← Volver a la lista de capítulos

Plantillas Turbo Frames y Turbo Stream

Publicado el 28 de enero de 2022

En este capítulo aprenderemos a dividir nuestra página en partes independientes gracias a Turbo Frames y al formato Turbo Stream. Después de leer este capítulo, todas las acciones CRUD sobre citas se realizarán en la página de índice de citas.

¡Patrocina este proyecto en Github!

Este tutorial es de código abierto para siempre. Si quieres apoyar mi trabajo, ¡puedes patrocinarlo en Github! **Te invitaré a un repositorio con el código fuente del tutorial** .

Conviértete en patrocinador

Lo que construiremos en este capítulo

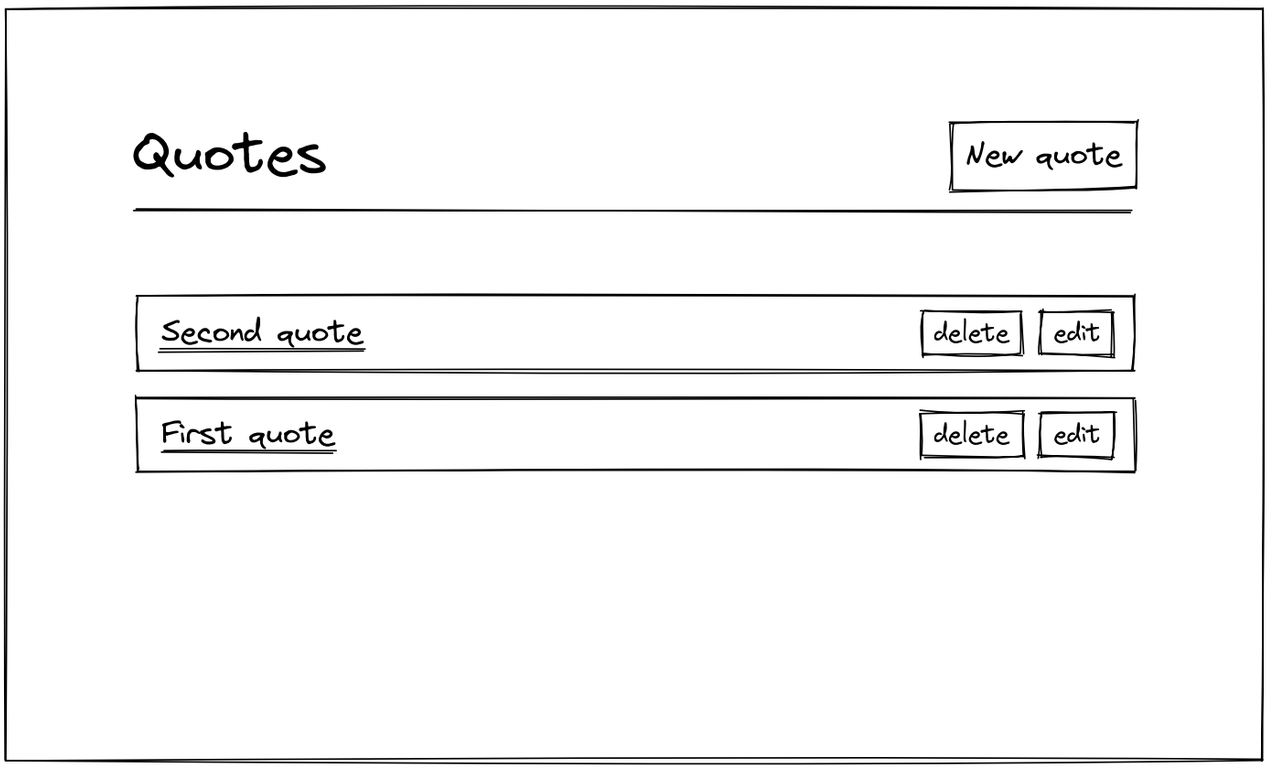
Las acciones `#new` y `#edit` ocurren en distintas páginas de nuestro editor de cotizaciones actual. Cuando estamos en la `Quotes#index` página:

- Al hacer clic en el botón "Nueva cotización" se abre una página completamente diferente que contiene solo un título y un formulario para crear una nueva cotización.
- Al hacer clic en el botón "Editar" de una cita, se abre otra página que contiene solo un título y un formulario para editar la cita.

Nos gustaría evitar este *cambio de contexto* . En su lugar, nos gustaría realizar esas dos acciones directamente en la `Quotes#index` página, como en el editor de citas final . Eso es lo que vamos a aprender a hacer en este capítulo. Requerirá solo unas pocas líneas de código gracias al increíble poder de Turbo Frames y Turbo Streams.

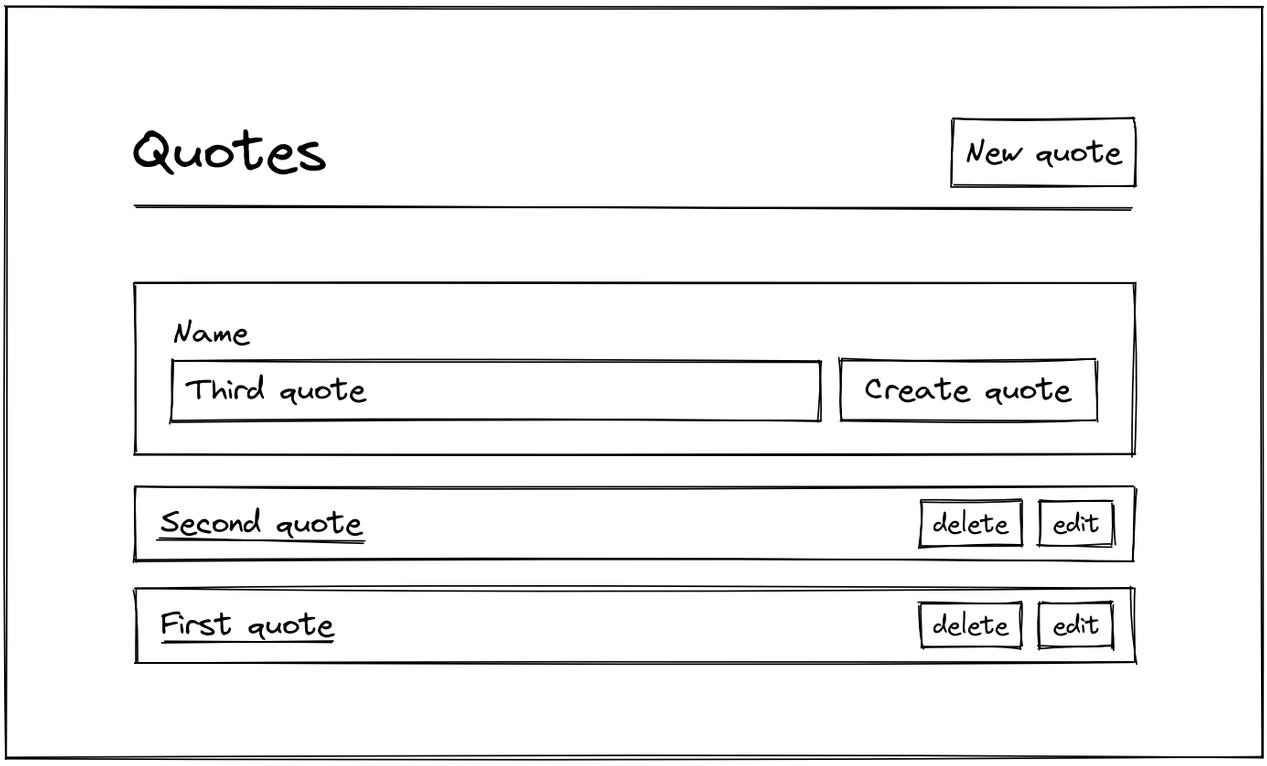
Antes de practicar nuestras habilidades en Turbo Frame, hagamos algunos bocetos de lo que construiremos y actualicemos nuestras pruebas del sistema.

Nuestra `Quotes#index` página actualmente se ve así:



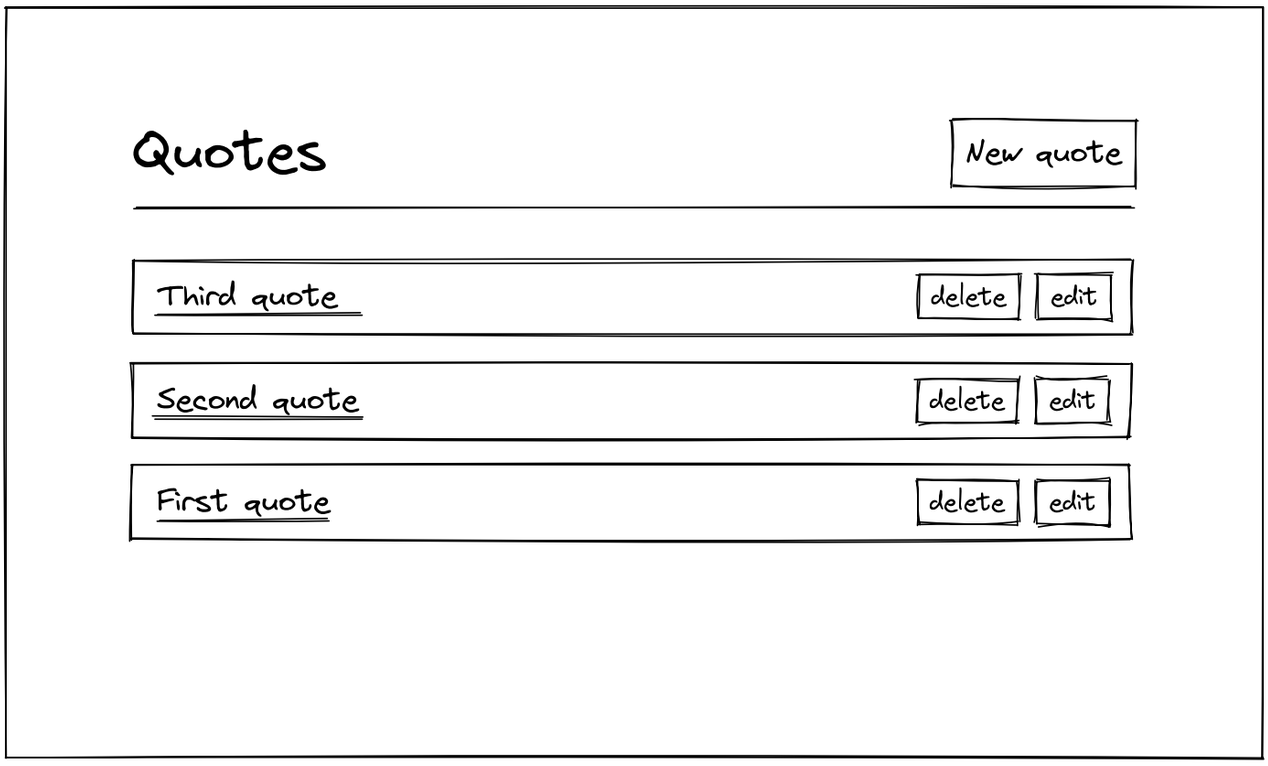
Boceto de la página de citas#índice

Al hacer clic en el botón "Nueva cotización", queremos que el nuevo formulario de cotización se adjunte a la página justo debajo del encabezado:



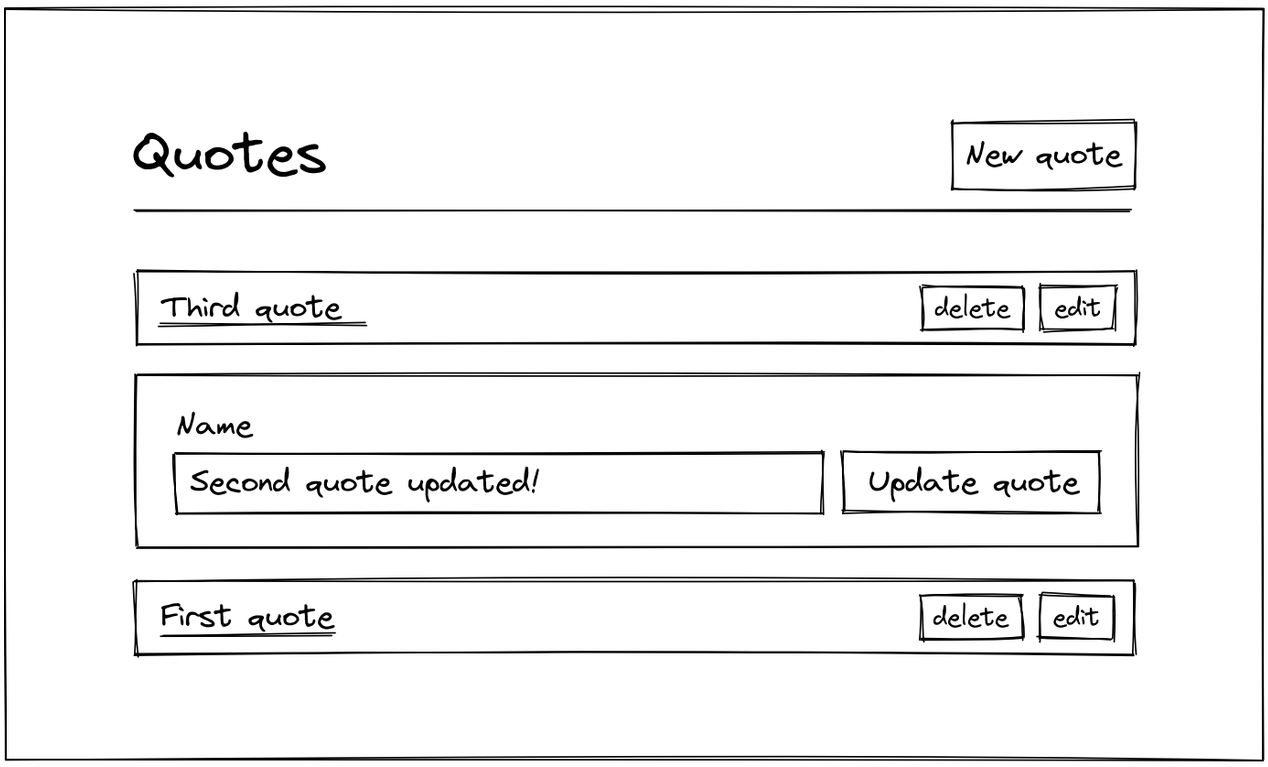
Boceto de la página de índice de cotizaciones con el nuevo formato de cotización

Al hacer clic en el botón "Crear cotización", la cotización creada debe agregarse al comienzo de la lista de cotizaciones:



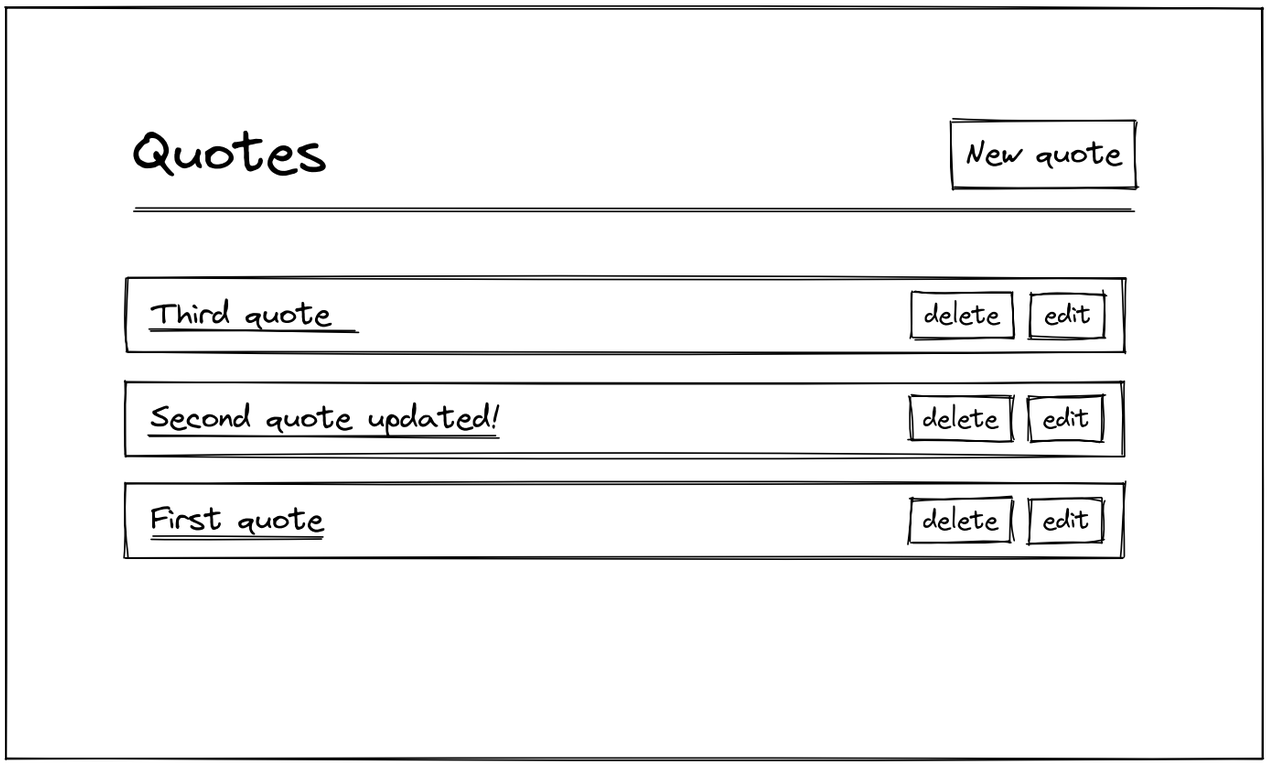
Boceto de la página Quotes#index con la cita creada antepuesta a la lista

Al hacer clic en el botón "Editar" **de la segunda cotización** , la tarjeta de cotización debe reemplazarse por un formulario para editar la cotización correspondiente:



Boceto de la página de índice de citas con un formulario para editar la segunda cita

Al hacer clic en el botón "Actualizar cotización", el formulario de la segunda cotización debe reemplazarse por la cotización actualizada:



Boceto de la página de índice de citas con la cita actualizada

El resto del comportamiento debería permanecer sin cambios:

- Al hacer clic en el botón "Eliminar" de una cita, debería eliminarla de la lista de citas.
- Al hacer clic en el nombre de una cita, deberíamos acceder a la `Quotes#show` página.

Ahora actualizaremos nuestras pruebas del sistema Capybara para que coincidan con el comportamiento deseado:

```
# test/system/quotes_test.rb

require "application_system_test_case"

class QuotesTest < ApplicationSystemTestCase
  setup do
    @quote = quotes(:first)
  end

  test "Showing a quote" do
    visit quotes_path
    click_link @quote.name

    assert_selector "h1", text: @quote.name
  end
```

```
  test "Creating a new quote" do
    visit quotes_path
    assert_selector "h1", text: "Quotes"

    click_on "New quote"
    fill_in "Name", with: "Capybara quote"

    assert_selector "h1", text: "Quotes"
    click_on "Create quote"

    assert_selector "h1", text: "Quotes"
    assert_text "Capybara quote"
  end

  test "Updating a quote" do
    visit quotes_path
    assert_selector "h1", text: "Quotes"

    click_on "Edit", match: :first
    fill_in "Name", with: "Updated quote"

    assert_selector "h1", text: "Quotes"
    click_on "Update quote"

    assert_selector "h1", text: "Quotes"
    assert_text "Updated quote"
  end

  test "Destroying a quote" do
    visit quotes_path
    assert_text @quote.name

    click_on "Delete", match: :first
    assert_no_text @quote.name
  end
end
```

Si ejecutamos las pruebas ahora, las dos pruebas correspondientes a la creación y edición de cotizaciones fallarán. Nuestro objetivo es hacer que vuelvan a estar en verde con Turbo Frames y Turbo Streams. ¿Listo para aprender cómo funcionan? ¡Vamos a profundizar!

¿Qué son los Turbo Frames?

Los Turbo Frames son piezas independientes de una página web que se pueden agregar, anteponer, reemplazar o eliminar sin tener que actualizar completamente la página y sin escribir una sola línea de JavaScript.

En esta sección aprenderemos todo lo que hay que saber sobre Turbo Frames con una serie de pequeños ejemplos. Luego volveremos a nuestras citas e implementaremos el comportamiento deseado con solo unas pocas líneas de código.

Creemos nuestro primer Turbo Frame. Para crear Turbo Frames, usamos el `turbo_frame_tag` asistente. Envolvamos el encabezado de la `Quotes#index` página en un Turbo Frame con un id de "first_turbo_frame":

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

<main class="container">
  <%= turbo_frame_tag "first_turbo_frame" do %>
    <div class="header">
      <h1>Quotes</h1>
      <%= link_to "New quote", new_quote_path, class: "btn btn--primary"
    </div>
  <% end %>

  <%= render @quotes %>
</main>
```

Si echamos un vistazo al DOM, el HTML generado para Turbo Frame se ve así:

```
<turbo-frame id="first_turbo_frame">
  <div class="header">
    <h1>Quotes</h1>
    <a class="btn btn--primary" href="/quotes/new">New quote</a>
  </div>
</turbo-frame>
```

Como podemos ver, el `turbo_frame_tag` asistente crea un `<turbo-frame>` elemento personalizado que contiene el HTML generado por el contenido del bloque. Este elemento personalizado tiene un identificador único que corresponde al primer argumento que pasamos al `turbo_frame_tag` asistente.

Esta `<turbo-frame>` etiqueta HTML no existe en el lenguaje HTML. Es un elemento personalizado que se creó en la biblioteca Turbo JavaScript . Intercepta los envíos de formularios y los clics en los enlaces dentro del marco, lo que convierte a esos marcos en partes independientes de su página web.

Ahora hagamos clic en el botón "Nueva cotización" y... El marco desaparece de la página y aparece un error en la consola: *La respuesta no tiene `<turbo-frame id="first_turbo_frame">` el elemento* correspondiente . Vamos a explicar este comportamiento extraño.

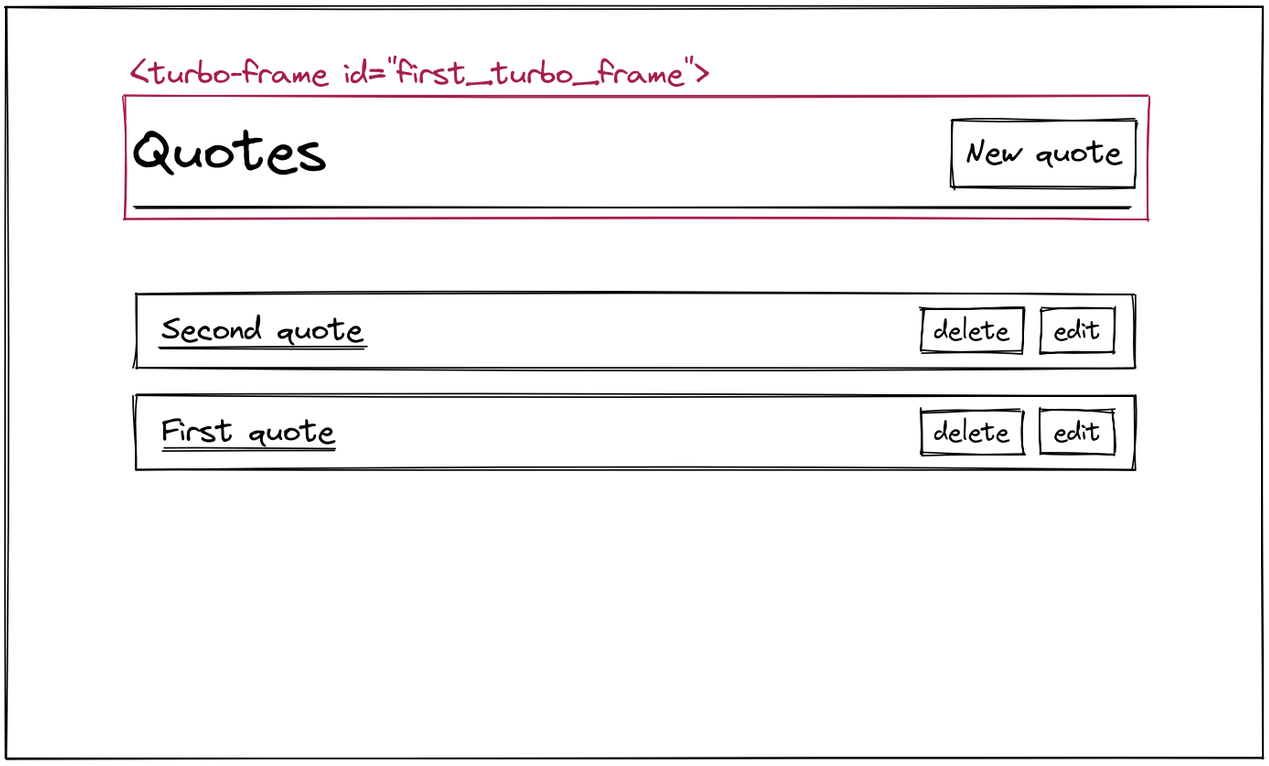
Hoja de trucos de Turbo Frames

En esta sección explicaremos las reglas que se aplican a los Turbo Frames.

Incluso si los ejemplos están escritos con enlaces, ¡esas reglas se aplican tanto a los enlaces como a los formularios!

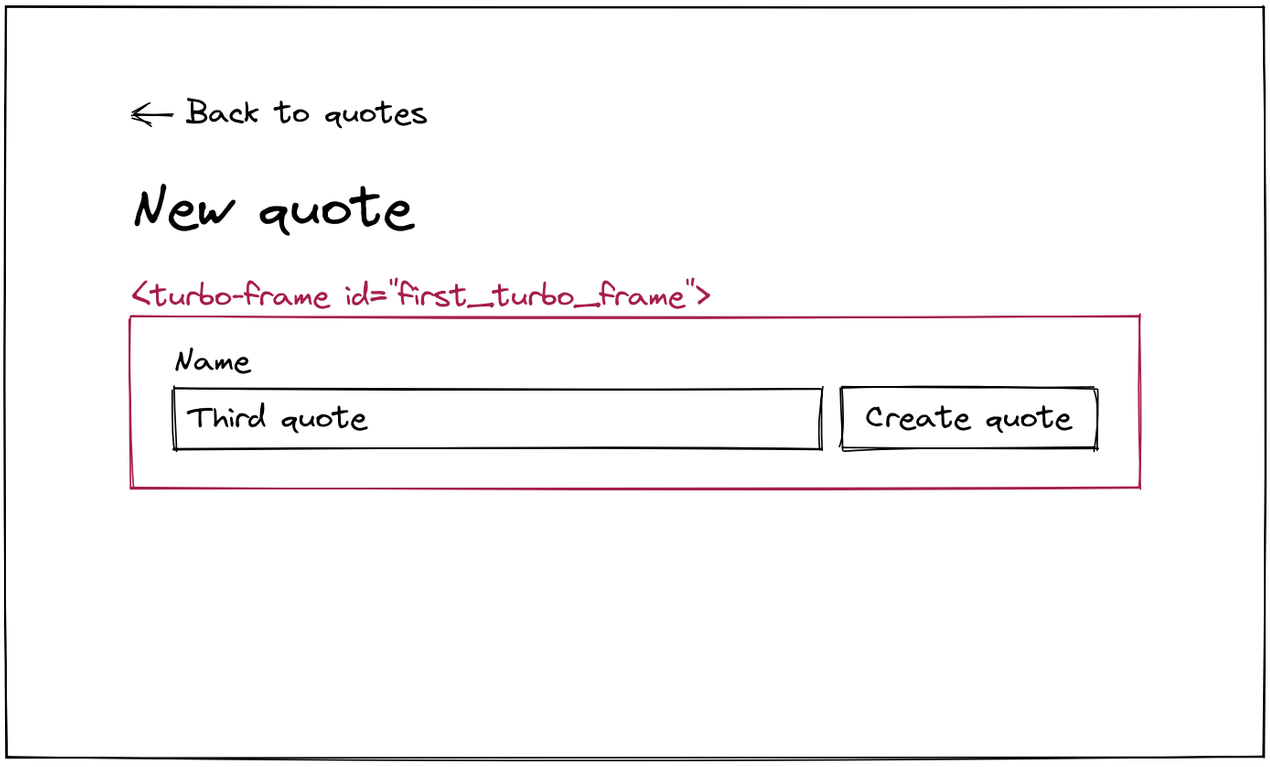
Regla 1 : al hacer clic en un enlace dentro de un Turbo Frame, Turbo espera un frame con el mismo ID en la página de destino. Luego, reemplazará el contenido del Frame en la página de origen con el contenido del Frame en la página de destino.

Si no te ha quedado muy claro ahora, vamos a hacer algunos bocetos del experimento que vamos a realizar. Nuestra `Quotes#index` página actual tiene este aspecto:



Boceto de la página de índice de citas con nuestro primer Turbo Frame

Ahora, envolvamos una parte de la `Quotes#new` página en un Turbo Frame con el mismo ID. En nuestro ejemplo, envolveremos el formulario en ese Turbo Frame de la siguiente manera:



Boceto de las Cotizaciones#nueva página con nuestro primer Turbo Frame

Para que coincidan con los bocetos que acabamos de dibujar, agreguemos el Turbo Frame con la misma identificación en la `Quotes#new` página:

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

<main class="container">
```

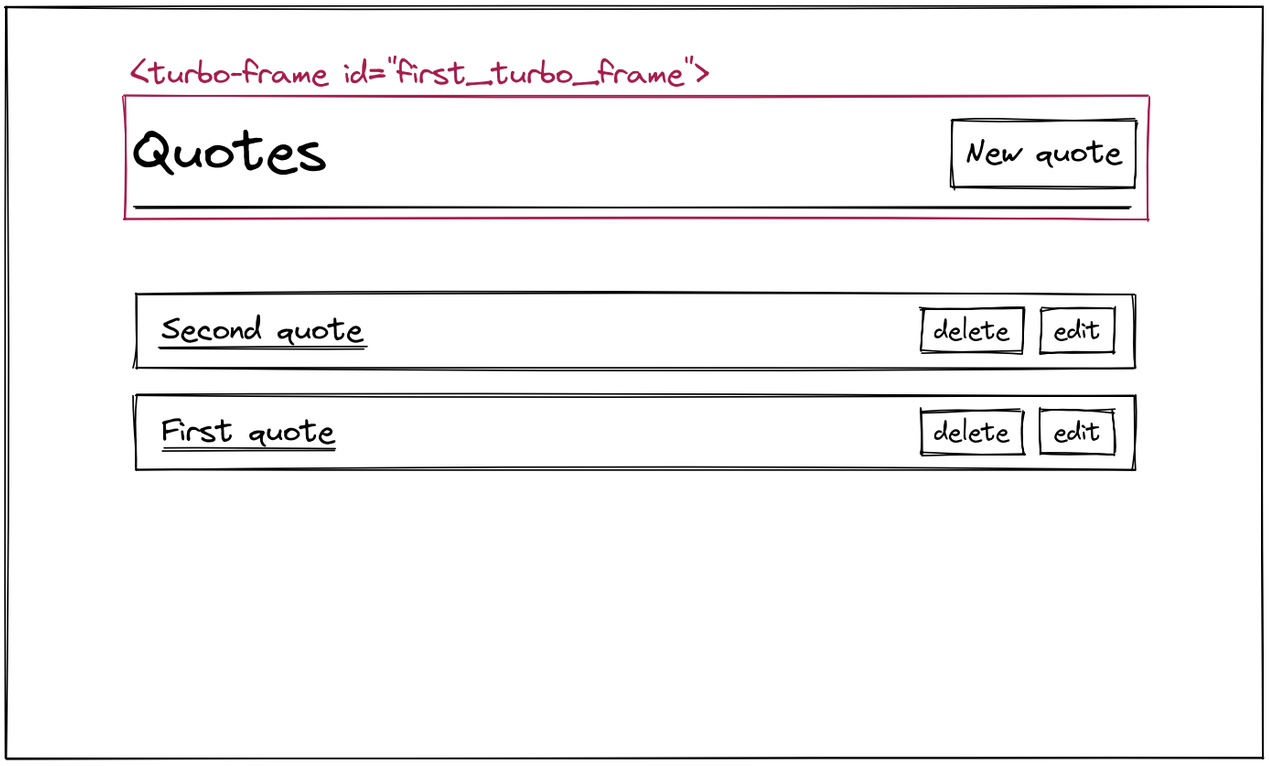
```
  <%= link_to sanitize("&larr; Back to quotes"), quotes_path %>

  <div class="header">
    <h1>New quote</h1>
  </div>

  <%= turbo_frame_tag "first_turbo_frame" do %>
    <%= render "form", quote: @quote %>
  <% end %>
</main>
```

Experimentemos ahora en el navegador. Actualicemos la `Quotes#index` página y hagamos clic en el botón "Nueva cita". Podemos ver que el contenido de nuestro Turbo Frame con id `"first_turbo_frame"` en la `Quotes#index` página se reemplaza por el contenido del Turbo Frame en la `Quotes#new` página.

Al hacer clic en un enlace dentro de un Turbo Frame, si hay un marco con la misma identificación en la página de destino, Turbo reemplazará el contenido del Turbo Frame de la página de origen con el contenido del Turbo Frame de la página de destino .

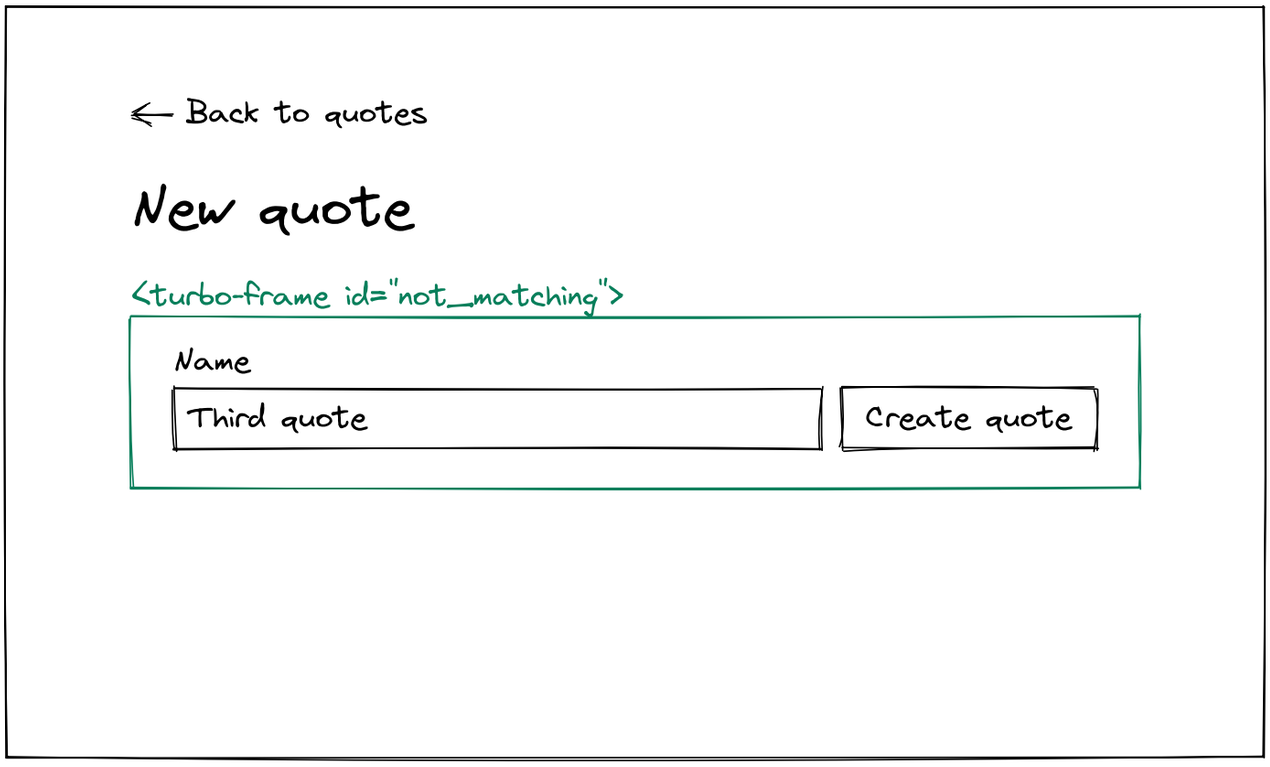
Regla 2 : Al hacer clic en un enlace dentro de un Turbo Frame, si no hay ningún Turbo Frame con el mismo ID en la página de destino, el frame desaparece y se registra el error La *respuesta no tiene ningún* `<turbo-frame id="name_of_the_frame">` *elemento coincidente en la consola.*

Recuerde el comportamiento extraño que tuvimos cuando no teníamos un Turbo Frame con el mismo ID en la `Quotes#new` página. De eso se trata exactamente esta segunda regla. Nuestra `Quotes#index` página actual se ve así:



Boceto de la página de índice de citas con nuestro primer Turbo Frame

En la `Quotes#new` página, cambiemos la identificación del Turbo Frame alrededor del formulario a "not_matching" como se describe en el siguiente boceto:



Boceto de las citas#nueva página sin Turbo Frame correspondiente

Actualicemos el marcado de la `Quotes#new` página para que coincida con nuestros bocetos:

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

<main class="container">
```

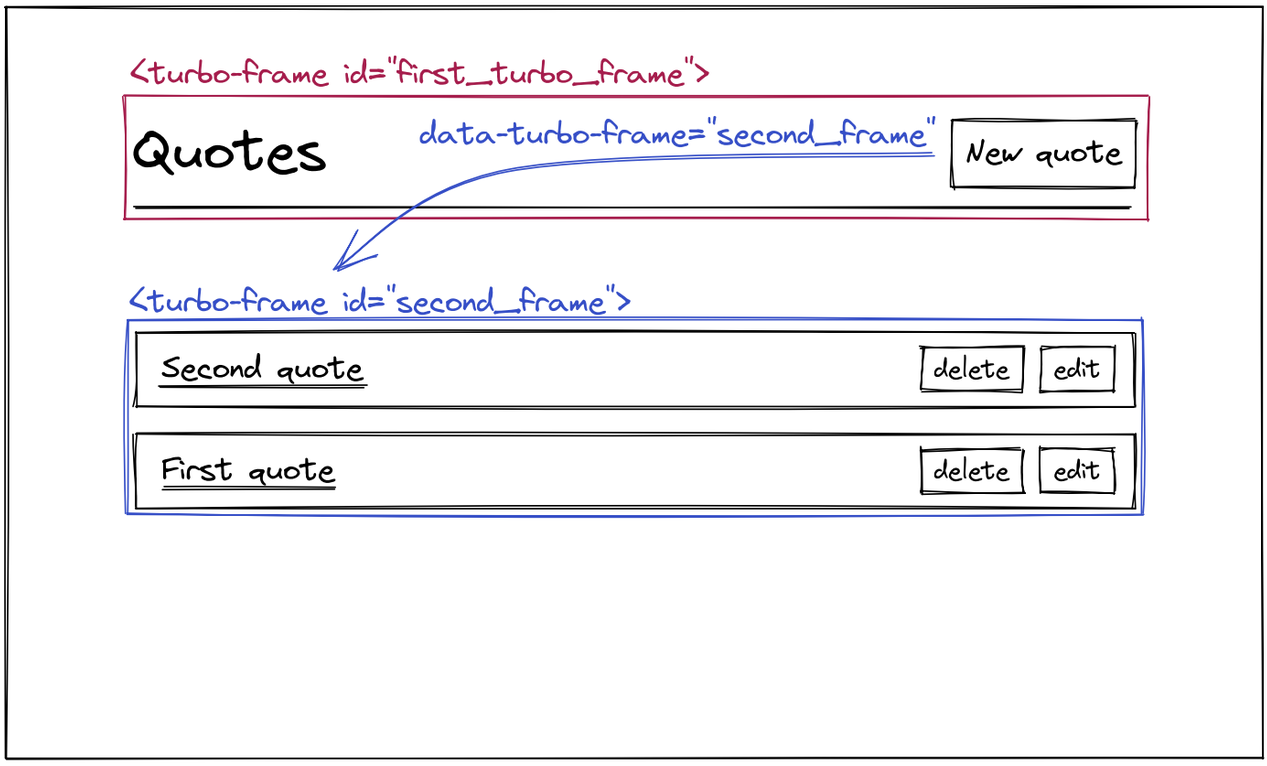
```
  <%= link_to sanitize("&larr; Back to quotes"), quotes_path %>

  <div class="header">
    <h1>New quote</h1>
  </div>

  <%= turbo_frame_tag "not_matching" do %>
    <%= render "form", quote: @quote %>
  <% end %>
</main>
```

Realicemos el experimento nuevamente, naveguemos a la `Quotes#index` página, actualicemos la página y hagamos clic en el botón "Nueva cotización". El Turbo Frame con el encabezado desaparece y el error *La respuesta no tiene ningún `<turbo-frame id="name_of_the_frame">` elemento* coincidente se registra en la consola como se esperaba.

Al hacer clic en un enlace dentro de un Turbo Frame, si no hay ningún Turbo Frame con la misma identificación en la página de destino, Turbo eliminará el contenido del Turbo Frame de la página de origen y registrará un error .

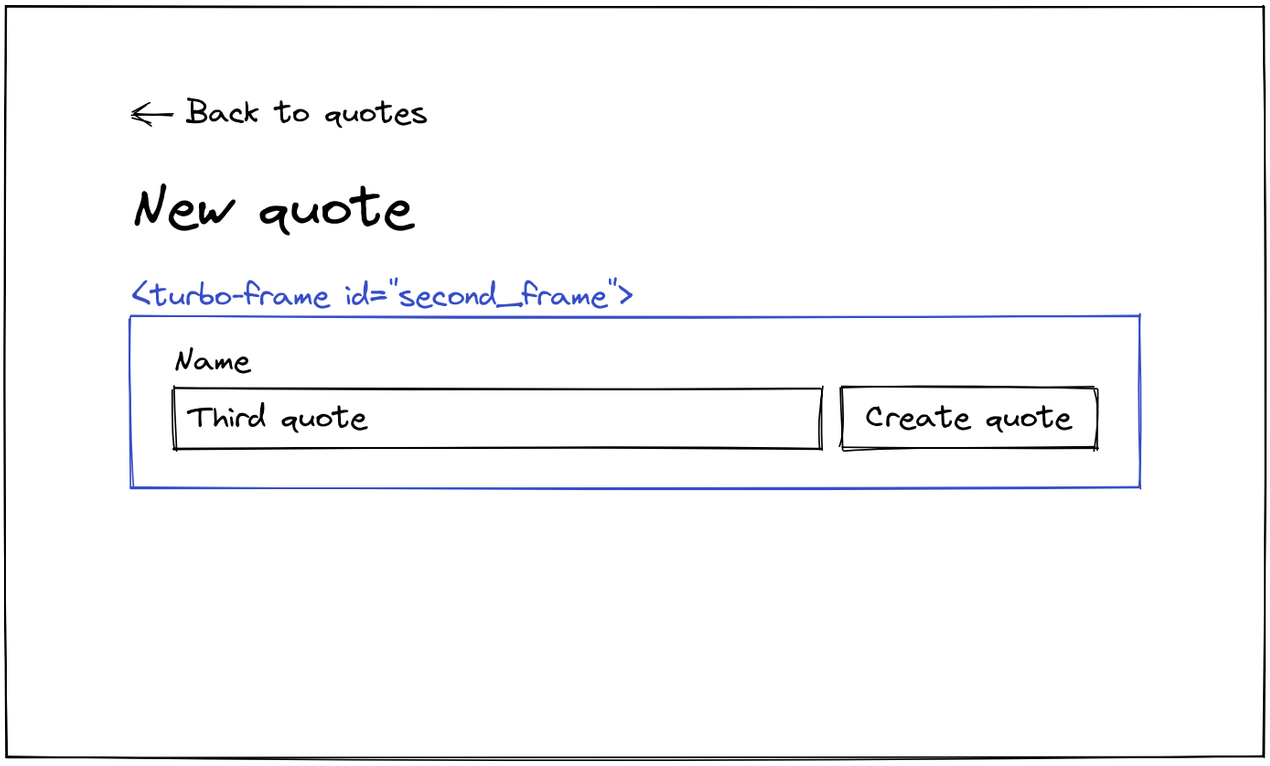
Regla 3 : Un enlace puede apuntar a otro marco que no sea aquel en el que está anidado directamente gracias al `data-turbo-frame` atributo de datos.

Esta regla es muy útil, pero necesitaremos más bocetos para entenderla con claridad. Primero, agreguemos otro Turbo Frame con el id de "second_frame" alrededor de la lista de citas en la `Quotes#index` página:



Boceto de la página de índice de citas con nuestro primer Turbo Frame y un segundo Turbo Frame

En la `Quotes#new` página, envolvamos el formulario en un marco con la misma identificación que el segundo marco:



Boceto de las Cotizaciones#nuevo con el segundo Turbo Frame

Ahora, vamos a actualizar nuestro código para que coincida con los bocetos. En la `Quotes#index` página, debemos agregar el segundo Turbo Frame y el `data-turbo-frame` atributo de datos con el mismo ID que este segundo Turbo Frame:

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

```
<main class="container">
  <%= turbo_frame_tag "first_turbo_frame" do %>
    <div class="header">
      <h1>Quotes</h1>
      <%= link_to "New quote",
                  new_quote_path,
                  data: { turbo_frame: "second_frame" },
                  class: "btn btn--primary" %>
    </div>
  <% end %>

  <%= turbo_frame_tag "second_frame" do %>
    <%= render @quotes %>
  <% end %>
</main>
```

En la `Quote#new` página, envolvamos nuestro formulario en un Turbo Frame con el mismo nombre que el segundo frame:

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

<main class="container">
  <%= link_to sanitize("&larr; Back to quotes"), quotes_path %>

  <div class="header">
    <h1>New quote</h1>
  </div>

  <%= turbo_frame_tag "second_frame" do %>
    <%= render "form", quote: @quote %>
  <% end %>
</main>
```

Ahora, experimentemos nuevamente. Visitemos la `Quotes#index` página, actualicemos la página y hagamos clic en el botón "Nueva cita". Deberíamos ver que nuestra lista de citas se reemplaza por el nuevo formulario de citas. Esto se debe a que nuestro enlace ahora apunta al segundo marco gracias al `data-turbo-frame` atributo.

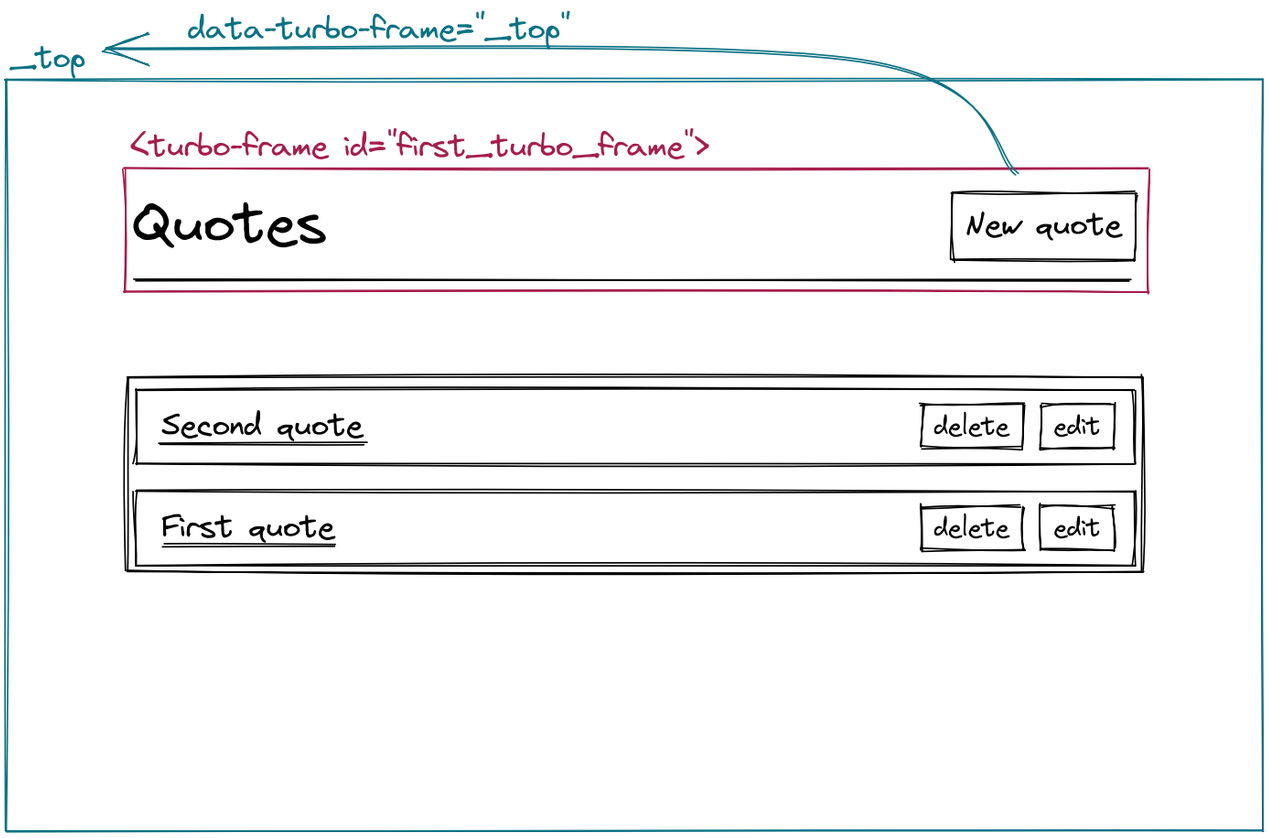
Un enlace puede dirigirse a un Turbo Frame en el que no está anidado directamente, gracias al `data-turbo-frame` atributo de datos. En ese caso, el Turbo Frame con el mismo ID que el `data-turbo-frame` atributo de datos en

la página de origen será reemplazado por el Turbo Frame con el mismo ID que el `data-turbo-frame` atributo de datos en la página de destino .

Nota :

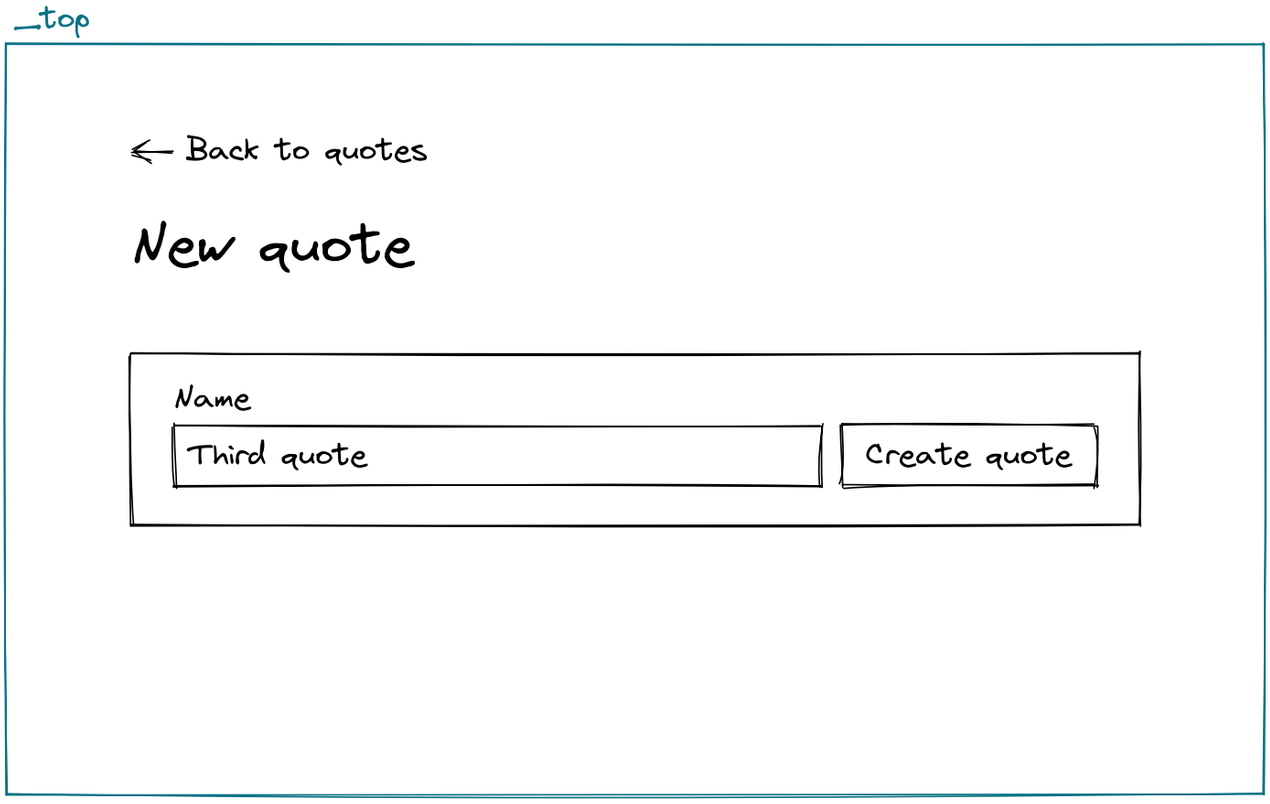
Existe un *marco* especial llamado `_top` que representa toda la página. **En realidad no es un Turbo Frame, pero se comporta casi como tal, por lo que haremos esta aproximación para nuestro modelo mental .**

Por ejemplo, si quisiéramos que nuestro botón "Nueva cotización" reemplace toda la página, podríamos usarlo `data-turbo-frame="_top"` así:



Boceto del índice Quotes# con el marco _top representado

Por supuesto, cada página tiene el *marco* "_top" por defecto, por lo que nuestra `Quotes#new` página también lo tiene:



Boceto de la página de citas#nueva con el marco superior representado

Para que nuestro marcado coincida con nuestros bocetos en la `Quotes#index` página, indiquemos a nuestro enlace "Nueva cita" que apunte al *marco* "_top" :

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

<main class="container">
  <%= turbo_frame_tag "first_turbo_frame" do %>
    <div class="header">
      <h1>Quotes</h1>
      <%= link_to "New quote",
                  new_quote_path,
                  data: { turbo_frame: "_top" },
                  class: "btn btn--primary" %>
    </div>
  <% end %>

  <%= render @quotes %>
</main>
```

Podemos añadir lo que queramos a la `Quotes#new` página. No importa, ya que el navegador reemplazará la página completa. Para nuestro ejemplo, simplemente volveremos a nuestro estado inicial:

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

<main class="container">
  <%= link_to sanitize("&larr; Back to quotes"), quotes_path %>

  <div class="header">
    <h1>New quote</h1>
  </div>

  <%= render "form", quote: @quote %>
</main>
```

Ahora, experimentemos nuevamente. Naveguemos hasta la `Quotes#index` página y hagamos clic en el botón "Nueva cita". Podemos ver que toda la página se reemplaza por el contenido de la `Quotes#new` página.

Al utilizar la palabra clave "_top", la URL de la página cambia a la URL de la página de destino, lo que constituye otra diferencia con respecto al uso de un Turbo Frame normal .

Como podemos observar, los Turbo Frames son una incorporación importante a nuestra caja de herramientas como desarrolladores de Ruby on Rails. Nos permiten dividir páginas en contextos independientes sin tener que escribir ningún código JavaScript personalizado.

Con esas tres reglas, tenemos más que suficiente conocimiento de Turbo Frames para construir nuestro editor de citas, pero aún necesitamos aprender dos cosas:

- Cómo utilizar Turbo Frames en combinación con el `TURBO_STREAM` formato
- Cómo nombrar nuestros Turbo Frames con algunas buenas convenciones

¡Practiquemos y hagamos que nuestras pruebas del sistema pasen! Pero antes, restablezcamos `Quotes#index` el marcado de nuestra página a su estado inicial:

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

<main class="container">
  <div class="header">
```

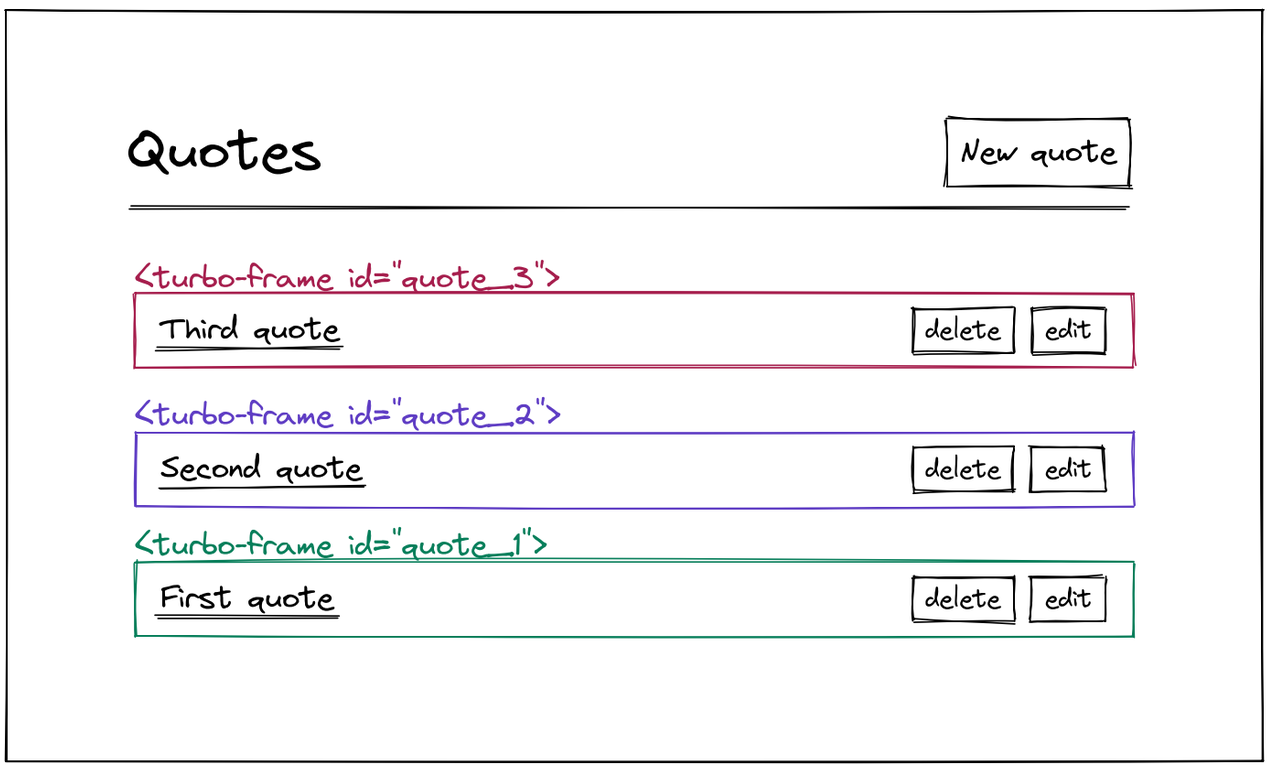
```
    <h1>Quotes</h1>
    <%= link_to "New quote", new_quote_path, class: "btn btn--primary" %>
  </div>

  <%= render @quotes %>
</main>
```

Edición de citas con Turbo Frames

Comencemos con la función de edición de cotizaciones, ya que es la más sencilla de las dos.

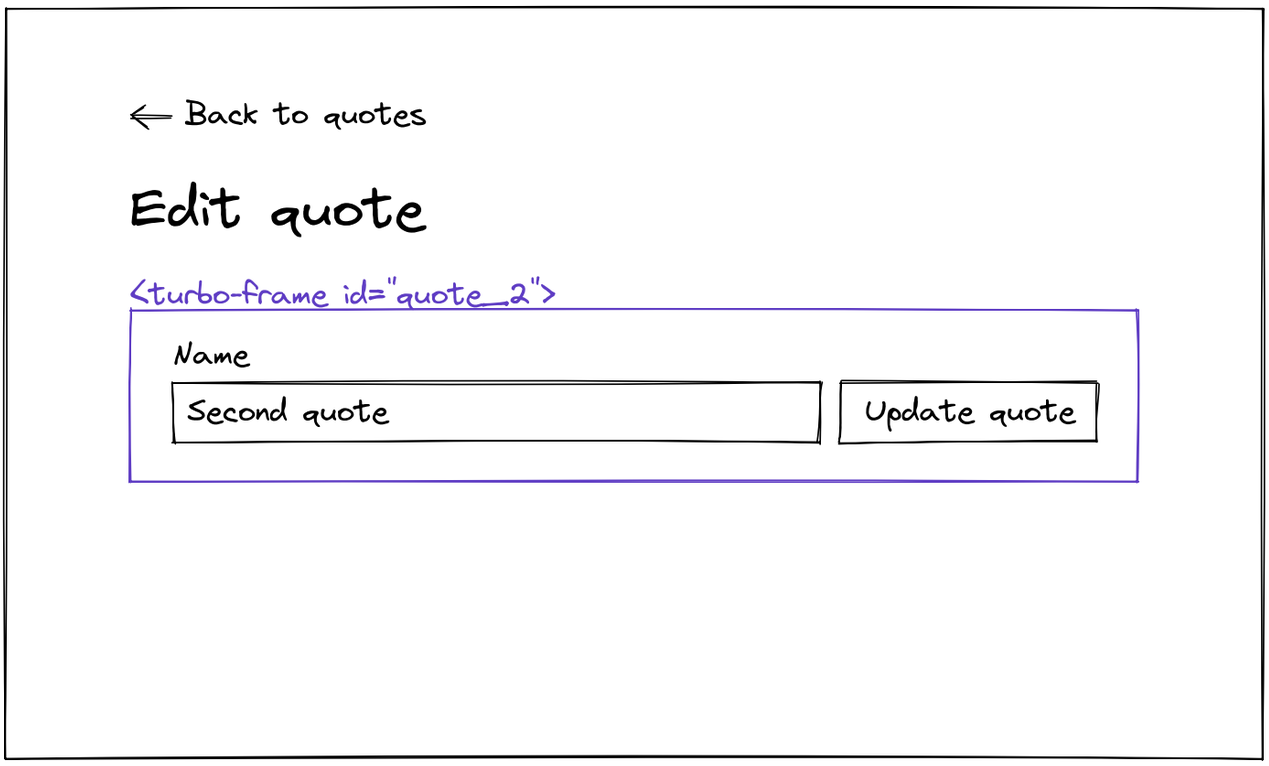
Lo primero que debemos conseguir es que al pulsar el botón "Editar" de una cita en la `Quotes#index` página, la tarjeta que contiene la cita sea sustituida por una tarjeta que contenga el formulario de edición. Sustituir partes de una página web es precisamente el tipo de trabajo que Turbo Frames puede hacer por nosotros. Pero, ¿qué identificación debemos darle a nuestros Turbo Frames?

En la `Quotes#index` página, cada Turbo Frame alrededor de cada tarjeta de cita debe tener un identificador único. Una buena convención es utilizar el nombre singular del modelo seguido del identificador de la cita. Dibujemos un boceto de cómo debería verse:



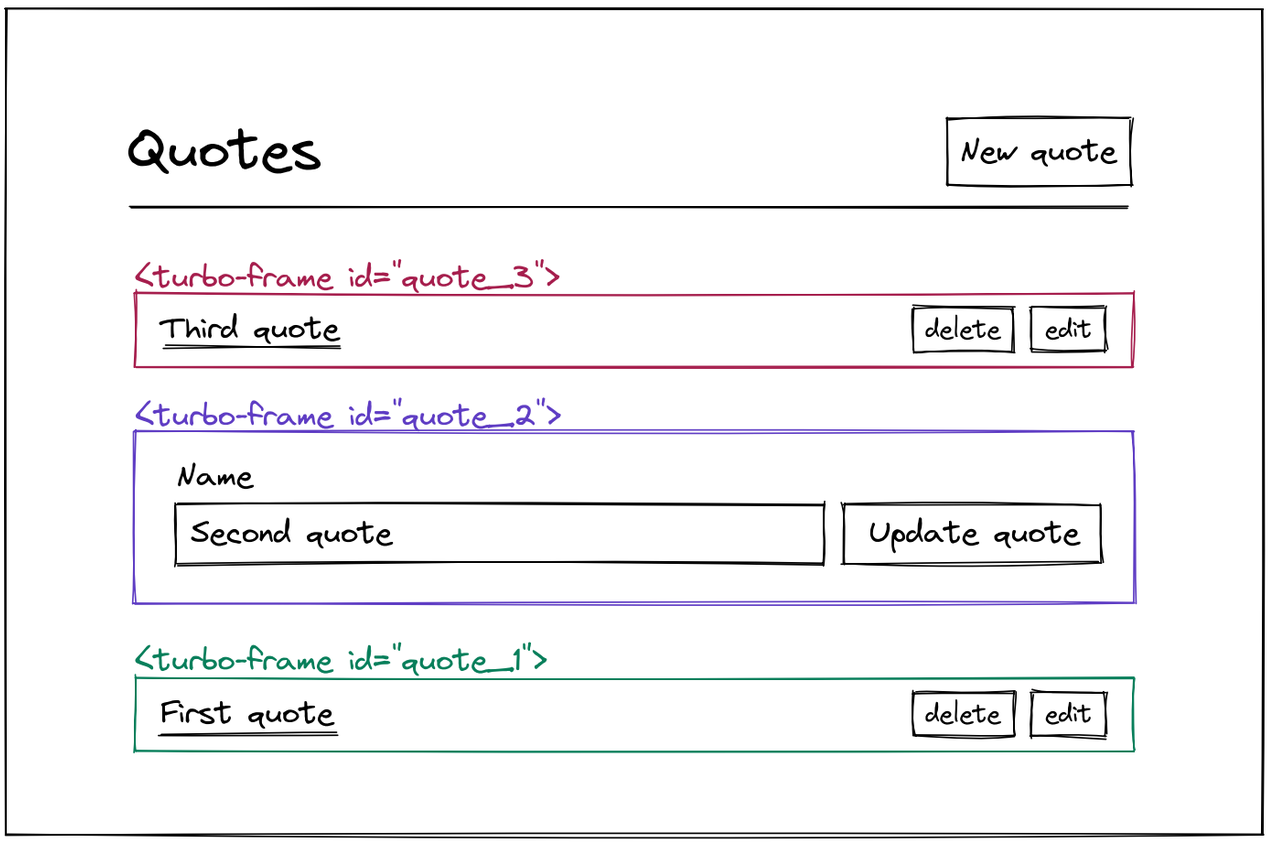
Boceto de la página de índice de citas con Turbo Frames alrededor de cada cita

Ahora supongamos que queremos editar la **segunda cita** . Al hacer clic en el botón "Editar" de la **segunda cita** , necesitamos un Turbo Frame con el mismo ID en la `Quotes#edit` página, como se describe en el siguiente esquema:



Boceto de la página de edición de citas con el marco Turbo alrededor del formulario

Con nuestros Turbo Frames nombrados apropiadamente, al hacer clic en el botón "Editar" de la segunda cita en la `Quotes#index` página, el contenido del Turbo Frame que contiene el formulario debe reemplazar el contenido del Turbo Frame que contiene la segunda tarjeta de cita como se describe en el siguiente boceto:



Boceto de la página de índice de citas con formulario para editar la segunda cita

Con esos bocetos en mente y las reglas de la sección anterior, implementemos este comportamiento. En la `Quotes#index` página, envolvamos cada cita en un Turbo Frame con un id de `"quote_#{quote_id}"`. Como cada tarjeta de cita en la `Quotes#index` página se renderiza a partir del `_quote.html.erb` parcial, simplemente necesitamos envolver cada cita dentro de un Turbo Frame con este id:

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

<%= turbo_frame_tag "quote_#{quote.id}" do %>
  <div class="quote">
    <%= link_to quote.name, quote_path(quote) %>
    <div class="quote__actions">
      <%= button_to "Delete",
                    quote_path(quote),
                    method: :delete,
                    class: "btn btn--light" %>
      <%= link_to "Edit",
                  edit_quote_path(quote),
                  class: "btn btn--light" %>
    </div>
  </div>
<% end %>
```

Necesitamos un Turbo Frame del mismo id alrededor del formulario de la `Quotes#edit` página:

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

<main class="container">
  <%= link_to sanitize("&larr; Back to quote"), quote_path(@quote) %>

  <div class="header">
    <h1>Edit quote</h1>
  </div>

  <%= turbo_frame_tag "quote_#{@quote.id}" do %>
    <%= render "form", quote: @quote %>
  <% end %>
</main>
```

Ahora, con solo esas cuatro líneas de código agregadas, probemos nuestro código en el navegador. Hagamos clic en el botón "Editar" para obtener una cotización. El formulario reemplaza correctamente la tarjeta de cotización.

Enviemos el formulario para ver si funciona como se espera.

Primero, envíenos un **formulario en blanco no válido** :

1. Al hacer clic en el botón "Actualizar cotización", Turbo intercepta el evento de envío ya que el formulario está anidado dentro de un Turbo Frame.
2. El envío del formulario no es válido, por lo que el controlador muestra la `app/quotes/edit.html.erb` vista con los errores en el formulario.
3. Gracias al estado de respuesta 422 agregado por la `status: :unprocessable_entity` opción, Turbo sabe que debe reemplazar el contenido del Turbo Frame con el nuevo que contiene los errores.
4. Los errores se muestran correctamente en la página.

Ahora vamos a enviar un **formulario válido** :

1. Al hacer clic en el botón "Actualizar cotización", Turbo intercepta el evento de envío ya que el formulario está anidado dentro de un Turbo Frame.
2. El envío del formulario es válido en el lado del controlador, por lo que éste redirecciona a la `Quotes#index` página.
3. La `Quotes#index` página actualizada contiene un Turbo Frame con la misma identificación que contiene una tarjeta con el nombre de la cita actualizada.
4. Turbo reemplaza el contenido del marco que contiene el formulario con el contenido del marco que contiene la tarjeta de cotización actualizada.

Ahora todo funciona como se esperaba para las acciones `#edit` y `#update` .

Sin embargo, es posible que notes un comportamiento inesperado. Hacer clic en el enlace para mostrar una cita ya no funciona y es posible que veas un error en la consola al eliminar una cita. Hablaremos de esto muy pronto (¡tiene que ver con la regla número 2!), pero antes de continuar, hablemos del `dom_id` asistente que nos ayudará a escribir identificadores de Turbo Frame más limpios.

Turbo Frames y el asistente dom_id

Hay una cosa más que debes saber sobre el `turbo_frame_tag` ayudante. Puedes pasarle una cadena o cualquier objeto que se pueda convertir en un `dom_id`. El `dom_id` ayudante nos ayuda a convertir un objeto en un identificador único como este:

```
# If the quote is persisted and its id is 1:
dom_id(@quote) # => "quote_1"

# If the quote is a new record:
dom_id(Quote.new) # => "new_quote"

# Note that the dom_id can also take an optional prefix argument
# We will use this later in the tutorial
dom_id(Quote.new, "prefix") # "prefix_new_quote"
```

El `turbo_frame_tag` asistente pasa automáticamente el objeto dado a `dom_id`. Por lo tanto, podemos refactorizar nuestras dos `turbo_frame_tag` llamadas en nuestras `Quotes#index` vistas `Quotes#edit` y pasando un objeto en lugar de una cadena. Los siguientes bloques de código son equivalentes:

```
<%= turbo_frame_tag "quote_#{@quote.id}" do %>
  ...
<% end %>

<%= turbo_frame_tag dom_id(@quote) do %>
  ...
<% end %>

<%= turbo_frame_tag @quote %>
  ...
<% end %>
```

Refactoricemos el código que acabamos de escribir para utilizar este *azúcar sintáctico* :

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

<%= turbo_frame_tag quote do %>
  <div class="quote">
    <%= link_to quote.name, quote_path(quote) %>
    <div class="quote__actions">
      <%= button_to "Delete",
```

```
                    quote_path(quote),
                    method: :delete,
                    class: "btn btn--light" %>
      <%= link_to "Edit",
                  edit_quote_path(quote),
                  class: "btn btn--light" %>
    </div>
  </div>
<% end %>
```

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

<main class="container">
  <%= link_to sanitize("&larr; Back to quote"), quote_path(@quote) %>

  <div class="header">
    <h1>Edit quote</h1>
  </div>

  <%= turbo_frame_tag @quote do %>
    <%= render "form", quote: @quote %>
  <% end %>
</main>
```

Ahora que nuestros Turbo Frames tienen grandes nombres, podemos continuar con nuestro trabajo.

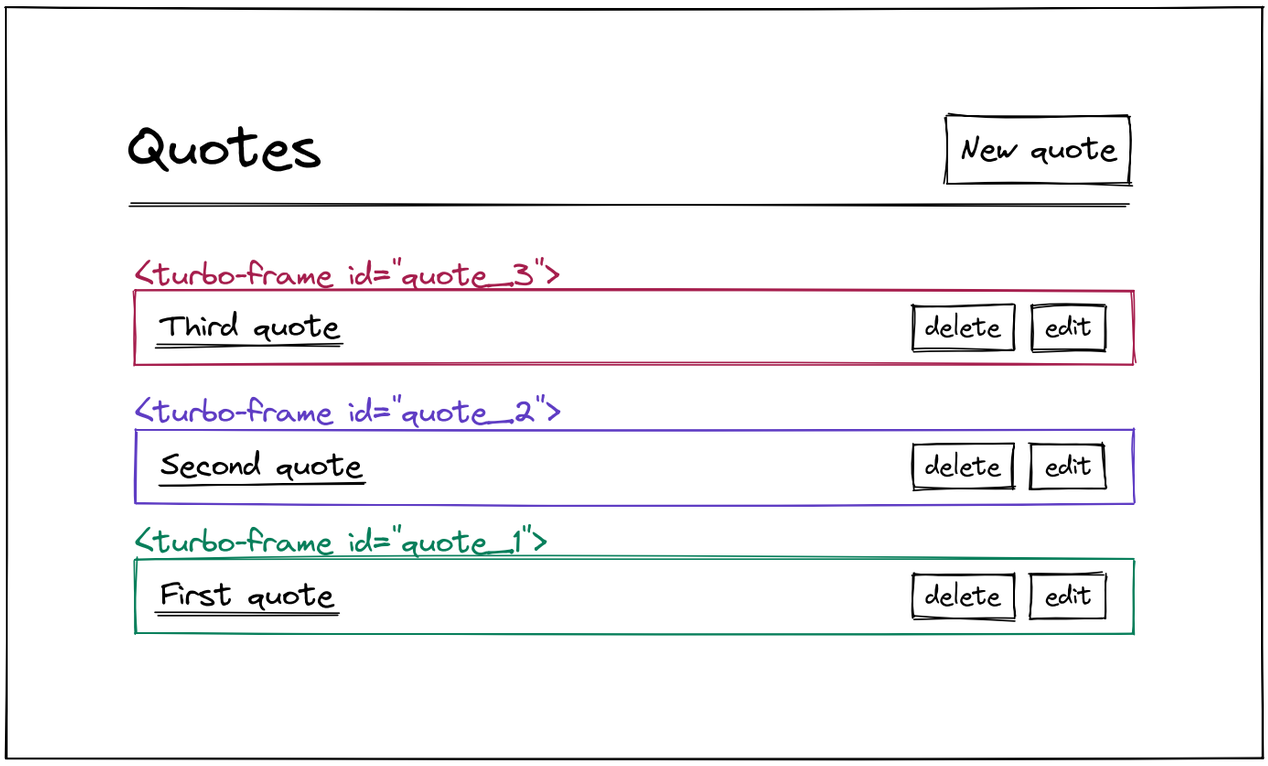
Mostrar y eliminar citas

Nuestra implementación anterior funciona bien para las acciones `#edit` y `#update`, pero introdujimos dos problemas nuevos:

1. El enlace para mostrar una cita no funciona como se esperaba: el Turbo Frame que contiene la cita desaparece y se registra un error en la consola.
2. El botón para destruir una cotización registra un error en la consola.

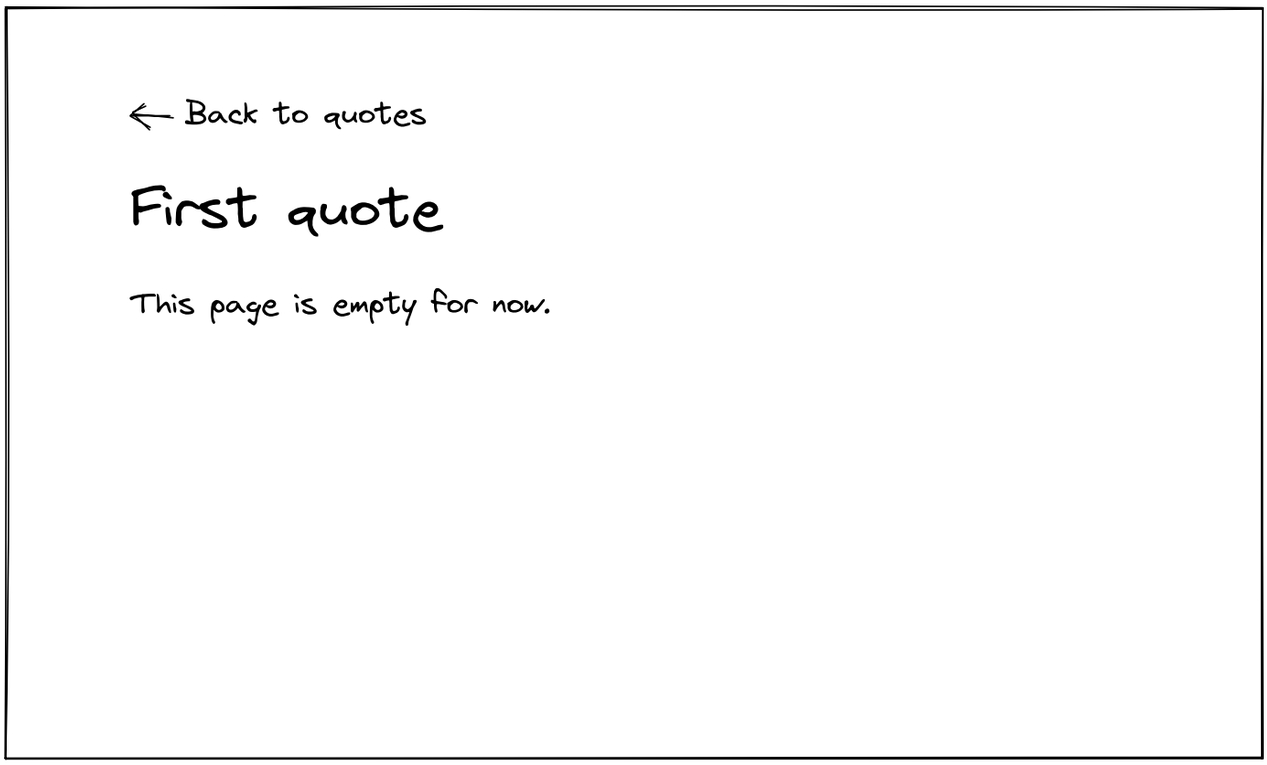
Ambos problemas tienen que ver con la regla número 2 de Turbo Frames que vimos antes. Vamos a resolverlos uno por uno.

Nuestra `Quotes#index` página actualmente se ve así:



Boceto de la página de índice de citas con Turbo Frames alrededor de cada cita

Como podemos ver, el enlace para mostrar una cita está anidado dentro de un Turbo Frame de id `"quote_#{quote.id}"`. Sin embargo, en la `Quotes#show` página no hay ningún Turbo Frame con el mismo id:



Boceto de las citas#mostrar página

Turbo espera un Turbo Frame con el mismo ID en la `Quotes#show` página. Para solucionar el problema, haremos que los enlaces a la `Quote#show` página apunten al `"_top"` frame para reemplazar toda la página:

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

```
<%= turbo_frame_tag quote do %>
  <div class="quote">
    <%= link_to quote.name,
                quote_path(quote),
                data: { turbo_frame: "_top" } %>
    <div class="quote__actions">
      <%= button_to "Delete",
                    quote_path(quote),
                    method: :delete,
                    class: "btn btn--light" %>
      <%= link_to "Edit",
                  edit_quote_path(quote),
                  class: "btn btn--light" %>
    </div>
  </div>
<% end %>
```

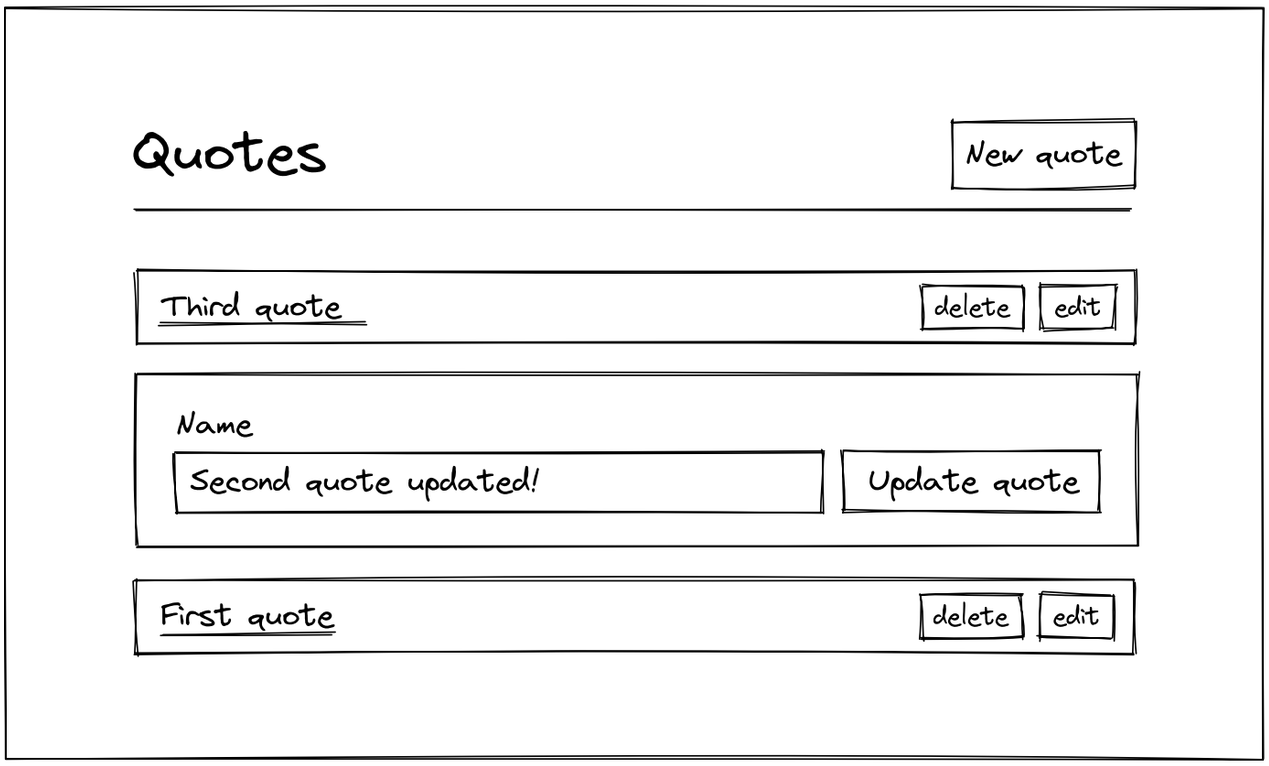
Probémoslo en el navegador. Nuestro primer problema está resuelto. ¡Nuestros enlaces a la `Quotes#show` página ahora funcionan como se esperaba!

Podríamos resolver el segundo problema con el mismo método haciendo que el formulario para eliminar la cita tenga como destino el marco "_top" :

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

<%= turbo_frame_tag quote do %>
  <div class="quote">
    <%= link_to quote.name,
                quote_path(quote),
                data: { turbo_frame: "_top" } %>
    <div class="quote__actions">
      <%= button_to "Delete",
                    quote_path(quote),
                    method: :delete,
                    form: { data: { turbo_frame: "_top" } },
                    class: "btn btn--light" %>
      <%= link_to "Edit",
                  edit_quote_path(quote),
                  class: "btn btn--light" %>
    </div>
  </div>
<% end %>
```

Si probamos en el navegador, funciona como se esperaba. ¡Ya no hay ningún error en la consola!

Si bien esta es una solución perfectamente válida, tiene un efecto secundario no deseado que tal vez queramos abordar. Imaginemos que abrimos el formulario para la **segunda cotización** y hacemos clic en el botón "Eliminar" para la **tercera cotización,** como en este ejemplo:



Boceto de la página de índice de citas con un formulario para editar la segunda cita

Continúe y pruébelo en el navegador. La **tercera cita** se elimina como se esperaba, pero la respuesta también cierra el formulario para la **segunda cita** . Esto se debe a que, como el formulario para eliminar la **tercera cita** apunta al `"_top"` marco, se reemplaza toda la página.

Sería bueno si pudiéramos eliminar únicamente el Turbo Frame que contiene la cita eliminada y dejar el resto de la página sin cambios para preservar el *estado* de la página. Bueno, Turbo y Rails nos respaldan una vez más. Eliminemos lo que acabamos de hacer con el botón "Eliminar":

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

<%= turbo_frame_tag quote do %>
  <div class="quote">
    <%= link_to quote.name,
                quote_path(quote),
                data: { turbo_frame: "_top" } %>
```

```
    <div class="quote__actions">
      <%= button_to "Delete",
                    quote_path(quote),
                    method: :delete,
                    class: "btn btn--light" %>
      <%= link_to "Edit",
                  edit_quote_path(quote),
                  class: "btn btn--light" %>
    </div>
  </div>
<% end %>
```

Es hora de una introducción al `TURBO_STREAM` formato.

El formato Turbo Stream

Los formularios en Rails 7 ahora se envían con el `TURBO_STREAM` formato. Destruyamos una cita e inspeccionemos lo que sucede en el registro de nuestro servidor Rails:

```
Started DELETE "/quotes/908005781" for 127.0.0.1 at 2022-01-27 15:30:13
Processing by QuotesController#destroy as TURBO_STREAM
```

Como podemos ver, `QuotesController` procesará la `#destroy` acción con el `TURBO_STREAM` formato. Exploremos lo que podemos hacer con este formato haciendo que nuestra acción de destrucción solo elimine el Turbo Frame que contiene la cita eliminada y dejando el resto de la página intacta.

En el controlador, admitamos tanto los `HTML` formatos `TURBO_STREAM` como gracias al `respond_to` método:

```
# app/controllers/quotes_controller.rb

def destroy
  @quote.destroy

  respond_to do |format|
    format.html { redirect_to quotes_path, notice: "Quote was successful
    format.turbo_stream
```

```
  end
end
```

Al igual que con cualquier otro formato, creemos la vista correspondiente:

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

<%= turbo_stream.remove "quote_#{@quote.id}" %>
```

Eliminemos una cita e inspeccionemos el cuerpo de la respuesta en la pestaña "Red" del navegador. El HTML recibido por el navegador debería verse así, excepto que probablemente tenga una identificación diferente para la cita:

```
<turbo-stream action="remove" target="quote_908005780">
</turbo-stream>
```

¿De dónde proviene este HTML? En la `TURBO_STREAM` vista que acabamos de crear, el `turbo_stream` asistente recibió el `remove` método con el `"quote_#{@quote.id}"` como argumento. Como podemos ver, este asistente convierte este en un `<turbo-stream>` elemento personalizado con la acción "eliminar" y el objetivo "quote_908005780".

Cuando el navegador recibe este HTML, Turbo sabrá cómo interpretarlo y realizará la **acción** deseada en el Turbo Frame con el id especificado por el atributo **target** . En nuestro caso, Turbo *elimina* el Turbo Frame correspondiente a la cita eliminada y deja el resto de la página intacta. ¡Eso es exactamente lo que queríamos!

Nota : Al momento de escribir este capítulo, el `turbo_stream` ayudante responde a los siguientes métodos, de modo que puede realizar las siguientes acciones:

```
# Remove a Turbo Frame
turbo_stream.remove

# Insert a Turbo Frame at the beginning/end of a list
turbo_stream.append
turbo_stream.prepend
```

```
# Insert a Turbo Frame before/after another Turbo Frame
turbo_stream.before
turbo_stream.after

# Replace or update the content of a Turbo Frame
turbo_stream.update
turbo_stream.replace
```

Por supuesto, a excepción del `remove` método, el `turbo_stream` ayudante espera un valor *parcial* y *local* como argumentos para saber qué código HTML necesita *agregar* , *anteponer* o *reemplazar* del DOM. En la siguiente sección, aprenderemos cómo pasar *valores parciales* y *locales* al `turbo_stream` ayudante.

Con la combinación de Turbo Frames y el nuevo `TURBO_STREAM` formato, podremos realizar operaciones precisas sobre partes de nuestras páginas web sin tener que escribir una sola línea de JavaScript, preservando así el *estado* de nuestras páginas web.

Una última cosa antes de pasar a la siguiente sección: el `turbo_stream` ayudante también se puede utilizar con `dom_id` . Podemos refactorizar nuestra vista de esta manera:

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

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

Es hora de abordar la última característica: nuestra creación de cotizaciones.

Creando una nueva cotización con Turbo Frames

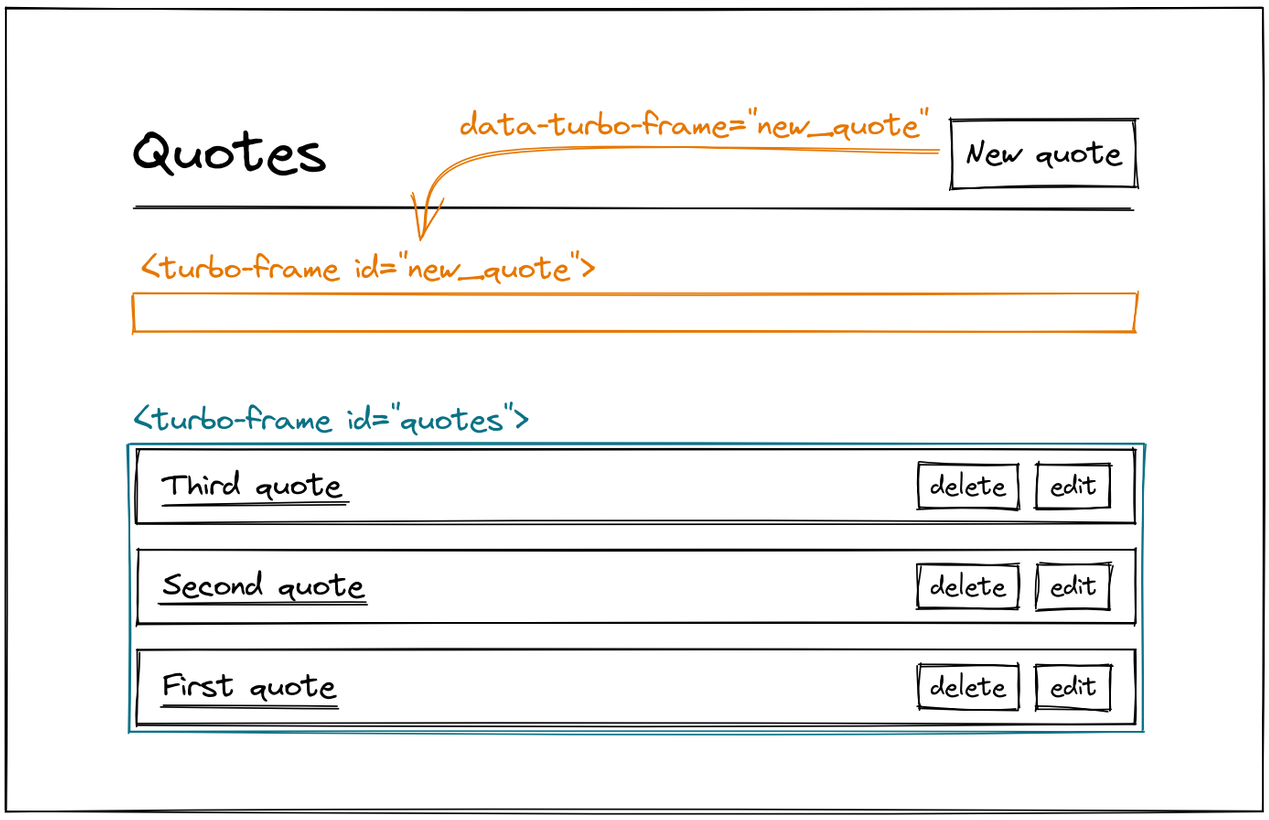
La última característica que necesitamos es la creación de cotizaciones. Antes de sumergirnos en la implementación, dibujemos algunos bocetos de lo que crearemos.

Al hacer clic en el botón "Nueva cotización", ya no accederemos a la `Quotes#new` página. En su lugar, el nuevo formulario de cotización aparecerá justo debajo del encabezado de la `Quotes#index` página. Luego, al hacer clic en el botón "Crear cotización", se *agregará* la cotización recién creada a la lista y

se eliminará el nuevo formulario de cotización de la página. Para ello, necesitaremos dos Turbo Frames más:

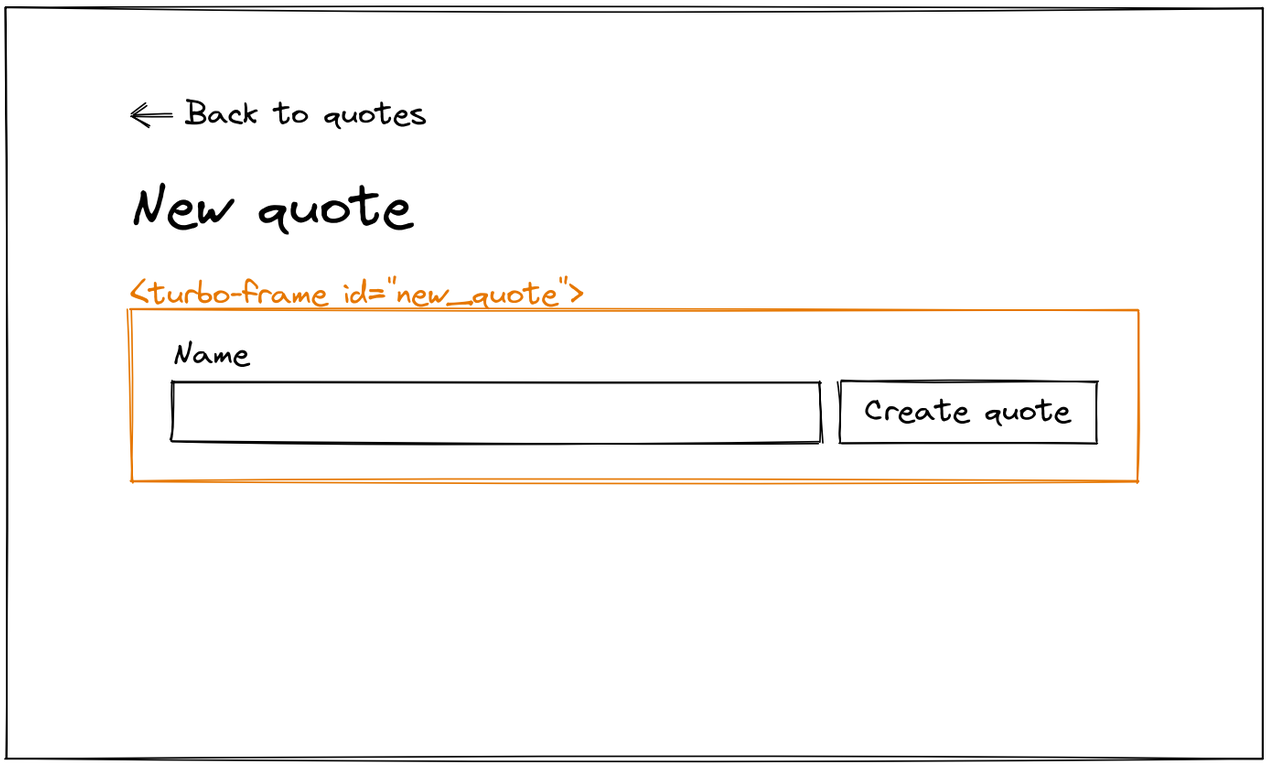
- Un Turbo Frame vacío que recibirá el nuevo formulario de cotización.
- Un Turbo Frame que envuelve la lista de citas para que podamos *anteponer* la cita recién creada en la posición correcta.

Esto se describe en el siguiente esquema:



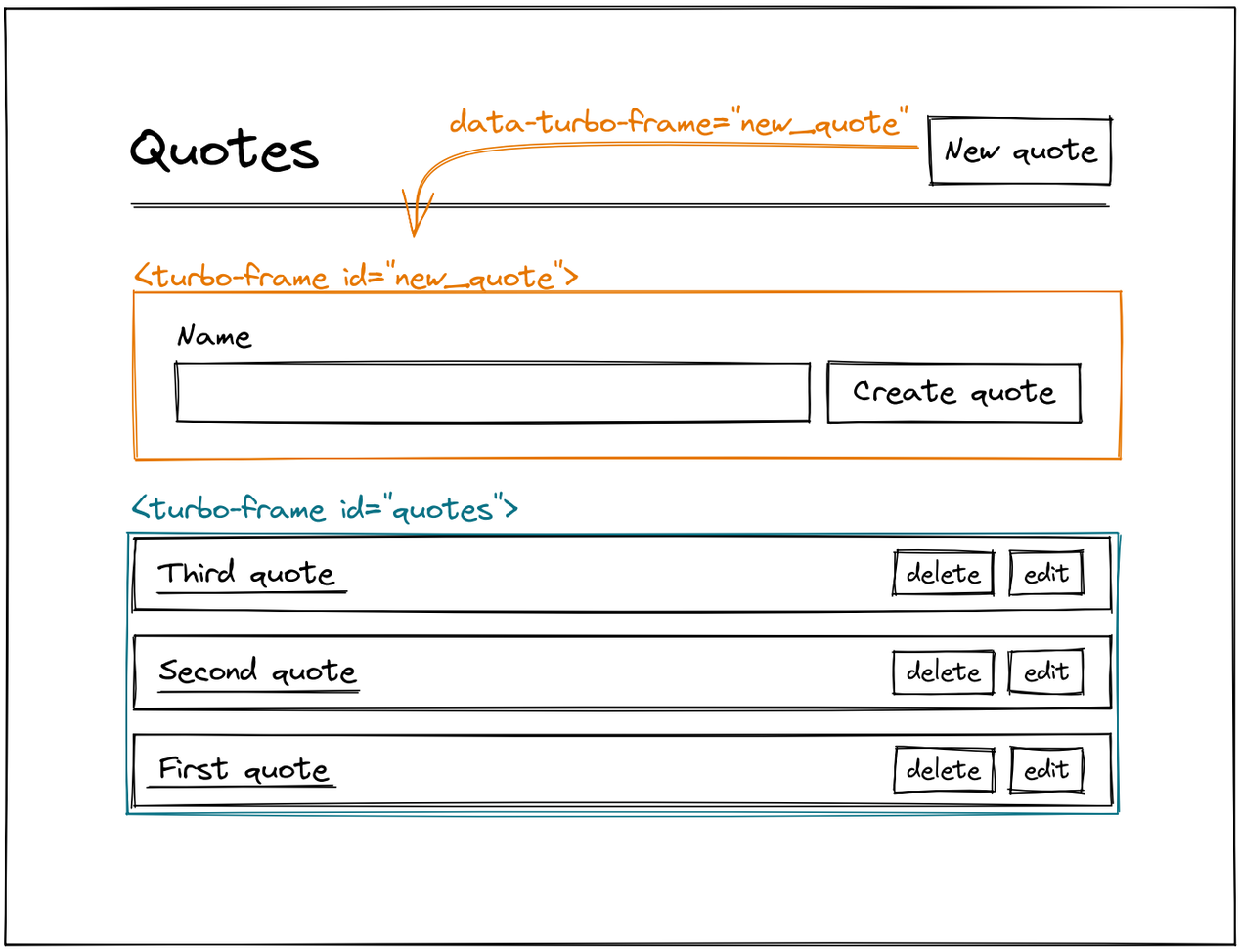
Boceto de la página de citas#índice

En la `Quotes#new` página, envolveremos el formulario en un Turbo Frame con el mismo id que el vacío en la `Quotes#index` página:



Boceto de la página de citas n.° nueva con el formulario envuelto en un Turbo Frame

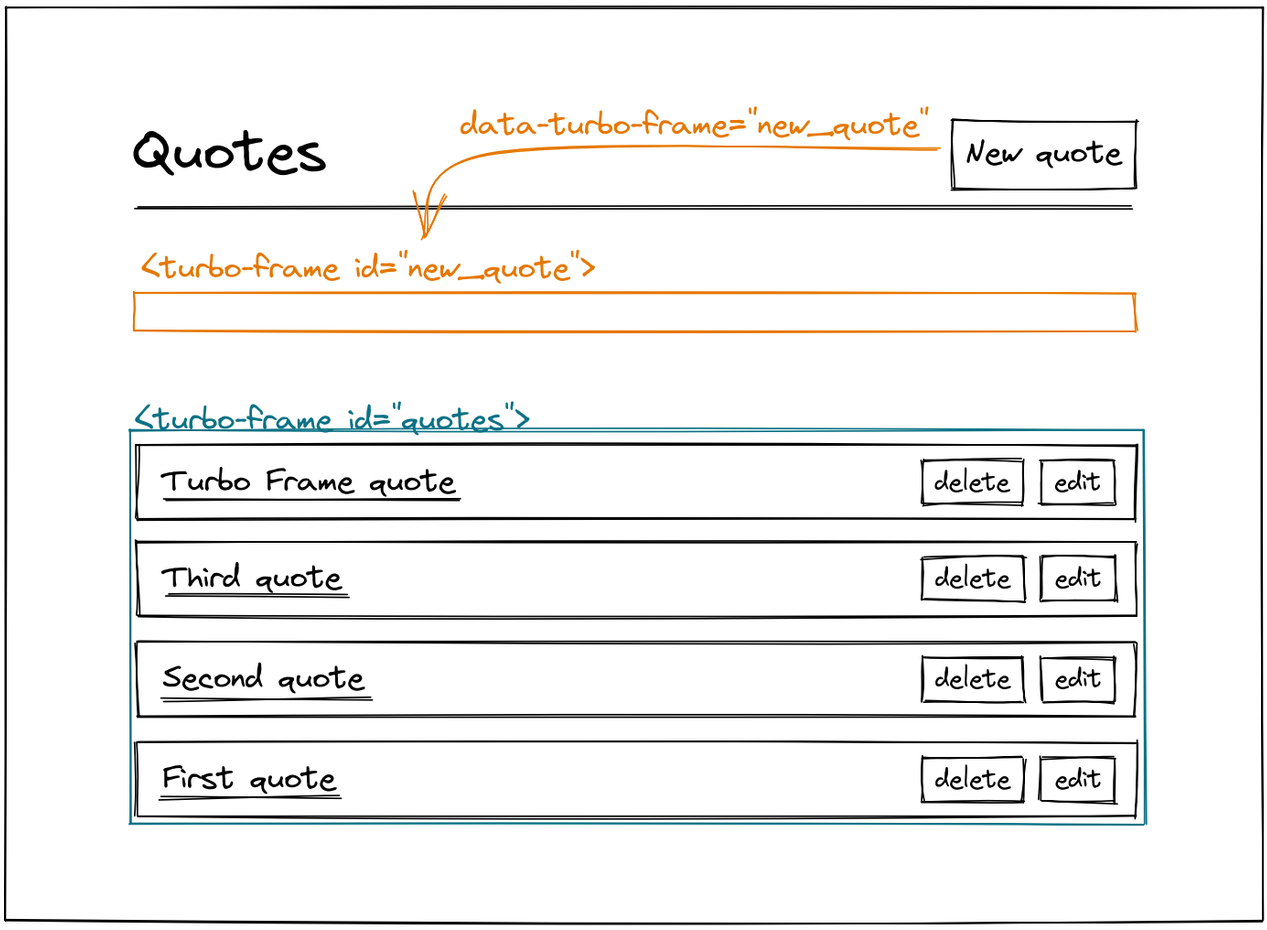
Con nuestros Turbo Frames apropiadamente nombrados, al hacer clic en el botón "Nueva cotización" se reemplazará el contenido vacío del `"new_quote"` marco con el formulario para crear una nueva cotización:



Boceto de la página de índice de cotizaciones con el nuevo formulario de cotización debajo del encabezado

Al enviar el formulario, aprovecharemos el poder de Turbo Streams para *anteponer* la nueva cita a la lista de citas y *actualizar* el `"new_quote"` marco para

que quede vacío nuevamente:



Boceto de la página Quotes#index con la cita creada antepuesta a la lista de citas

Implementemos nuestra solución.

Como se mencionó anteriormente, necesitamos dos marcos con el mismo id en la `Quotes#index` vista y la `Quotes#new` página. Esos marcos tendrán el id de `dom_id(Quote.new)`, que es equivalente a la cadena `"new_quote"`.

En la `Quotes#new` página, envolvamos el nuevo formulario de cotización en el Turbo Frame:

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

<main class="container">
  <%= link_to sanitize("&larr; Back to quotes"), quotes_path %>

  <div class="header">
    <h1>New quote</h1>
  </div>

  <%= turbo_frame_tag @quote do %>
    <%= render "form", quote: @quote %>
```

```
  <% end %>
</main>
```

Aquí `@quote` hay un nuevo registro, por lo que las tres expresiones siguientes son equivalentes:

```
turbo_frame_tag "new_quote"
turbo_frame_tag Quote.new
turbo_frame_tag @quote
```

Ahora agreguemos un Turbo Frame vacío con la misma identificación a la `Quotes#index` página que recibirá este nuevo formulario de cotización:

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

<main 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 %>
  <%= render @quotes %>
</main>
```

Como puede ver, el marco está vacío en la `Quotes#index` página. El nuevo formulario de cotización solo debería aparecer al hacer clic en el botón "Nueva cotización". Para vincular nuestro botón "Nueva cotización" con el Turbo Frame correcto que acabamos de crear, tenemos que usar el `data-turbo-frame` atributo de datos:

Como podemos ver, el `data-turbo-frame` atributo del enlace coincide con el id del Turbo Frame vacío, lo que conecta los dos. Analicemos lo que sucede aquí:

1. Al hacer clic en el enlace "Nueva cotización", el clic será interceptado por Turbo.
2. Turbo sabe que tiene que interactuar con el marco de id `new_quote` gracias al atributo `data-turbo-frame` en el enlace "Nueva cotización".

3. La solicitud se envía en AJAX y nuestro servidor renderizará la `Quotes#new` página con un marco con id `new_quote`.
4. Cuando el navegador recibe el HTML, Turbo extraerá el marco con el id de `new_quote` la `Quotes#new` página y reemplazará el marco vacío con el mismo id en la `Quotes#index` página.

Probémoslo en nuestro navegador. ¡Funciona! ¡Nuestro formulario aparece en la página como se esperaba!

Ahora, intentemos enviar el formulario **con un nombre en blanco** haciendo clic en el botón "Crear cotización". La cotización no será válida y los errores deberían aparecer en la página. Expliquemos qué sucede aquí:

1. Al hacer clic en el botón “Crear cotización”, el envío del formulario es interceptado por Turbo.
2. El formulario está envuelto en un marco con id `new_quote`, por lo que Turbo sabe que solo necesita reemplazar este marco.
3. El servidor recibe los parámetros no válidos en la `QuotesController#create` acción y muestra la `Quotes#new` vista con el formulario que contiene errores.
4. Cuando el navegador recibe la respuesta con `status: :unprocessable_entity`, reemplaza el marco con id `new_quote` por el nuevo que contiene errores.

Ya casi estamos. Para completar la función tal como está diseñada en nuestro boceto, debemos anteponer la cita recién creada a la lista de citas cuando se le asigna un nombre válido.

Si lo probamos ahora en el navegador, nos daremos cuenta de que la cita se crea en la base de datos, pero la cita creada no se agrega a la lista de citas.

¿Porqué es eso?

Al enviar el formulario con atributos válidos, la `QuotesController#create` acción mostrará la `Quotes#index` página que contiene un marco vacío de id `new_quote` que reemplazará nuestro formulario. Sin embargo, Turbo no sabe qué hacer con la cita recién creada. ¿Dónde debería insertarse en la página? ¿Debería agregarse a una lista de citas? ¿O quizás anteponerse? Para hacer esto, ¡usaremos una vista de Turbo Stream!

Digamos `QuotesController` que necesita soportar tanto el formato `HTML` como el `TURBO_STREAM`:

```
# app/controllers/quotes_controller.rb

def create
  @quote = Quote.new(quote_params)

  if @quote.save
    respond_to do |format|
      format.html { redirect_to quotes_path, notice: "Quote was successf
      format.turbo_stream
    end
  else
    render :new, status: :unprocessable_entity
  end
end
```

Vamos a crear la vista correspondiente:

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

<%= turbo_stream.prepend "quotes", partial: "quotes/quote", locals: { qu
<%= turbo_stream.update Quote.new, "" %>
```

En esta vista, le indicamos a Turbo que haga dos cosas:

1. La primera línea le indica a Turbo que *anteponga* `quotes` el parcial al Turbo Frame con id `app/views/quotes/_quote.html.erb`. Como podemos ver, es sencillo pasar un *parcial* y *variables locales* al `turbo_stream` ayudante.
2. La segunda línea le dice a Turbo que actualice el Turbo Frame con id `new_quote` con contenido vacío.

Lo último que debemos hacer para que funcione es agregar un Turbo Frame con id "quotes" para envolver la lista de citas en la `Quotes#index` página.

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

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

Ahora, probémoslo en el navegador. ¡Funciona! Si inspeccionamos la pestaña "Red" en las herramientas de desarrollo al enviar un nuevo formulario de cotización válido, el cuerpo de la respuesta debería verse así:

```
<turbo-stream action="prepend" target="quotes">
  <template>
    <turbo-frame id="quote_123">
      <!-- The HTML for the quote partial -->
    <turbo-frame>
  </template>
</turbo-stream>

<turbo-stream action="update" target="new_quote">
  <template>
    <!-- An empty template! -->
  </template>
</turbo-stream>
```

Como puede ver, coincide con nuestras dos líneas en la `create.turbo_stream.erb` vista que acabamos de crear traducidas a un lenguaje que Turbo puede entender. Al recibir la respuesta, Turbo ejecuta `action` (agregar, anteponer, reemplazar, eliminar) en el `target` Turbo Frame.

Nota : Existen distintas formas de escribir lo mismo en las vistas de Turbo Stream. Veamos la `create.turbo_stream.erb` vista que acabamos de crear.

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

```
<%= turbo_stream.prepend "quotes", partial: "quotes/quote", locals: { qu
<%= turbo_stream.update Quote.new, "" %>
```

Si bien esta es una forma perfectamente válida de escribir nuestra vista, hay otra sintaxis con un bloque que a veces uso cuando las líneas son demasiado largas:

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

<%= turbo_stream.prepend "quotes" do %>
  <%= render partial: "quotes/quote", locals: { quote: @quote } %>
<% end %>

<%= turbo_stream.update Quote.new, "" %>
```

En Ruby on Rails, las siguientes dos expresiones son equivalentes:

```
render partial: "quotes/quote", locals: { quote: @quote }
render @quote
```

Con esto en mente, podemos acortar nuevamente la forma en que escribimos nuestra vista:

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

<%= turbo_stream.prepend "quotes" do %>
  <%= render @quote %>
<% end %>

<%= turbo_stream.update Quote.new, "" %>
```

No necesitamos la sintaxis de bloque aquí porque las líneas son cortas, por lo que esta será la forma final en que escribiremos la vista:

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

<%= turbo_stream.prepend "quotes", @quote %>
<%= turbo_stream.update Quote.new, "" %>
```

Elegante, ¿verdad? ¡Esta nota fue solo una forma de aprender las diferentes formas de escribir las mismas vistas de Turbo Stream!

Solicitar nuestras cotizaciones

Hay un último detalle que debemos tener en cuenta. Hemos decidido anteponer la cita creada a la lista de citas, pero cuando actualizamos la página, el orden de las citas en la lista cambia. Para mantener siempre las citas ordenadas, comenzando por las más nuevas, agreguemos un ámbito a nuestro `Quote` modelo:

```
# app/models/quote.rb

class Quote < ApplicationRecord
  validates :name, presence: true

  scope :ordered, -> { order(id: :desc) }
end
```

Entonces usemos este alcance en nuestro controlador en la `#index` acción:

```
# app/controllers/quotes_controller.rb

def index
  @quotes = Quote.ordered
end
```

Ahora el orden de las comillas es el mismo incluso cuando actualizamos la página. Es un pequeño detalle, pero puede ser importante para que nuestros usuarios comprendan lo que sucede.

💎 Trenes calientes

que vamos a actualizar nuestra prueba del sistema para que vuelva a pasar:

```
# test/system/quotes_test.rb

setup do
  # We need to order quote as well in the system tests
  @quote = Quote.ordered.first
end
```

Ejecutemos nuestras pruebas; ¡todas deberían estar en verde! Nuestro editor de cotizaciones ahora luce exactamente como se describe en los primeros bocetos. Tuvimos que aprender algunas habilidades nuevas, pero la implementación solo consistió en unas pocas líneas de código. ¡Turbo es un software increíble!

Agregar un botón de cancelación

Ahora que todo funciona como se espera, agreguemos la última mejora a nuestra página. **Queremos que nuestros usuarios puedan cerrar formularios de cotización nuevos o de edición sin enviarlos** . Para ello, agregaremos un enlace "Cancelar" que dirija a la `Quotes#index` página del `quotes/_form.html.erb` parcial:

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

<%= simple_form_for quote, html: { class: "quote form" } do |f| %>
  <% if quote.errors.any? %>
    <div class="error-message">
      <%= quote.errors.full_messages.to_sentence.capitalize %>
    </div>
  <% end %>

  <%= f.input :name, input_html: { autofocus: true } %>
  <%= link_to "Cancel", quotes_path, class: "btn btn--light" %>
  <%= f.submit class: "btn btn--secondary" %>
<% end %>
```

Vamos a probarlo en el navegador. Gracias a la potencia de Turbo Frames, ¡ya funciona! Te explicamos qué sucede.

Cuando nuestro usuario hace clic en el enlace "Cancelar" para el **nuevo formulario de cotización** :

1. El enlace está dentro de un Turbo Frame de id `new_quote` , por lo que Turbo solo reemplazará el contenido de este frame
2. El enlace navega a la `Quotes#index` página que contiene un Turbo Frame vacío con id `new_quote`
3. Turbo reemplaza el contenido del `new_quote` marco con el contenido vacío, por lo que el formulario desaparece

Cuando nuestro usuario hace clic en el enlace "Cancelar" para **editar el formulario de cotización** :

1. El enlace está dentro de un Turbo Frame de id `dom_id(quote)` , por lo que Turbo solo reemplazará el contenido de este frame
2. El enlace navega a la `Quotes#index` página que contiene un Turbo Frame con id `dom_id(quote)` que contiene el HTML para esta cita.
3. Turbo reemplaza el contenido del `dom_id(quote)` marco que contiene el formulario con el HTML para esta cotización

Si intentamos crear una cita y editar varias citas simultáneamente, notaremos que el *estado* de la página se conserva. Por ejemplo, al crear una cita, todos los formularios de edición que estén abiertos permanecerán abiertos. Los Turbo Frames son piezas independientes de la página web que podemos manipular sin necesidad de escribir ningún JavaScript personalizado.

Envolver

En este capítulo, reemplazamos nuestro controlador CRUD clásico en el recurso de cotizaciones con una aplicación *reactiva* moderna casi sin código y sin JavaScript.

Tomémonos un tiempo para jugar con lo que hemos creado. Abramos un formulario para editar una cita, destruyamos otra cita y hagamos clic en el botón "Nueva cita". En comparación con el uso de una biblioteca de interfaz como React, no hay estados que gestionar, ni acciones complejas que enviar, ni reductores... ¡Es un placer trabajar con Turbo!

Este fue un capítulo denso, así que tomemos un descanso y asegurémonos de que todo esté claro en nuestras cabezas antes de comenzar con el siguiente.

En el próximo capítulo hablaremos sobre cómo realizar actualizaciones en tiempo real en nuestra aplicación mediante la transmisión de Turbo Streams con Action Cable. ¡Nos vemos allí!

← anterior Siguiente →

Recibir notificaciones cuando escriba nuevos artículos

Si te gustó este artículo y quieres estar al día con Ruby on Rails y Hotwire, ¡puedes suscribirte a mi boletín (sin spam, sin seguimiento, cancelar la suscripción en cualquier momento)!

Suscríbete al boletín

Github Gorjeo Hoja informativa

Hecho con remotamente