1. Widoki i partiale

Tworzymy prosty widok pokazujący wiadomość.

Mamy już akcję `show` i widok `show` dla wiadomości.
Dodajmy teraz widok komentarzy do danej wiadomości.

```
<!-- views/messages/_comments.html.erb -->

<h3>Comments</h3>
<%= comments.each do |comment| %>
  <p>
    <strong>Content:</strong>
    <%= comment.content %>
  </p>
<% end %>
```

Wywołanie partiala w widoku głównym

```
<!-- views/message/show.html.erb -->

<%= render partial: :comments, locals: { comments: @message.comments } %>
```

Uproszczona wersja (render kolekcji)

```
<!-- views/messages/show.html.erb -->
<h3>Comments</h3>
<%= render partial: 'comments', collection: @message.comments, as: :comment %>


<!-- views/messages/_comments.html.erb -->
<p>
  <strong>Content:</strong>
  <%= comment.content %>
</p>
```

2. Formularz dla obiektu Message

Definiujemy akcję wyświetlającą formularz

```
<!-- views/messages/index.html.erb -->
def new
  @message = Message.new
end
```

Tworzymy nowy obiekt Message (bez danych). Dzięki temu `Form Builder` będzie wiedział jakie atrybuty powinny znaleźć się w formularzu.

Definiujemy formularz

```
<!-- views/messages/new.html.erb -->
<%= form_for(@message) do |form| %>
  <div class="field">
    <%= form.label :content %>
    <%= form.text_field :content %>
  </div>

  <div class="field">
    <%= form.label :author %>
    <%= form.text_field :author %>
  </div>

  <div class="actions">
    <%= form.submit "Submit" %>
  </div>
<% end %>
```

Dodajemy też przycisk do tworzenia nowych wiadomości

```
<!-- views/messages/index.html.erb -->
<%= link_to 'New message', new_message_path %>
```

Definiujemy akcję create, która obsłuży formularz

```
# controllers/messages_controller.rb
def create
  @message = Message.new(message_params)
  if @message.save
    redirect_to @message
  else
    render :new
  end
end
```

Jeśli uda nam się zapisać wiadomość to przekierowujemy do akcji, która ją wyświetli.
Jeśli nie uda się zapisać wiadomości to renderujemy formularz jeszcze raz (możemy dodać tu błędy).

Metoda `render` jedynie wywołuje odpowiedni widok. Jeśli nie przekażemy do tego widoku wymaganych zmiennych, to napotkamy na błędy.

Definiujemy metodę filtrującą parametry formularza

```
# controllers/messages_controller.rb
private

def message_params
  params.require(:message).permit(:content, :author)
end
```

Dodajemy błędy do formularza

```
<% if @message.errors.any? %>
  <div id="error_explanation">
    <ul>
      <% @message.errors.full_messages.each do |error_msg| %>
        <li><%= error_msg %></li>
      <% end %>
    </ul>
  </div>
<% end %>
```

Błędy walidacyjne są zapisywane w atrybucie `errors`, skąd możemy je później odczytać.

3. Praca własna

1. Zaimplementować akcję `edit` i `update` dla wiadomości.
2. Zaimplementować odpowiedni formularz dla obsługi edycji wiadomości.
3. Zaimplementować odpowiednie akcje (`new`, `create`, `edit` i `update`) i formularze dla komentarzy.

4. Devise

Instalacja i konfiguracja

```
# Gemfile

gem 'devise'
```

```
$ bundle install
$ bundle exec rails g devise:install
```

```
# config/environments/development.rb

config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }
```

```
$ bundle exec rails g devise User
$ bundle exec rails db:migrate
```

```
# controllers/messages_controller.rb

before_action :authenticate_user!
```

```
<!-- views/layouts/application.html.erb -->

<body>
  <p>
    <% if user_signed_in? %>
      Logged in as <strong><%= current_user.email %></strong>.
      <%= link_to "Logout", destroy_user_session_path, method: :delete %>
    <% else %>
      <%= link_to "Sign up", new_user_registration_path %> |
      <%= link_to "Login", new_user_session_path %>
    <% end %>
  </p>

  <% if notice %>
    <p class="alert alert-success"><%= notice %></p>
  <% end %>
  <% if alert %>
    <p class="alert alert-danger"><%= alert %></p>
  <% end %>

  <%= yield %>
</body>
```

5. Praca własna

1. Podłączyć model User do modelu Message (zamiast `author`).
2. Usprawnić dodawanie wiadomości, tak aby użytkownik był dołączany automatycznie (`current_user`).
3. Poprawić wyświetlanie wiadomości, tak aby pokazywała imię i nazwisko autora.

W przypadku pytań i eksperymentowania w domu pisz na slacku, na kanale `#pomoc`. Zaglądamy tam, więc będziemy starali się odpowiadać i doradzać.