota : Nuestro ejemplo con *comillas* parece un poco complicado. Sin embargo, imaginemos por un segundo que esas *comillas* fueran *notificaciones* , como por ejemplo, la notificación de Github:

1. Cuando no tenemos notificaciones en la página, queremos ver el estado vacío
2. Cuando se envía una notificación a nuestra vista, queremos que el estado vacío desaparezca
3. Cuando eliminamos la notificación, queremos que el estado vacío vuelva a aparecer en la pantalla.

Ese es el comportamiento que implementaremos en esta sección, y las notificaciones son un gran caso de uso.

Analicemos el problema aquí. Como se explicó en el Capítulo 5 y el Capítulo 6 , gracias al `broadcasts_to` método del `Quote` modelo:

- Cuando se crea una cita, el contenido del `quotes/_quote.html.erb` parcial se antepone a la lista de citas.
- Cuando se elimina una cita, la cita se elimina de la lista.

De forma predeterminada, no se mencionan los estados vacíos. Si bien podríamos *jugar* con las devoluciones de llamadas y anular las opciones predeterminadas del `broadcasts_to` método, existe una forma elegante de lograr lo que queremos, gracias a la `:only-child` pseudoclase en CSS. El comportamiento que queremos lograr es el siguiente:

- Cuando el estado vacío es el **único hijo de la lista de citas** , queremos que sea visible
- Cuando el estado vacío **no es el único hijo** de la lista de citas, queremos que sea **invisible**

El comportamiento que queremos es ligeramente diferente del primer método, ya que esta vez no reemplazaremos el estado vacío con el nuevo formulario de cotización .

¡Comencemos a codificar! Primero, debemos mover el contenido del `quotes/empty_state` parcial a la lista de comillas:

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

<%= turbo_stream_from current_company, "quotes" %>

<div class="container">
  <div class="header">
    <h1>Quotes</h1>
    <%= link_to "New quote",
                new_quote_path,
                class: "btn btn--primary",
                data: { turbo_frame: dom_id(Quote.new) } %>
  </div>

  <%= turbo_frame_tag Quote.new %>

  <%= turbo_frame_tag "quotes" do %>
    <%= render "quotes/empty_state" %>
    <%= render @quotes %>
  <% end %>
</div>
```

Luego, tenemos que usar la `:only-child` pseudoclase en nuestro CSS para mostrar el estado vacío cuando es el único hijo del Turbo Frame con id `quotes` y ocultarlo cuando no lo es:

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

.empty-state {
  padding: var(--space-m);
  border: var(--border);
  border-style: dashed;
  text-align: center;

  &__text {
    font-size: var(--font-size-l);
    color: var(--color-text-header);
    margin-bottom: var(--space-l);
    font-weight: bold;
  }

  &--only-child {
    display: none;

    &:only-child {
```

```
      display: revert;
    }
  }
}
```

Utilizamos lo que la metodología BEM llama un *modificador* aquí para nuestra `.empty-state--only-child` clase CSS porque queremos soportar los dos métodos presentados en este capítulo con la misma `.empty-state` clase.

En nuestro estado parcial vacío, necesitamos que el enlace "Agregar cita" **se dirija explícitamente** al Turbo Frame con el id de, `new_quote` ya que ya no es un elemento secundario del Turbo Frame. Podemos lograr esto gracias al `data-turbo-frame="new_quote"` atributo de datos:

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

<div class="empty-state empty-state--only-child">
  <p class="empty-state__text">
    You don't have any quotes yet!
  </p>

  <%= link_to "Add quote",
              new_quote_path,
              class: "btn btn--primary",
              data: { turbo_frame: dom_id(Quote.new) } %>
</div>
```

También podemos restablecer el contenido de la `destroy.turbo_stream.erb` vista ya que ya no necesitamos ningún

💎 Trenes calientes

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

<%= turbo_stream.remove @quote %>
<%= render_turbo_stream_flash_messages %>
```

Ahora podemos probar el comportamiento en nuestro navegador:

- Cuando tenemos comillas en la lista, el estado vacío no es visible
- Cuando no tenemos comillas en la lista, el estado vacío es visible

¡La mejor parte es que pudimos lograr este comportamiento solo con CSS!

Envolver

En este capítulo, vimos dos métodos para administrar estados vacíos con Turbo Frames y Turbo Streams.

En el primer método, usamos Turbo Frames y Turbo Streams para agregar o eliminar con precisión el estado vacío en la `Quotes#index` página cuando fuera necesario. Si bien este método es excelente en la mayoría de los casos, puede que no sea el más adecuado cuando se transmite HTML a la vista.

En el segundo método, aprovechamos el poder de la `:only-child` pseudoclase CSS para que hiciera todo el trabajo por nosotros. ¡No tuvimos que escribir ningún código personalizado relacionado con Turbo/Turbo Rails!

En los siguientes tres capítulos trabajaremos en la `Quotes#show` página para finalizar nuestro editor de citas. ¡Nos vemos allí!

← anterior Siguiente →