← Volver a la lista de capítulos

Otro controlador CRUD con Turbo Rails

Publicado el 7 de marzo de 2022

En este capítulo, crearemos el controlador CRUD para las fechas de nuestras cotizaciones. ¡Es la oportunidad perfecta para practicar lo que hemos aprendido desde el comienzo del tutorial!

Lo que construiremos en los siguientes tres capítulos

Ahora que nuestros usuarios pueden crear, actualizar y eliminar cotizaciones, ¡es hora de hacer que nuestro editor de cotizaciones haga algo útil!

En los próximos tres capítulos, trabajaremos en la `Quotes#show` página. Cuando terminemos esos tres capítulos, nuestros usuarios podrán agregar varias fechas a sus cotizaciones. Esas fechas tendrán varias líneas de artículo, cada una con un nombre, una descripción opcional, una cantidad y un precio unitario.

Antes de comenzar a codificar, probemos el editor de cotizaciones en hotrails.dev . Creemos una cotización y luego naveguemos a la `Quotes#show` página correspondiente. Podemos crear algunas fechas y agregar líneas de pedido a esas fechas. Cuando creamos, actualizamos o eliminamos

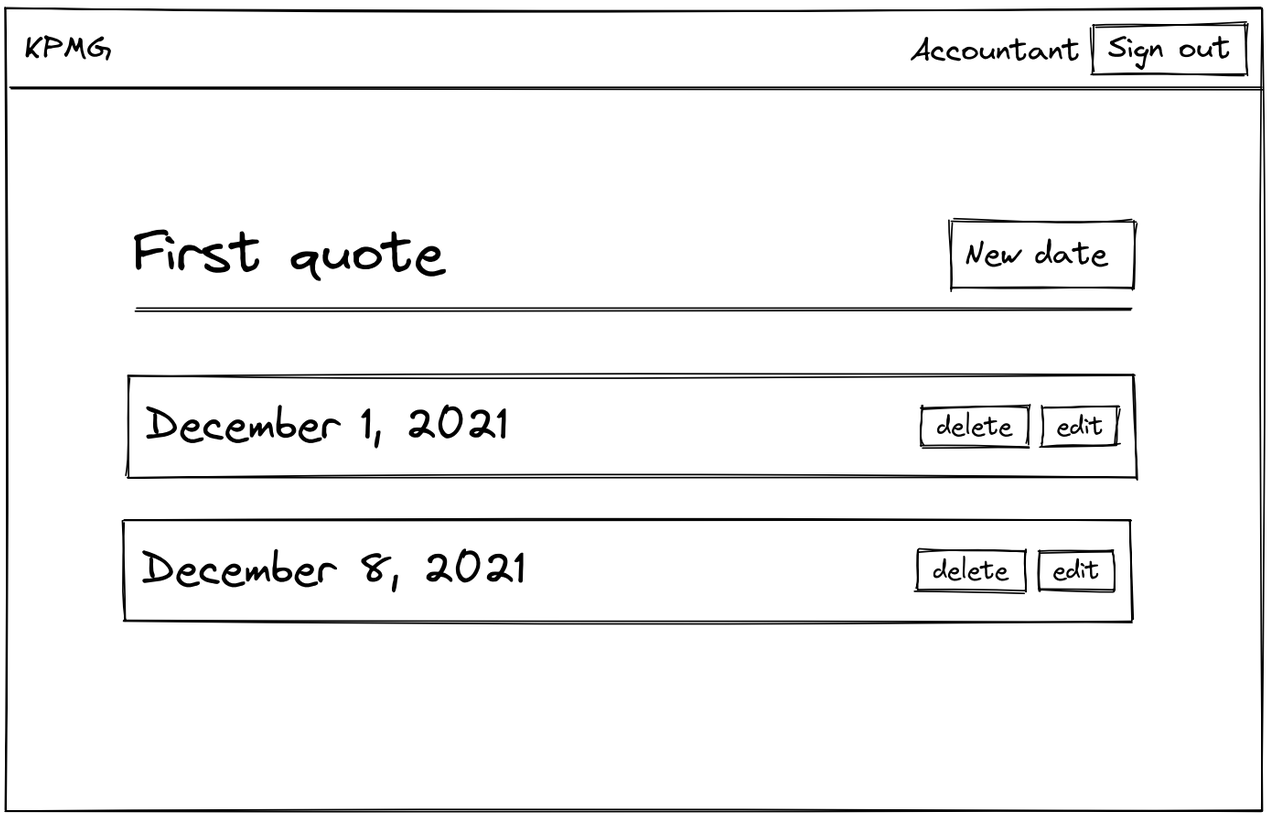
líneas de pedido, se actualiza el monto total de la cotización en la parte inferior de la página.

¡Gracias al poder de Ruby on Rails con Turbo y lo que aprendimos en los capítulos anteriores, será bastante fácil hacerlo!

Lo que construiremos en este capítulo

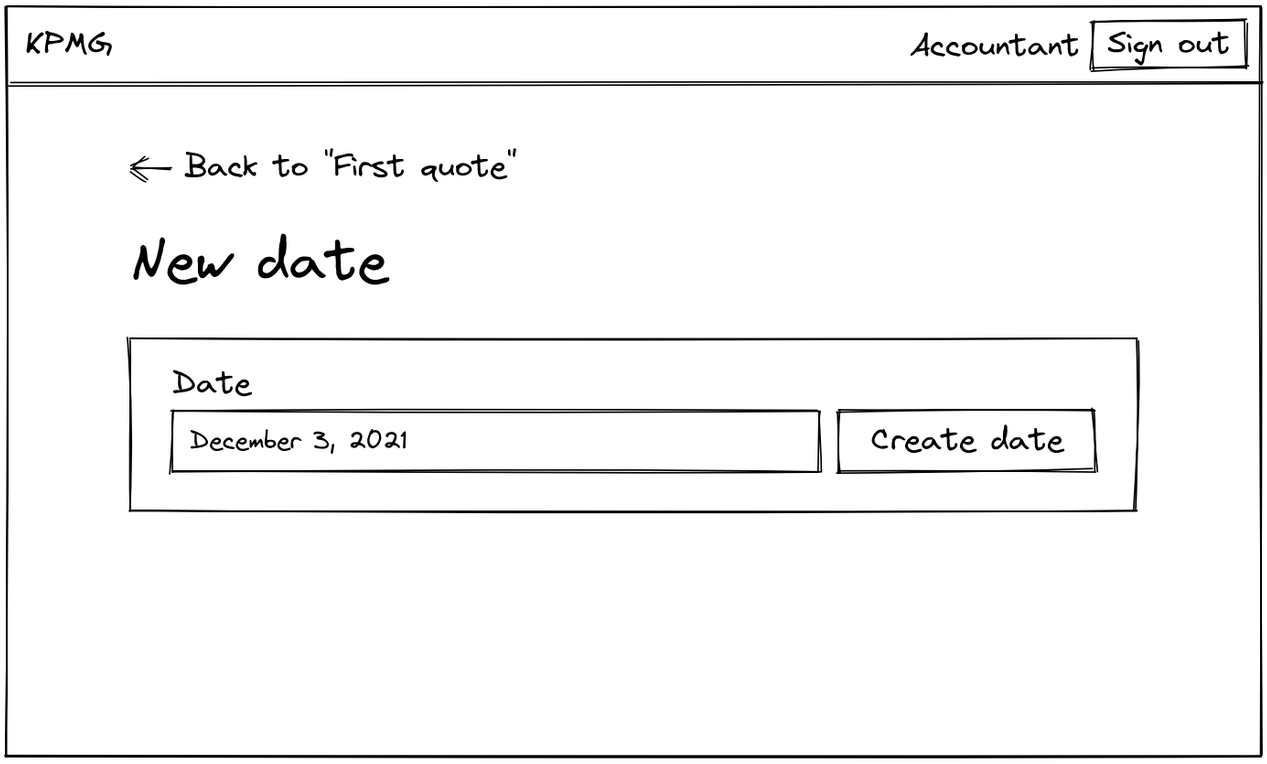
Al igual que en el primer capítulo , comenzaremos haciendo el CRUD sobre las fechas **sin usar Turbo Frames ni Turbo Streams** . Siempre comenzaremos de esta manera, ya que necesitamos que nuestros controladores funcionen correctamente antes de realizar cualquier mejora. Luego, solo se necesitarán unas pocas líneas de código para dividir la página en Turbo Frames.

Vamos a esbozar lo que vamos a construir primero. Cuando visitemos la `Quotes#show` página, deberíamos poder ver todas las fechas de la cotización:



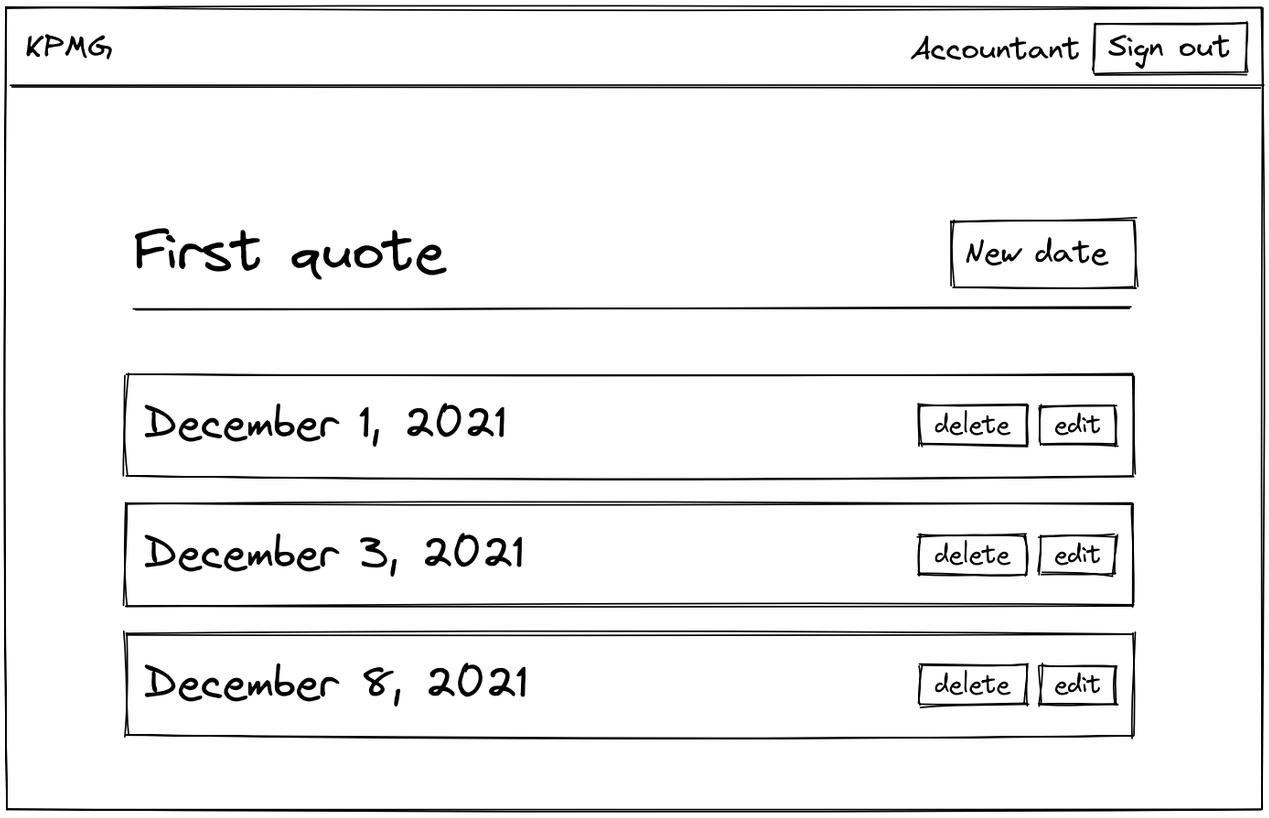
Boceto de la página de citas#mostrar con algunas fechas

Como primero estamos construyendo el CRUD **sin Turbo** , hacer clic en el enlace "Nueva fecha" nos llevará a la `LineItemDates#new` página:



Boceto de la página LineItemDates#new

Si enviamos un formulario válido, seremos redirigidos a la `Quotes#show` página con la nueva fecha agregada. Las fechas se ordenarán **en orden ascendente** :

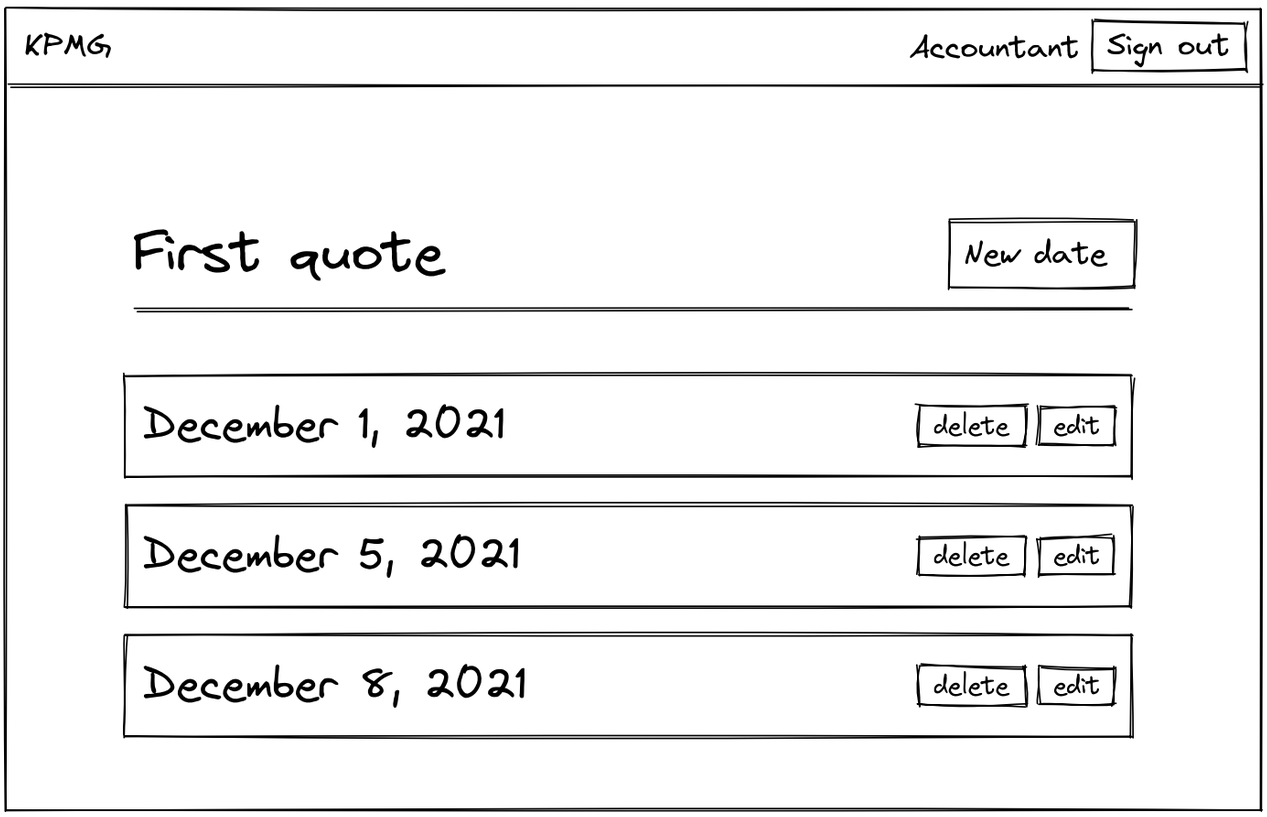


Boceto de la página de citas#show con la fecha de creación agregada a la lista

Si decidimos actualizar la fecha que acabamos de crear, podemos hacer clic en el enlace "Editar" de esta fecha para navegar a la `LineItemDates#edit` página:

Boceto de la página de edición LineItemDates#

Si enviamos un formulario válido, seremos redirigidos a la `Quotes#show` página con la fecha actualizada. Las fechas deben seguir estando **ordenadas en orden ascendente** :



Boceto de la página de citas#mostrar con la fecha actualizada

Por último, pero no por ello menos importante, podemos eliminar una fecha haciendo clic en el enlace "Eliminar" correspondiente a esa fecha. La fecha se eliminará de la lista.

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

Creando el modelo

Comencemos creando un modelo `LineItemDate` con un campo de fecha y una referencia a la cotización a la que pertenece. Agregamos esta referencia porque cada fecha de ítem de línea pertenece a una cotización y cada cotización tiene muchas fechas de ítem de línea. Generemos la migración:

```
bin/rails generate model LineItemDate quote:references date:date
```

Antes de ejecutar el `rails db:migrate` comando, debemos agregar algunas restricciones al archivo de migración:

- La fecha **debe estar presente** en cada `LineItemDate` registro. Agregaremos algunas validaciones en el modelo, pero aún así debemos agregar la `null: false` restricción para hacer cumplir la presencia de la fecha en el nivel de la base de datos incluso si, por alguna razón, se omiten las validaciones.
- Queremos evitar que una cita tenga varias veces la misma fecha. Para aplicar esto a nivel de base de datos, agregamos una restricción de unicidad para la pareja `quote_id` y `date`.
- Queremos poder ordenar las fechas de los artículos de línea por fecha ascendente. Deberíamos agregar un índice en el campo de la base de datos por razones de rendimiento cuando sabemos que lo usaremos para fines de ordenamiento.

La migración final se ve así:

```
# db/migrate/XXXXXXXXXXXXXX_create_line_item_dates.rb

class CreateLineItemDates < ActiveRecord::Migration[7.0]
  def change
    create_table :line_item_dates do |t|
      t.references :quote, null: false, foreign_key: true
      # Adding null: false constraint on date
      t.date :date, null: false

      t.timestamps
    end

    # Adding uniqueness constraint for the couple date and quote_id
    add_index :line_item_dates, [:date, :quote_id], unique: true
    # Adding index to the date field for performance reasons
```

```
    add_index :line_item_dates, :date
  end
end
```

Ahora que nuestra migración está lista, podemos ejecutarla:

```
bin/rails db:migrate
```

Agreguemos las asociaciones y las validaciones correspondientes en el `LineItemDate` modelo y un alcance para ordenar las fechas de nuestras líneas de pedido en orden ascendente:

```
# app/models/line_item_date.rb

class LineItemDate < ApplicationRecord
  belongs_to :quote

  validates :date, presence: true, uniqueness: { scope: :quote_id }

  scope :ordered, -> { order(date: :asc) }
end
```

La línea de validación aquí refuerza que:

- La fecha debe estar presente en cada línea de fecha del artículo gracias a la `presence: true` opción
- Una cotización no podrá tener la misma fecha dos veces gracias a la `uniqueness: { scope: :quote_id }` opción

Agreguemos ahora la `has_many` asociación al `Quote` modelo:

```
# app/models/quote.rb

class Quote < ApplicationRecord
  has_many :line_item_dates, dependent: :destroy

  # All the previous code...
end
```

¡Nuestra capa de modelo ya está completa! Ahora trabajaremos en las rutas.

Agregar rutas para fechas de artículos de línea

Queremos realizar las siete acciones CRUD en el `LineItemDate` modelo excepto dos de ellas:

- No necesitaremos la `LineItemDates#index` acción porque todas las fechas ya estarán presentes en la `Quotes#show` página.
- No necesitaremos la `LineItemDates#show` acción, ya que no tendría sentido ver la fecha de un solo ítem de línea. Siempre queremos ver la cotización en su totalidad.

Estas dos excepciones se reflejan en el archivo de rutas a continuación:

```
# config/routes.rb

Rails.application.routes.draw do
  # All the previous routes

  resources :quotes do
    resources :line_item_dates, except: [:index, :show]
  end
end
```

Este es un archivo de rutas muy limpio, ya que solo usamos recursos RESTful. En la siguiente sección, agregaremos algunos datos falsos a nuestros accesorios y semillas para poder diseñar nuestra `Quotes#show` página.

Diseño de fechas de partidas individuales

La `Quotes#show` página está vacía actualmente. Comencemos agregando algunos datos falsos a nuestra primera cotización en los partidos:

```
# test/fixtures/line_item_dates.yml

today:
  quote: first
  date: <%= Date.current %>

next_week:
```

```
  quote: first
  date: <%= Date.current + 1.week %>
```

Ahora podemos volver a cargar la base de datos ejecutando el `bin/rails db:seed` comando. Esas cargas nos permitirán diseñar nuestro editor de cotizaciones con datos falsos. Ahora abramos la aplicación en la `Quotes#show` página de la "Primera cotización" y comencemos a diseñar la página. Por ahora, el marcado de nuestra `Quotes#show` página se ve así:

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

<main class="container">
  <%= link_to sanitize("&larr; Back to quotes"), quotes_path %>
  <div class="header">
    <h1>
      <%= @quote.name %>
    </h1>
  </div>
</main>
```

Para que coincidan con nuestros bocetos, necesitamos agregar un enlace a la `LineItemDates#new` página y una línea para representar la colección de fechas de elementos de línea:

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

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

  <div class="header">
    <h1>
      <%= @quote.name %>
    </h1>

    <%= link_to "New date",
                new_quote_line_item_date_path(@quote),
                class: "btn btn--primary" %>
  </div>

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

Para representar esta colección de fechas de elementos de línea, primero debemos recuperar esos elementos de línea en la `QuotesController#show` acción:

```
# app/controllers/quotes_controller.rb

class QuotesController < ApplicationController
  # All the previous code...

  def show
    @line_item_dates = @quote.line_item_dates.ordered
  end

  # All the previous code...
end
```

Usamos el `ordered` alcance de nuestra colección de fechas de artículos de línea para **ordenarlas en orden ascendente** . Ahora que recuperamos correctamente nuestra colección de la base de datos, es momento de crear el HTML para una sola fecha de artículo de línea:

```
<%# app/views/line_item_dates/_line_item_date.html.erb %>

<div class="line-item-date">
  <div class="line-item-date__header">
    <h2 class="line-item-date__title">
      <%= l(line_item_date.date, format: :long) %>
    </h2>

    <div class="line-item-date__actions">
      <%= button_to "Delete",
                    [quote, line_item_date],
                    method: :delete,
                    class: "btn btn--light" %>
      <%= link_to "Edit",
                  [:edit, quote, line_item_date],
                  class: "btn btn--light" %>
    </div>
  </div>
</div>
```

La mayor parte del marcado está dentro de un div con una `.line-item-date__header` clase. Esto se debe a que tendremos un `.line-item-`

`date__body` y una `.line-item-date__footer` clase en el próximo capítulo que contendrán los elementos de línea de nuestra cotización y un enlace para crear un nuevo elemento de línea. Para minimizar la cantidad de CSS/HTML que tendremos que cambiar, aquí tomaremos un pequeño atajo.

Nota : aquí utilizo rutas polimórficas para facilitar la lectura. Quiero que las líneas de código sean un poco más cortas en este tutorial, para que no tengas que desplazarte por las secciones de código. Si no estás familiarizado con las rutas polimórficas, las dos líneas siguientes son equivalentes (pero la segunda es más larga):

```
<%= button_to "Delete", [quote, line_item_date] %>
<%= button_to "Delete", quote_line_item_date_path(quote, line_item_date)
```

También es posible utilizarlos en controladores. Por ejemplo, las dos líneas de código siguientes son equivalentes:

```
redirect_to @quote
redirect_to quote_path(@quote)
```

Si quieres saber más sobre ellos, aquí tienes un enlace a la documentación .

Ahora que tenemos el marcado HTML, agreguemos un poco de CSS para que las fechas de nuestros artículos de línea sean un poco más bonitas:

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

.line-item-date {
  margin-top: var(--space-xl);
  margin-bottom: var(--space-xxs);

  &__header {
    display: flex;
    align-items: center;
    justify-content: space-between;
    gap: var(--space-xs);
  }
```

```
  &__title {
    font-size: var(--font-size-xl);

    @include media(tabletAndUp) {
      font-size: var(--font-size-xxl);
    }
  }

  &__actions {
    display: flex;
    gap: var(--space-xs);
  }
}
```

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

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

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

¡Todo debería verse bien ahora! Podemos inspeccionar nuestro diseño en el navegador y ver que es *lo suficientemente bueno* . Ahora es momento de comenzar a trabajar en nuestro controlador.

Nuestro controlador CRUD estándar

Creación de fechas de artículos de línea sin Turbo

Ahora que el esquema, el modelo, las rutas, el marcado y el diseño de nuestra base de datos están listos, es hora de comenzar a trabajar en el controlador. Como se mencionó en la introducción, primero crearemos un controlador estándar **sin Turbo Frames ni Turbo Streams; los agregaremos más adelante** .

Nuestro controlador contendrá las siete acciones del CRUD, excepto las acciones `#index` y `#show` . Comencemos por hacer que las acciones `#new` y `#create` funcionen:

```
# app/controllers/line_item_dates_controller.rb

class LineItemDatesController < ApplicationController
  before_action :set_quote

  def new
    @line_item_date = @quote.line_item_dates.build
  end

  def create
    @line_item_date = @quote.line_item_dates.build(line_item_date_params

    if @line_item_date.save
      redirect_to quote_path(@quote), notice: "Date was successfully cre
    else
      render :new, status: :unprocessable_entity
    end
  end

  private

  def line_item_date_params
    params.require(:line_item_date).permit(:date)
  end

  def set_quote
    @quote = current_company.quotes.find(params[:quote_id])
  end
end
```

Nuestro controlador es muy estándar y ya debería funcionar, pero nos falta la `line_item_dates/new.html.erb` vista y el `line_item_dates/_form.html.erb` parcial. Agreguemos esos dos archivos a nuestra aplicación:

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

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

  <div class="header">
    <h1>New date</h1>
  </div>
```

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

No necesitamos un diseño sofisticado para nuestra `LineItemDates#new` página, ya que utilizaremos Turbo más adelante solo para extraer el formulario de la página e insertarlo en `Quotes#show` ella. Sin embargo, debería seguir siendo utilizable para las personas que utilizan navegadores antiguos que no admiten Turbo. Agreguemos el marcado para nuestro formulario:

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

<%= simple_form_for [quote, line_item_date], html: { class: "form line-i
  <% if line_item_date.errors.any? %>
    <div class="error-message">
      <%= line_item_date.errors.full_messages.to_sentence.capitalize %>
    </div>
  <% end %>

  <%= f.input :date, html5: true, input_html: { autofocus: true } %>
  <%= link_to "Cancel", quote_path(quote), class: "btn btn--light" %>
  <%= f.submit class: "btn btn--secondary" %>
<% end %>
```

Con esos dos archivos creados, vamos a probar la creación de una nueva fecha en nuestro navegador. ¡Todo funciona como se esperaba!

Refactorización del mensaje de notificación de error

Aquí tenemos la oportunidad de refactorizar la forma en que manejamos los errores dentro de nuestros formularios. Podemos observar que usamos exactamente la misma forma de mostrar los errores en el formulario de cotización y en el formulario de fecha de partida:

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

<% if quote.errors.any? %>
  <div class="error-message">
    <%= quote.errors.full_messages.to_sentence.capitalize %>
```

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

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

<% if line_item_date.errors.any? %>
  <div class="error-message">
    <%= line_item_date.errors.full_messages.to_sentence.capitalize %>
  </div>
<% end %>
```

Queremos que la forma en que mostramos los errores sea coherente en toda la aplicación. Vamos a crear un asistente que utilizaremos en todos nuestros formularios para garantizar que los errores siempre se traten de la misma manera:

```
# app/helpers/application_helper.rb

module ApplicationHelper
  # All the previous code

  def form_error_notification(object)
    if object.errors.any?
      tag.div class: "error-message" do
        object.errors.full_messages.to_sentence.capitalize
      end
    end
  end
end
```

Por ahora, nuestra aplicación solo tiene dos ayudantes, por lo que está bien mantenerlos en el `ApplicationHelper`. Sin embargo, a medida que nuestra aplicación crezca, será importante organizar esos ayudantes en unidades lógicas. ¡Podríamos tener un dedicado `FormHelper` para todo el código relacionado con el formulario! Sin embargo, esta no es nuestra preocupación aquí y simplemente queremos eliminar la duplicación. Con ese ayudante en su lugar, podemos usarlo en ambas vistas:

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

<%# All the previous code %>
```

```
<%= form_error_notification(line_item_date) %>
<%# All the previous code %>
```

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

<%# All the previous code %>
<%= form_error_notification(quote) %>
<%# All the previous code %>
```

Gracias a este asistente, nuestro formulario de fecha de artículo final se ve así:

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

<%= simple_form_for [quote, line_item_date], html: { class: "form line-i
  <%= form_error_notification(line_item_date) %>

  <%= f.input :date, html5: true, input_html: { autofocus: true } %>
  <%= link_to "Cancel", quote_path(quote), class: "btn btn--light" %>
  <%= f.submit class: "btn btn--secondary" %>
<% end %>
```

Con nuestro asistente instalado, **solo necesitamos cinco líneas de código para diseñar nuestro formulario** . Si hacemos la prueba en el navegador, deberíamos ver que todo sigue funcionando como se espera después de refactorizar nuestro código.

Tomemos unos segundos para llenar el archivo de traducciones con el texto que queremos para las etiquetas y los botones de envío:

```
# config/locales/simple_form.en.yml

en:
  simple_form:
    placeholders:
      quote:
        name: Name of your quote
    labels:
      quote:
        name: Name
      line_item_date:
        date: Date
```

```
  helpers:
    submit:
      quote:
        create: Create quote
        update: Update quote
      line_item_date:
        create: Create date
        update: Update date
```

Con ese archivo completado, el texto del botón de envío será "Fecha de creación" cuando creemos un `LineItemDate` y "Fecha de actualización" cuando actualicemos un `LineItemDate`.

Actualización de fechas de artículos de línea sin Turbo

Ahora que nuestras acciones `#new` y `#create` funcionan, hagamos el mismo trabajo para las acciones `#edit` y `#update`. Comencemos con el controlador:

```
class LineItemDatesController < ApplicationController
  before_action :set_quote
  before_action :set_line_item_date, only: [:edit, :update, :destroy]

  # All the previous code

  def edit
  end

  def update
    if @line_item_date.update(line_item_date_params)
      redirect_to quote_path(@quote), notice: "Date was successfully upd
    else
      render :edit, status: :unprocessable_entity
    end
  end

  private

  def set_line_item_date
    @line_item_date = @quote.line_item_dates.find(params[:id])
  end

  # All the previous code
end
```

Sabemos que también vamos a necesitar la `set_line_item_date` devolución de llamada para la `#destroy` acción, por lo que podemos anticiparnos y agregarla a la lista de acciones que requieren esta devolución de llamada.

Ahora que nuestras acciones `#edit` y `#update` están implementadas, agreguemos la `LineItemDates#edit` vista para poder probar en el navegador:

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

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

  <div class="header">
    <h1>Edit date</h1>
  </div>
```

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

Como podemos observar, la `LineItemDates#edit` vista es muy similar a la `LineItemDates#new` vista anterior. Solo cambia el título. Como ya construimos el formulario en la sección anterior, estamos listos para experimentar en el navegador. Todo funciona como se esperaba, ¡solo falta una acción más!

Eliminar fechas de artículos de línea sin Turbo

La `#destroy` acción es la más sencilla de las cinco, ya que no requiere una vista. Solo tenemos que eliminar la fecha de la línea de pedido y luego redirigir a la `Quotes#show` página:

```
class LineItemDatesController < ApplicationController
  # All the previous code

  def destroy
    @line_item_date.destroy

    redirect_to quote_path(@quote), notice: "Date was successfully destr
  end

  # All the previous code
end
```

Probémoslo en nuestro navegador y funciona como se esperaba. Podríamos hacer una cosa más para evitar que los usuarios eliminen fechas sin querer. Agreguemos un mensaje de confirmación cuando hagan clic en el botón "Eliminar" en la fecha de un elemento de línea. Para confirmar los envíos de formularios con Turbo, debemos agregar el `data-turbo-confirm="Your message"` a la `<form>` etiqueta HTML.

Hagamos esto en los botones "Eliminar" para las fechas de los artículos de línea:

```
<%# app/views/line_item_dates/_line_item_date.html.erb %>

<!-- All the previous code -->

<%= button_to "Delete",
```

```
              quote_line_item_date_path(quote, line_item_date),
              method: :delete,
              form: { data: { turbo_confirm: "Are you sure?" } },
              class: "btn btn--light" %>

<!-- All the previous code -->
```

El `button_to` asistente genera un formulario en HTML. Así es como debería verse el HTML si inspeccionamos el DOM:

```
<form data-turbo-confirm="Are you sure?" class="button_to" method="post"
  <input type="hidden" name="_method" value="delete" autocomplete="off">
  <button class="btn btn--light" type="submit">Delete</button>
  <input type="hidden" name="authenticity_token" value="long_token" auto
</form>
```

Lo importante aquí es observar que el `data-turbo-confirm` atributo de datos está en la `<form>` etiqueta. Cuando hacemos clic en el botón "Eliminar" para la fecha de un artículo de línea, ¡ahora aparece una alerta de confirmación en la pantalla!

Nuestro controlador CRUD ahora funciona como se esperaba, pero ahora queremos que todas las interacciones se realicen en la misma página. Gracias a la potencia de Turbo, solo se necesitarán unas pocas líneas de código para dividir nuestra página en partes que se puedan actualizar de forma independiente.

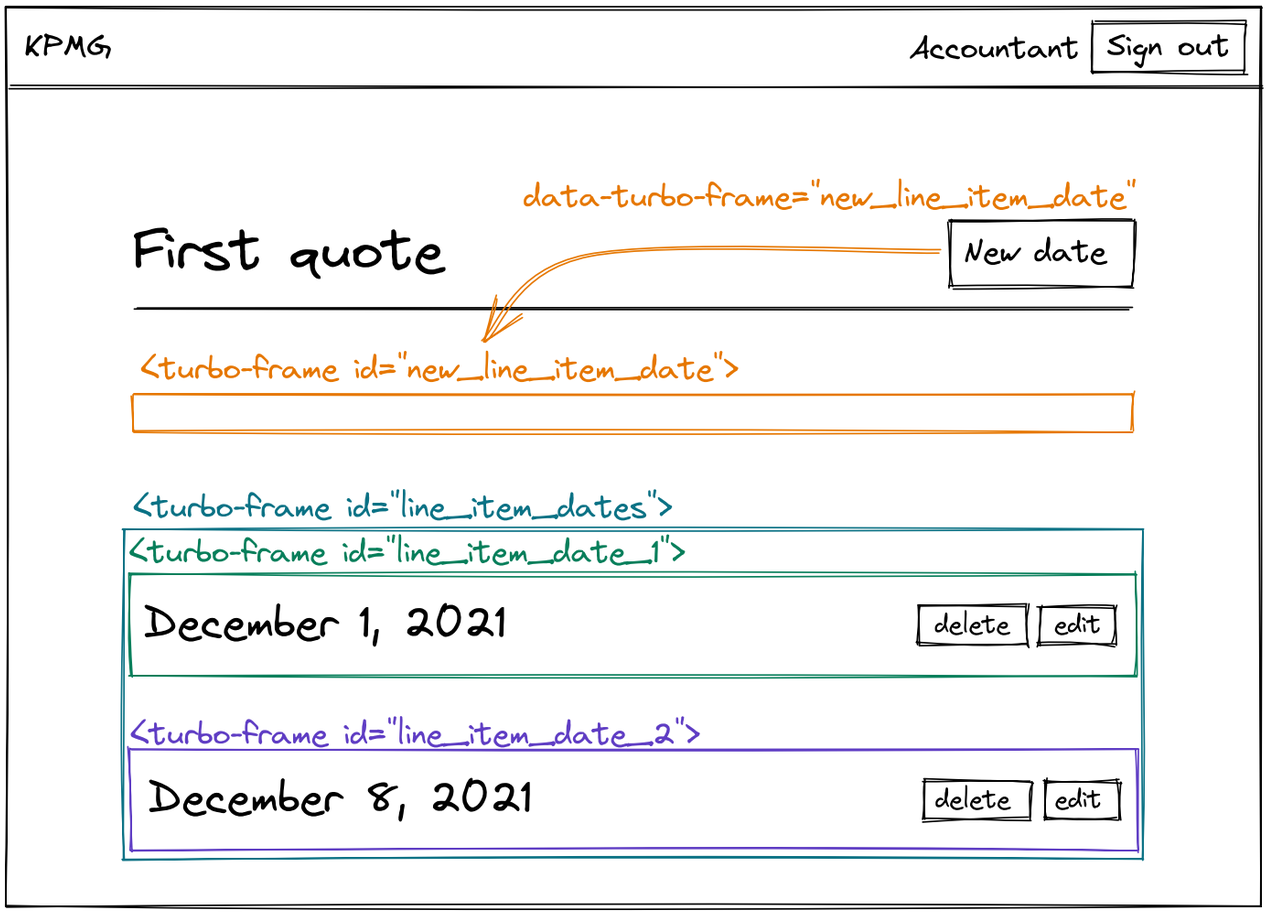
Cómo agregar Turbo Frames y Turbo Streams

Ahora que nuestro controlador CRUD funciona como se esperaba, es momento de mejorar la experiencia del usuario para que todas las interacciones se realicen en `Quotes#show` la página. La experiencia será muy similar a la que teníamos en la `Quotes#index` página.

Creación de fechas de artículos de línea con Turbo

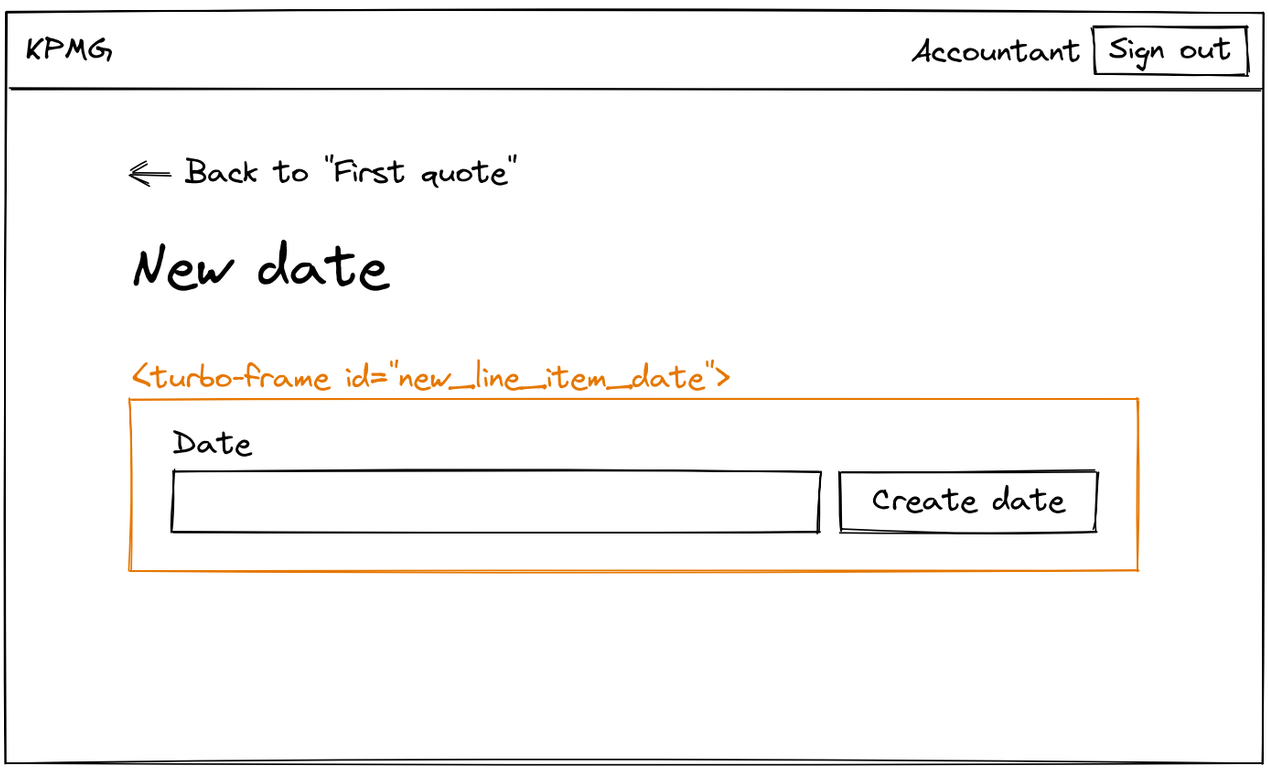
Para tener claros los requisitos, primero esbocemos el comportamiento deseado. Lo que queremos es que cuando un usuario visite la `Quotes#show` página y haga clic en el botón "Nueva fecha", el formulario

aparezca en la `Quotes#show` página. ¡Haremos esto con Turbo Frames, por supuesto! Para que funcione, tenemos que conectar el enlace "Nueva fecha" a un Turbo Frame vacío gracias al `data-turbo-frame` atributo data:



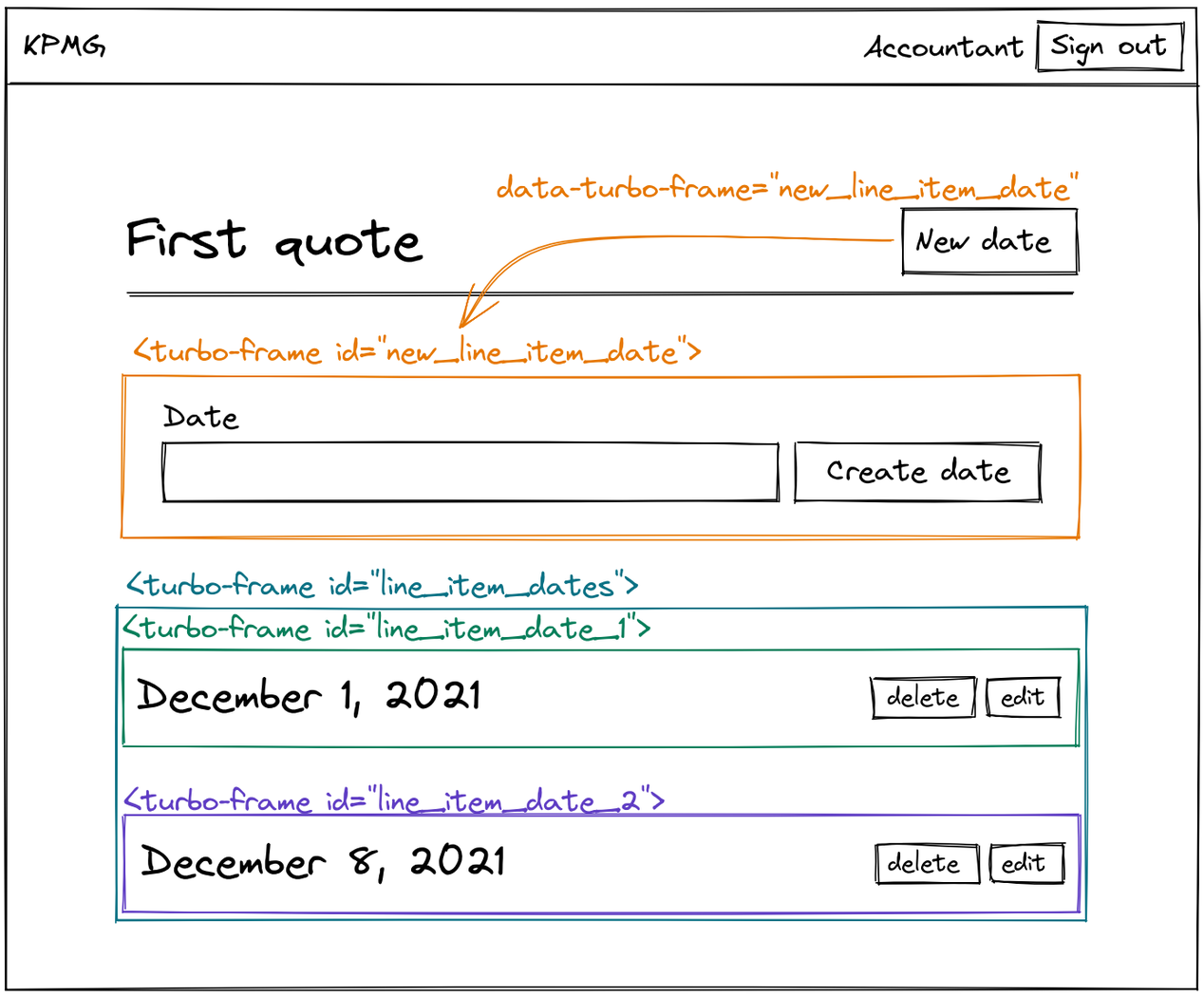
Boceto de la página de citas#show con Turbo Frames

Para que Turbo reemplace correctamente el Turbo Frame en la `Quotes#show` página, el Turbo Frame en la `LineItemDates#new` página debe tener el mismo id. Por convención, este id es `new_line_item_date` el de `dom_id` una nueva instancia de nuestro `LineItemDate` modelo:



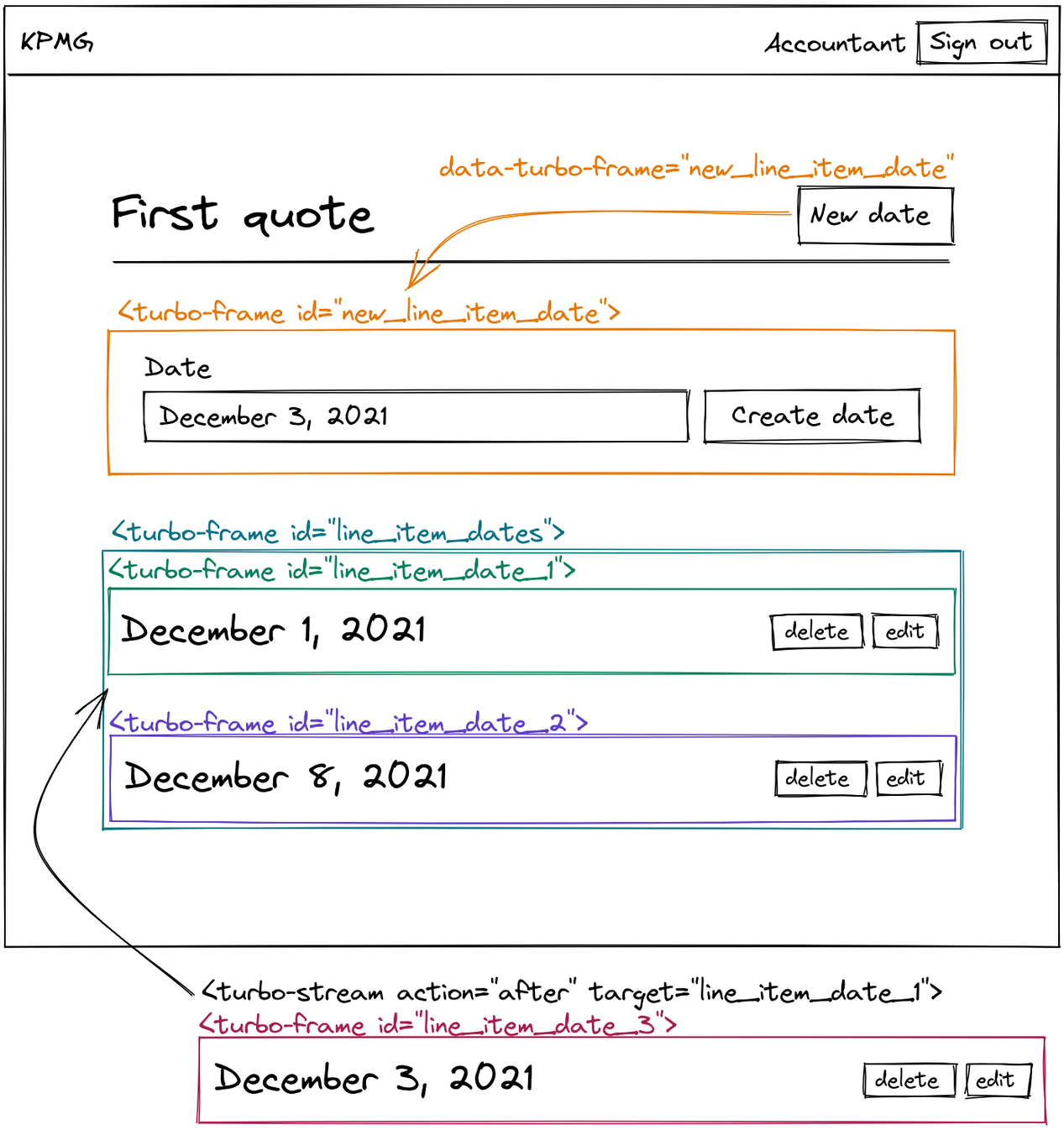
Boceto de la página LineItemDates#new con un Turbo Frame

Con esos Turbo Frames en su lugar, cuando un usuario hace clic en el botón "Nueva fecha", Turbo reemplazará exitosamente el Turbo Frame vacío en la `Quotes#show` página con el Turbo Frame que contiene el formulario en la `LineItemDates#new` página.



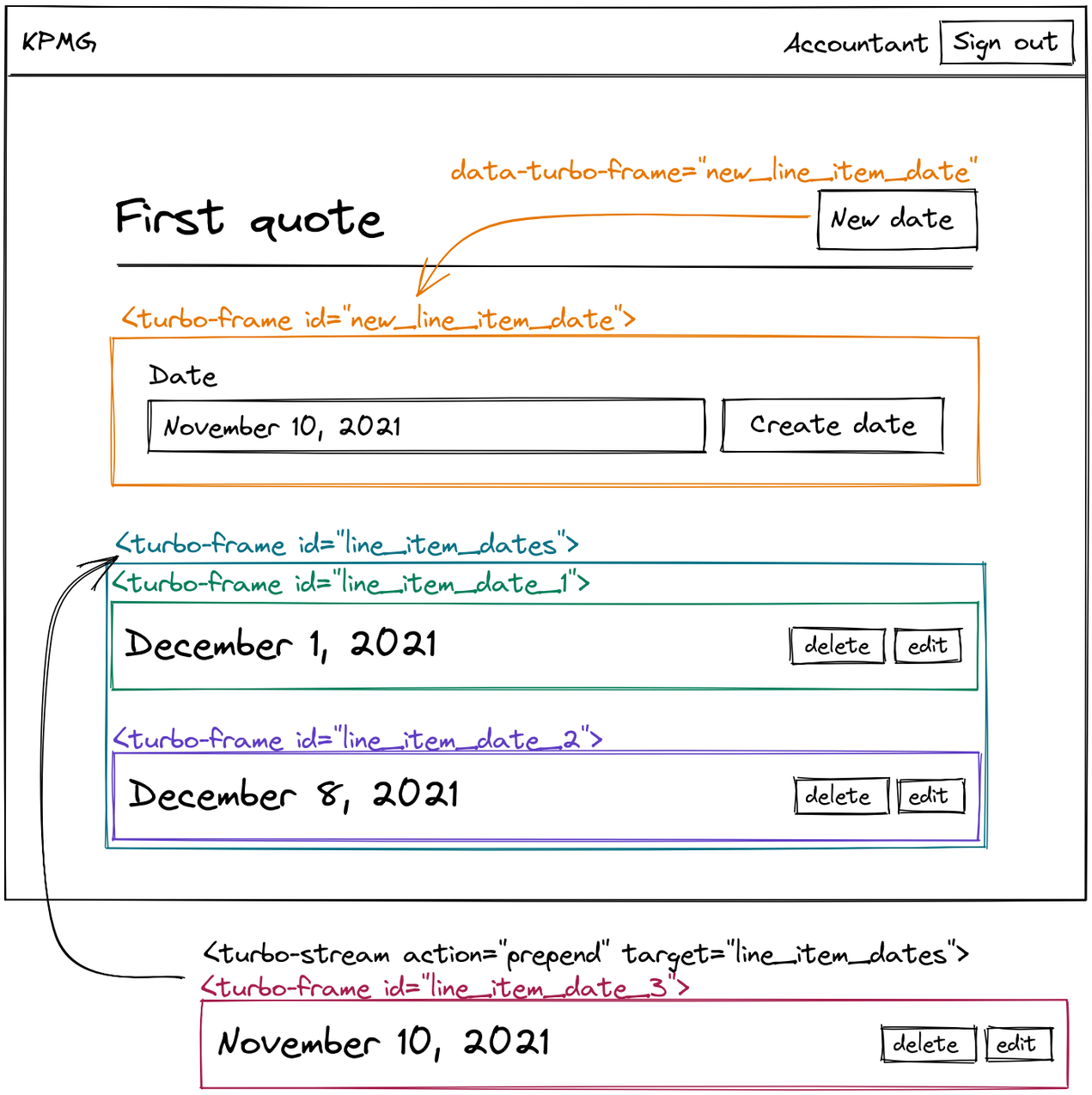
Boceto de la página Quotes#show con el formulario de la página LineItemDates#new

Cuando el usuario envía el formulario, queremos que la fecha de creación de la línea de pedido se inserte en la `Quotes#show` página en la posición correcta para que las fechas permanezcan en **orden ascendente** . Si una cotización ya tiene una fecha anterior a la que acabamos de crear, debemos insertar el HTML de la fecha de creación en el orden correcto, como se describe en el esquema siguiente:



Boceto de la fecha de creación insertada justo después de la fecha anterior

Por otro lado, si la cita no tiene fecha anterior a la que acaba de ser creada, se debe insertar el HTML de la fecha de creación al principio de la lista:



Boceto de la fecha de creación insertada al principio de la lista

Ahora que los requisitos están claros, solo deberían hacer falta unas pocas líneas de código para hacerlo realidad, ¡gracias al poder de Turbo Frames y Turbo Streams!

Comencemos a trabajar en la primera parte: hacer que el formulario aparezca en la `Quotes#show` página cuando un usuario haga clic en el botón "Nueva fecha". Este es el mismo trabajo que en la `Quotes#index` página. En la `Quotes#show` página, agreguemos un Turbo Frame vacío con el id de `new_line_item_date` y vinculemos el botón "Nueva fecha" a él:

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

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

  <div class="header">
```

```
    <h1>
      <%= @quote.name %>
    </h1>

    <%= link_to "New date",
                new_quote_line_item_date_path(@quote),
                data: { turbo_frame: dom_id(LineItemDate.new) },
                class: "btn btn--primary" %>
  </div>

  <%= turbo_frame_tag LineItemDate.new %>
  <%= render @line_item_dates, quote: @quote %>
</main>
```

Ahora que tenemos un `turbo_fram_tag` con un `id` de `new_line_item_date` en la `Quotes#show` página, necesitamos tener un `turbo_frame_tag` con el mismo `id` en la `LineItemDates#new` página para que Turbo pueda intercambiar los dos marcos. Envolvamos nuestro formulario en una etiqueta Turbo Frame:

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

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

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

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

¡Ahora, al hacer clic en el botón “Nueva fecha”, el formulario aparece en la `Quotes#show` página!

Cuando enviamos un formulario **no válido** , los errores aparecen en la página como se espera. Esto se debe a que nuestro `LineItemDates` controlador muestra la `LineItemDates#new` vista en un envío no válido. Como el formulario está dentro de un Turbo Frame de id `new_line_item_date` y la vista mostrada contiene un Turbo Frame con el mismo id, Turbo es lo suficientemente inteligente como para reemplazar el frame automáticamente.

Como se explicó en capítulos anteriores, las respuestas al envío de formularios no válidos deben tener el `unprocessable_entity` estado para que Turbo muestre errores.

Sin embargo, cuando enviamos un formulario **válido** , como el formulario está dentro de un Turbo Frame de id `new_line_item_date` y la respuesta redirige a la `Quotes#show` página que contiene un frame vacío con este id, el formulario desaparece ya que el Turbo Frame que contiene el formulario es reemplazado por uno vacío. No sucede nada con la fecha de creación porque Turbo no puede adivinar qué debemos hacer con ella. Para satisfacer el comportamiento que esbozamos anteriormente, necesitamos una `create.turbo_stream.erb` plantilla para indicarle a Turbo que:

1. Reemplace el Turbo Frame con id `new_line_item_date` por uno vacío
2. Agregue la fecha de la línea de artículo creada a la lista **en la posición correcta**

Si bien el paso 1 es exactamente el mismo que hicimos para la `Quotes#index` página, el paso 2 es más complicado. De hecho, para insertar la fecha de la línea de pedido recién creada en la posición correcta, tenemos que identificar la posición correcta de la fecha de la nueva línea de pedido en la lista ordenada de fechas. Aprendamos cómo hacerlo. Primero, editemos nuestra `LineItemDatesController#create` acción para que responda al `turbo_stream` formato:

```
# app/controllers/line_item_dates_controller.rb

class LineItemDatesController < ApplicationController
  # All the previous code...

  def create
    @line_item_date = @quote.line_item_dates.build(line_item_date_params

    if @line_item_date.save
      respond_to do |format|
        format.html { redirect_to quote_path(@quote), notice: "Date was
        format.turbo_stream { flash.now[:notice] = "Date was successfull
      end
    else
      render :new, status: :unprocessable_entity
    end
```

```
  end

  # All the previous code...
end
```

Ahora pensemos en lo que queremos lograr. Queremos insertar la fecha de la línea de pedido recién creada en la posición correcta en la lista ordenada. El orden de las fechas es **ascendente** . Esto significa que si nuestra cotización tiene fechas anteriores a la nueva fecha, debemos agregar nuestra nueva fecha justo *después de* la última de ellas. De lo contrario, debemos *anteponer* la fecha justo al comienzo de la lista de fechas.

Codifiquemos esto en nuestra vista Turbo Stream:

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

<%# Step 1: remove the form from the Quotes#index page %>
<%= turbo_stream.update LineItemDate.new, "" %>

<%# Step 2: add the date at the right place %>
<% if previous_date = @quote.line_item_dates.ordered.where("date < ?", @
  <%= turbo_stream.after previous_date do %>
    <%= render @line_item_date, quote: @quote %>
  <% end %>
<% else %>
  <%= turbo_stream.prepend "line_item_dates" do %>
    <%= render @line_item_date, quote: @quote %>
  <% end %>
<% end %>

<%= render_turbo_stream_flash_messages %>
```

Este fragmento de código se podría mejorar y lo refactorizaremos pronto, pero primero entendamos qué hace. El paso 1 es el mismo que para las comillas. Simplemente vaciamos el Turbo Frame que contiene el formulario.

El paso 2 es un poco más complicado. Primero recuperamos la fecha de la línea de pedido que es inmediatamente anterior a la que acabamos de crear. Si existe, agregamos la nueva fecha de la línea de pedido justo *después de* esta. Si no, *anteponemos* la nueva fecha de la línea de pedido a la lista de fechas de las líneas de pedido.

Para que funcione, tenemos que envolver las fechas de nuestros artículos de línea en un `turbo_frame_tag` if `id` en `line_item_dates` caso de que necesitemos *anteponer* la fecha del artículo de línea creado a la lista:

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

<%# All the previous code... %>

<%= turbo_frame_tag "line_item_dates" do %>
  <%= render @line_item_dates, quote: @quote %>
<% end %>

<%# All the previous code... %>
```

También debemos incluir la fecha de cada elemento de línea individual en un `turbo_frame_tag` . Esto se debe a que debemos poder identificar la fecha de cada elemento de línea mediante un identificador único en caso de que necesitemos insertar la fecha de creación justo *después de* uno de ellos:

```
<%# app/views/line_item_dates/_line_item_date.html.erb %>

<%= turbo_frame_tag line_item_date do %>
  <div class="line-item-date">
    <!-- All the previous code -->
  </div>
<% end %>
```

Probémoslo en nuestro navegador y debería funcionar. Ahora, refactoricemos nuestro código y agreguemos algunas pruebas. Lo primero que debemos hacer es extraer la lógica *de fecha anterior* al `LineItemDate` modelo:

```
# app/models/line_item_date.rb

class LineItemDate < ApplicationRecord
  # All the previous code...

  def previous_date
    quote.line_item_dates.ordered.where("date < ?", date).last
  end
end
```

Ahora reemplacemos la lógica en nuestra vista con el método que acabamos de crear:

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

<%# Step 1: remove the form from the Quotes#index page %>
<%= turbo_stream.update LineItemDate.new, "" %>

<%# Step 2: add the date at the right place %>
<% if previous_date = @line_item_date.previous_date %>
  <%= turbo_stream.after previous_date do %>
    <%= render @line_item_date, quote: @quote %>
  <% end %>
<% else %>
  <%= turbo_stream.prepend "line_item_dates" do %>
    <%= render @line_item_date, quote: @quote %>
  <% end %>
<% end %>

<%= render_turbo_stream_flash_messages %>
```

También probemos este método para asegurarnos de que funciona como se espera:

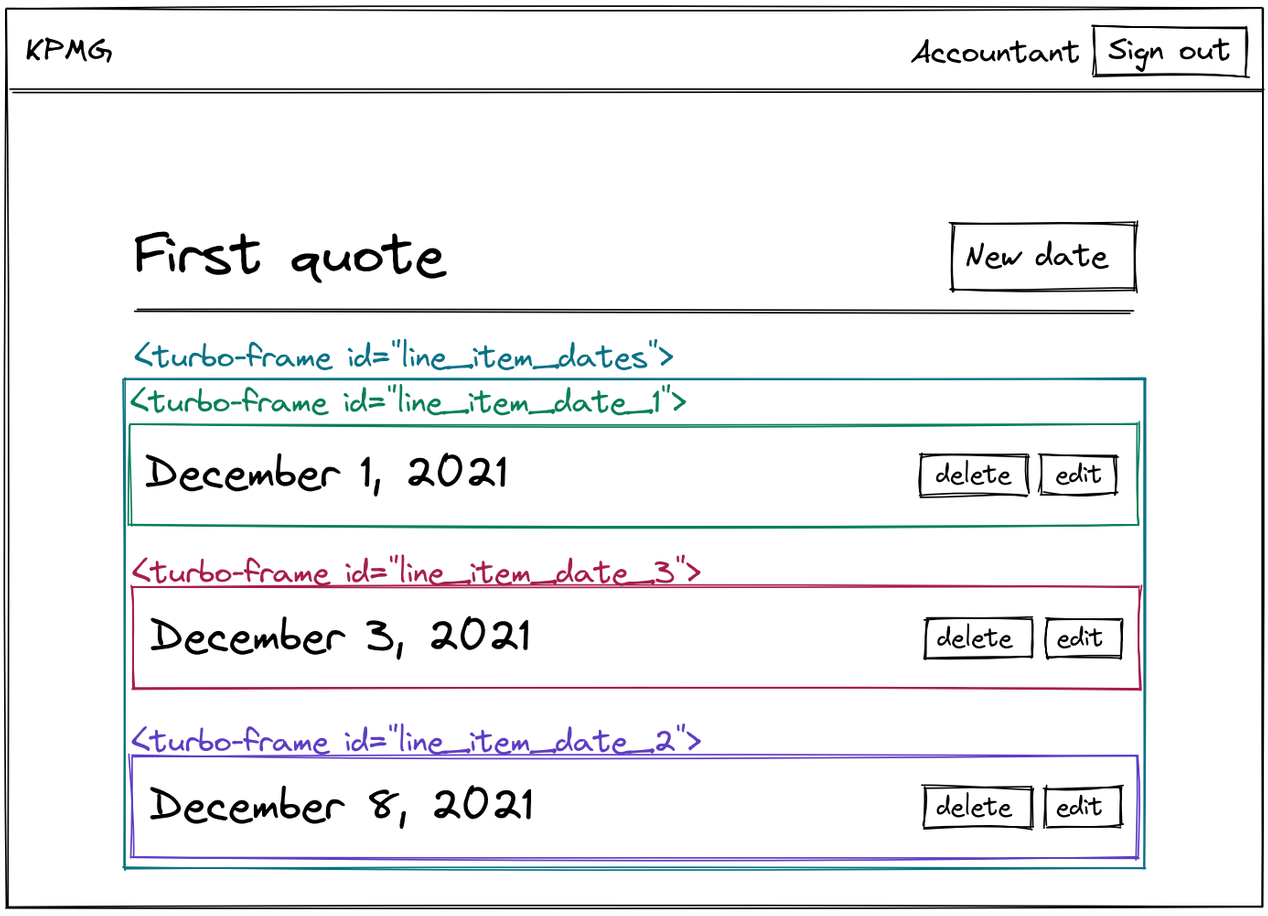
```
# test/models/line_item_date_test.rb

require "test_helper"

class LineItemDateTest < ActiveSupport::TestCase
  test "#previous_date returns the quote's previous date when it exitsts
    assert_equal line_item_dates(:today), line_item_dates(:next_week).pr
  end

  test "#previous_date returns nil when the quote has no previous date"
    assert_nil line_item_dates(:today).previous_date
  end
end
```

Esta es una prueba mínima pero nos da confianza de que nuestra aplicación está funcionando como se espera.

¡Eso fue mucho trabajo! Ya casi estamos listos, y las `#edit` acciones `#update` serán `#destroy` más fáciles de implementar ahora que casi todo está en su lugar.

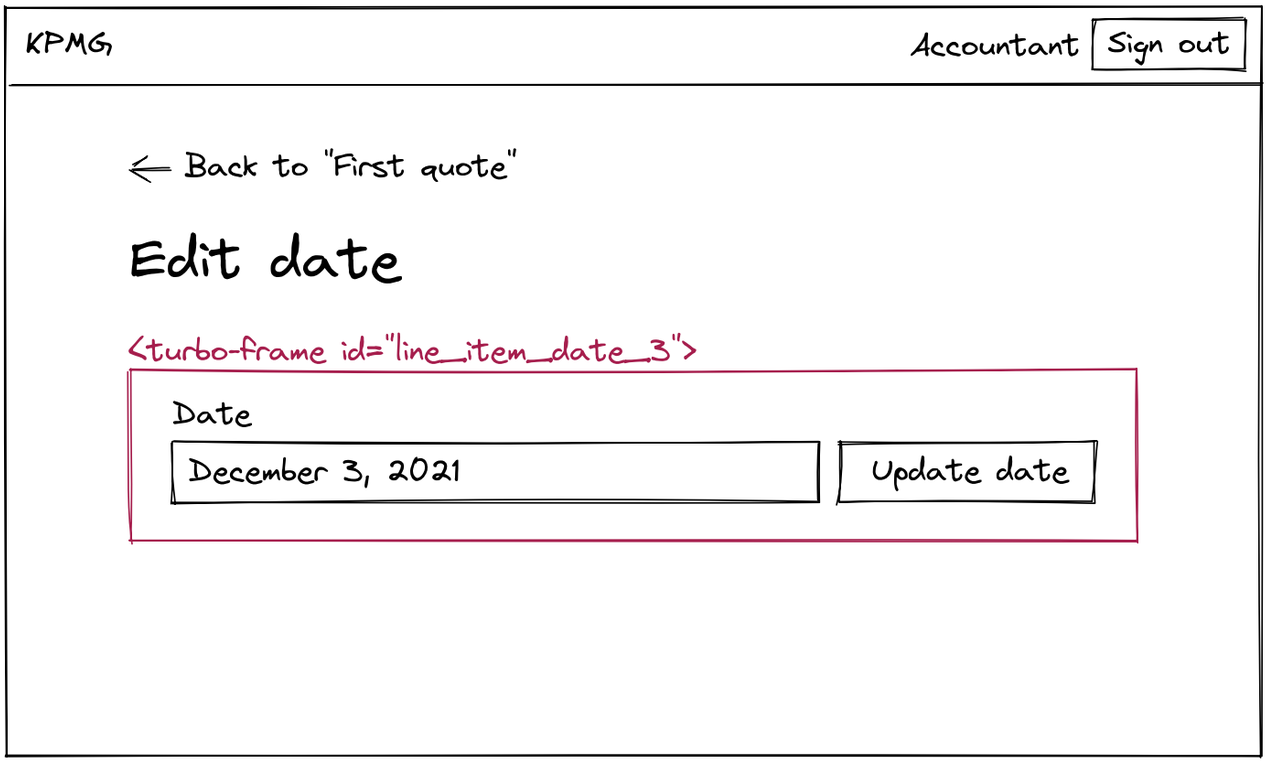
Actualización de fechas de artículos de línea con Turbo

Al igual que hicimos con las acciones `#new` y `#create` , queremos que las acciones `#edit` y `#update` de una cita se realicen en la `Quotes#show` página. Ya tenemos todos los Turbo Frames que necesitamos en la `Quotes#show` página, como se describe en el siguiente esquema:



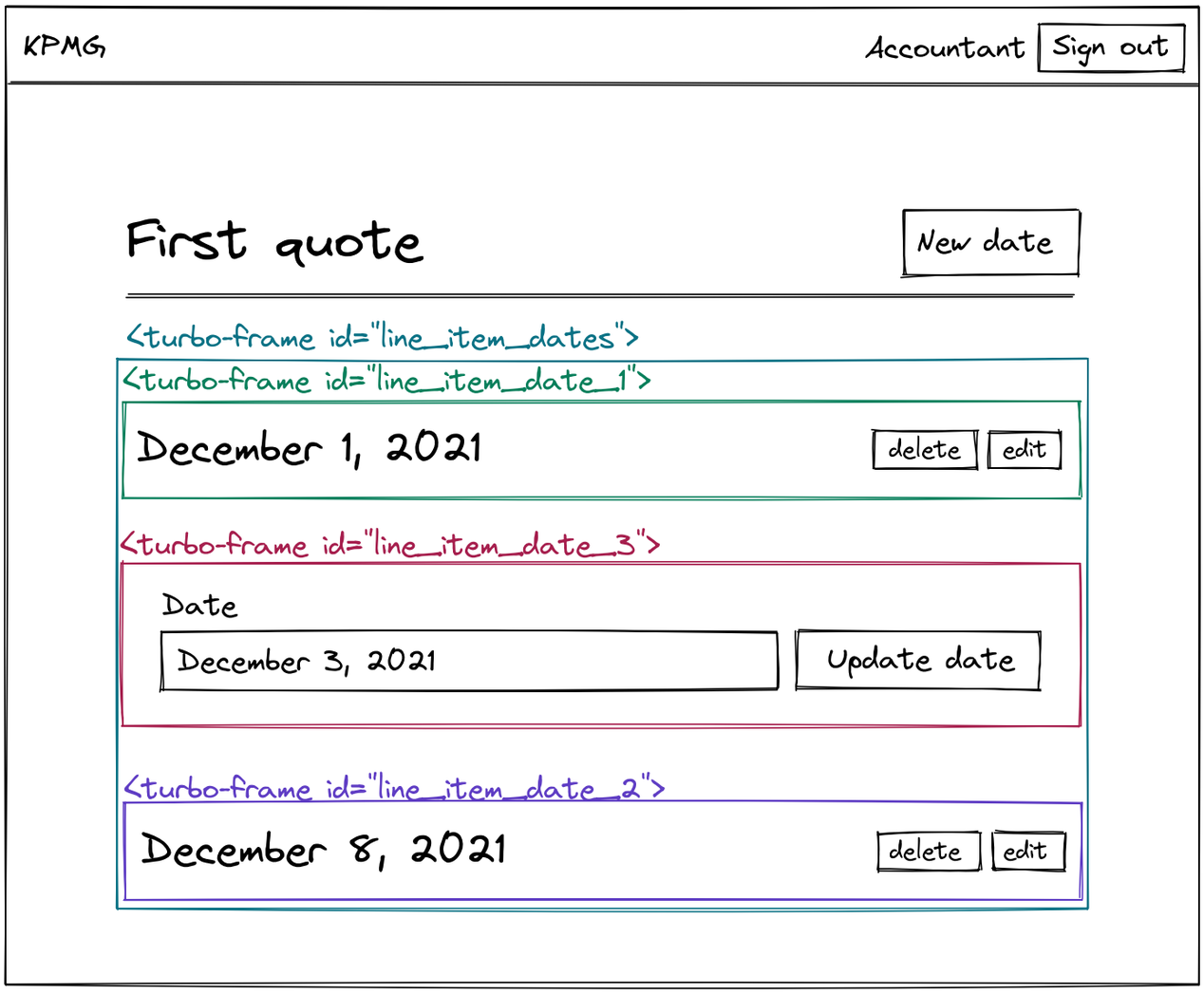
Boceto de las citas#mostrar página

Al hacer clic en el enlace "Editar" para la **segunda fecha** de nuestro boceto que está dentro de un Turbo Frame de id `line_item_date_3` , Turbo espera encontrar un Turbo Frame con el mismo id en la `LineItemDates#edit` página como se describe en el boceto a continuación:



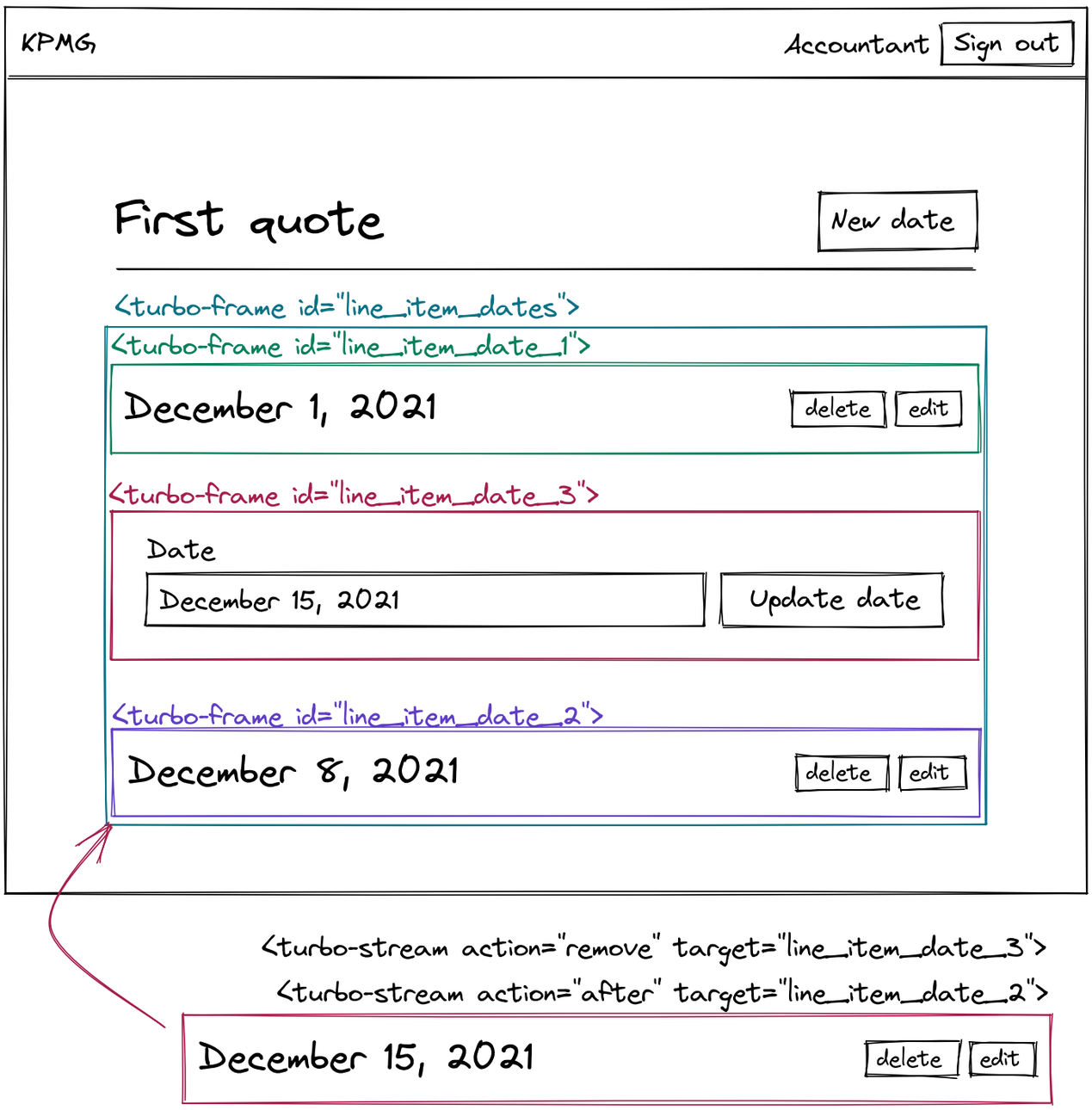
Boceto de la página LineItemDates#edit con un Turbo Frame con la misma identificación

Con esos Turbo Frames instalados, Turbo podrá reemplazar la fecha del elemento de línea en la `Quotes#show` página con el formulario de la `LineItemDates#edit` página al hacer clic en el enlace "Editar" de la fecha de un elemento de línea:



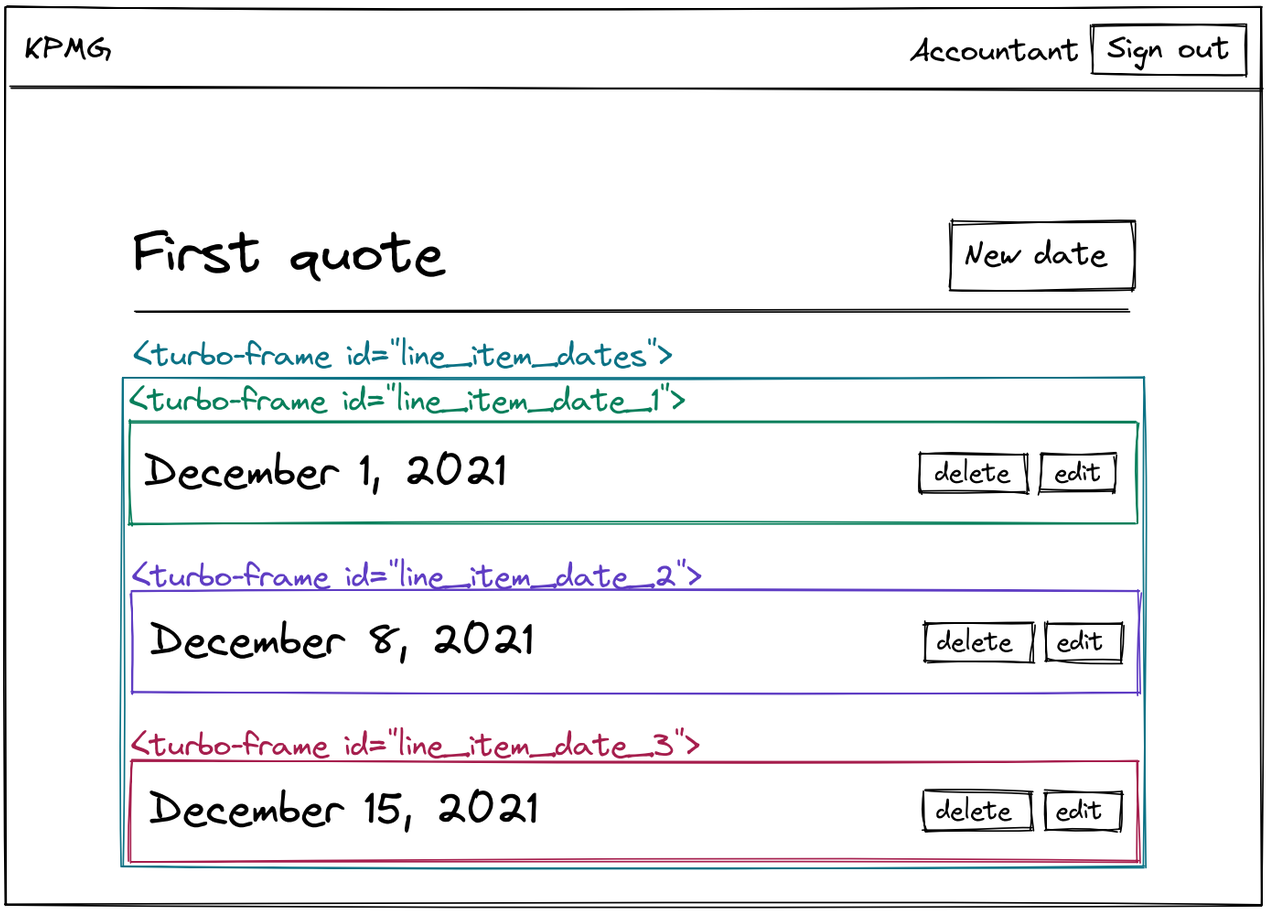
Boceto de las Citas#mostrar con la forma de edición

Al enviar el formulario, volveremos a tener el mismo problema que tuvimos para las acciones `#new` y `#create`. Necesitamos insertar la fecha de la línea de artículo actualizada en la posición correcta de la lista. Usaremos el mismo método que usamos en la `create.turbo_stream.erb` vista. Al enviar el formulario, primero eliminaremos la fecha que estamos actualizando actualmente de la vista y luego la insertaremos en el lugar correcto, tal como lo hicimos para la `#create` acción. Esto se describe en el siguiente esquema:



Boceto de la página de Cotizaciones#mostrar al enviar el formulario

El estado final de la `Quotes#show` página debe tener las fechas ordenadas en orden ascendente:



Boceto de la página de citas#mostrar con las fechas en el orden correcto

Ahora que los requisitos están claros, es hora de comenzar a codificar. La primera parte del trabajo es lograr que el formulario de edición reemplace correctamente el HTML de una fecha en la `Quotes#show` página. Para ello, solo tenemos que envolver el formulario en la `LineItemDate#edit` página dentro de un Turbo Frame:

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

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

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

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

Con este Turbo Frame instalado, podemos hacer pruebas en el navegador. Al hacer clic en el botón "Editar" en la fecha de un elemento de línea, el formulario reemplaza correctamente la fecha en la `Quotes#show` página.

Si enviamos un formulario no válido, todo funciona como se espera. Si enviamos un formulario válido, la fecha del elemento de línea se reemplaza correctamente, pero no garantiza que nuestras fechas estén en el orden correcto. Para garantizar que las fechas estén siempre en el orden correcto, debemos crear una vista de Turbo Stream que sea muy similar a la que creamos para nuestra `#create` acción en la sección anterior. Primero, habilitemos nuestro controlador para que represente una vista de Turbo Stream:

```
# app/controllers/line_item_dates_controller.rb

def update
  if @line_item_date.update(line_item_date_params)
    respond_to do |format|
      format.html { redirect_to quote_path(@quote), notice: "Date was su
      format.turbo_stream { flash.now[:notice] = "Date was successfully
    end
  else
    render :edit, status: :unprocessable_entity
  end
end
```

Ahora vamos a crear la `update.turbo_stream.erb` vista. Usaremos exactamente la misma lógica que para la `#create` acción, excepto que esta vez no vaciaremos el Turbo Frame que contiene el formulario, sino que lo eliminaremos por completo. Una vez eliminado, podemos volver a agregarlo en la posición correcta, tal como se describe en nuestros bocetos:

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

<%# Step 1: remove the form %>
<%= turbo_stream.remove @line_item_date %>

<%# Step 2: insert the updated date at the correct position %>
<% if previous_date = @line_item_date.previous_date %>
  <%= turbo_stream.after previous_date do %>
    <%= render @line_item_date, quote: @quote %>
  <% end %>
<% else %>
  <%= turbo_stream.prepend "line_item_dates" do %>
    <%= render @line_item_date, quote: @quote %>
  <% end %>
<% end %>
```

```
<%= render_turbo_stream_flash_messages %>
```

Probémoslo en el navegador. ¡Todo debería funcionar como se espera! La última acción que tenemos que hacer es la más fácil, así que sigamos.

Destrucción de fechas de artículos de línea con Turbo

La última característica que necesitamos es la capacidad de eliminar las fechas de partidas de nuestra cotización. Para ello, primero tenemos que admitir el formato Turbo Stream en la `#destroy` acción del controlador:

```
# app/controllers/line_item_dates_controller.rb

def destroy
  @line_item_date.destroy
```



```
    format.turbo_stream { flash.now[:notice] = "Date was successfully de
  end
end
```

En la vista, simplemente tenemos que eliminar la fecha del elemento de línea y mostrar el mensaje flash:

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

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

Por fin podemos probar en nuestro navegador que todo funciona como se espera. ¡El comportamiento es casi exactamente el mismo que el que teníamos para las comillas!

Probando nuestro código con pruebas del sistema

Nuestro trabajo no estaría realmente completo si no agregáramos algunas pruebas. Siempre deberíamos escribir al menos pruebas del sistema para asegurarnos de que se cubra el camino correcto. De esa manera, si cometemos un error, podemos corregirlo antes de enviar nuestro código a producción.

Agreguemos un archivo de prueba del sistema para probar la ruta correcta del CRUD en las fechas de nuestras líneas de pedido:

```
# test/system/line_item_dates_test.rb

require "application_system_test_case"

class LineItemDatesTest < ApplicationSystemTestCase
  setup do
    login_as users(:accountant)

    @quote          = quotes(:first)
    @line_item_date = line_item_dates(:today)

    visit quote_path(@quote)
  end

  test "Creating a new line item date" do
    assert_selector "h1", text: "First quote"

    click_on "New date"
    assert_selector "h1", text: "First quote"
    fill_in "Date", with: Date.current + 1.day

    click_on "Create date"
    assert_text I18n.l(Date.current + 1.day, format: :long)
  end

  test "Updating a line item date" do
    assert_selector "h1", text: "First quote"

    within id: dom_id(@line_item_date) do
      click_on "Edit"
    end

    assert_selector "h1", text: "First quote"

    fill_in "Date", with: Date.current + 1.day
    click_on "Update date"
```

```
    assert_text I18n.l(Date.current + 1.day, format: :long)
  end

  test "Destroying a line item date" do
    assert_text I18n.l(Date.current, format: :long)

    accept_confirm do
      within id: dom_id(@line_item_date) do
        click_on "Delete"
      end
    end

    assert_no_text I18n.l(Date.current, format: :long)
  end
end
```

Acabamos de agregar pruebas para la ruta de creación, actualización y eliminación de una fecha de ítem de línea. Podemos ejecutar pruebas unitarias y pruebas del sistema simultáneamente con el `bin/rails test:all` comando. ¡Ejecutémoslas y deberían estar todas en verde!

Envolver

En este capítulo, hicimos otro ejemplo de un controlador CRUD sobre un recurso, tal como lo hicimos para las citas, la única diferencia es que esta vez, tuvimos que mantener el orden correcto de las fechas en la `Quotes#show` página. Para hacer esto, tuvimos que aprender a insertar parciales en una posición precisa en el DOM gracias a `turbo_stream.after`.

También aprendimos a enviar una alerta de confirmación a los usuarios gracias al `data-turbo-confirm` atributo de datos que debe colocarse en `<form>` las etiquetas.

El próximo capítulo será el último gran capítulo y hablaremos sobre los Turbo Frames anidados. ¡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