Ruby on Rails

Autorización

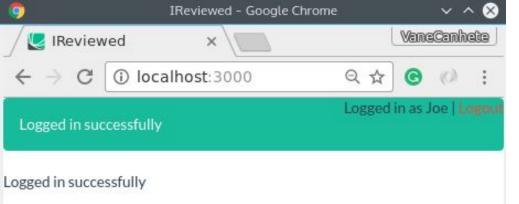
Security Helpers

- Para poder tener acceso al usuario logeado desde los views necesitamos crear un helper.
- Agreguemos los helpers logged_in? y current_user a
 ApplicationController y disponibilizarlos en todos los Controllers y
 views mediante helper_method.
- Haciendo esto podríamos agregar lógica al application.html.erb para el logout y también para desplegar información sobre el usuario logeado.

```
application controller.rb ×
                       sessions controller.rb ×
    class ApplicationController < ActionController::Base</pre>
      # Prevent CSRF attacks by raising an exception.
      # For APIs, you may want to use :null session instead.
      protect from forgery with: :exception
      before action :ensure login
      helper method :logged in?, :current user
 8
9
      protected
      def ensure login
10
        # Always go to login page unless session contains
11
        # reviewer id
12
13
        redirect to login path unless session[:reviewer id]
14
      end
15
      def logged in?
16
        session[:reviewer id] # nil is false
17
18
      end
19
20
      def current user
        @current user ||= Reviewer.find(session[:reviewer id])
21
22
      end
    end
```

views/layout/application.html.erb

```
<!DOCTYPE html>
<html>
<head>
  <title>IReviewed</title>
  <%= stylesheet link tag
                           'application', media: 'all', 'data-turbolinks-track' => true %>
  <%= javascript include tag 'application', 'data-turbolinks-track' => true %>
  <%= csrf meta tags %>
</head>
<body>
  <% if logged in? %>
  <div style='float: right;'>
   Logged in as <%= current user.name %> |
    <%= link to "Logout", logout path, method: :delete, class: "text-danger"%>
  </div>
  <% end %>
  <% flash.each do |key, value| %>
    <div class="alert alert-dismissible alert-success" id='<%= key %>'><%= value %></div>
  <% end %>
  <%= vield %>
</body>
</html>
```



Listing Books

New Book

Name	Author	
Eloquent Ruby	Russ Olsen	ShowEditDestroy
Beginning Ruby	Peter Cooper	ShowEditDestroy
Metaprogramming Ruby 2	Paolo Perrotta	ShowEditDestroy
Design Patterns in Ruby	Russ Olsen	ShowEditDestroy
The Ruby Programming Langu	ageDavid Flanagar	ShowEditDestroy

Autorización

Hemos implementado una autenticación básica, pero aún no hicimos nada por nuestra autorización.

Cualquier persona puede editar los libros y notas de cualquier otra persona.

Solución: Editar el BooksController para enviar solo datos del current_user a las vistas.

```
class BooksController < ApplicationController</pre>
  before action :set book, only: [:show, :edit, :update, :destroy]
 # GET /books
 # GET /books.json
  def index
   @books = current user.books.all
  end
 # GET /books/1
 # GET /books/1.json
                                    Listing Books
 def show
  end
 # GET /books/new
                                    Name
                                                             Author
 def new
                                    Metaprogramming Ruby 2Paolo PerrottaShowEditDestroy
   @book = current user.books.new
  end
                                    Design Patterns in Ruby Russ Olsen
                                                                          ShowEditDestrov
 # GET /books/1/edit
 def edit
                                    New Book
  end
 # POST /books
 # POST /books.json
 def create
```

@book = current user.books.new(book params)

Entonces...

Se puede utilizar la sesión para almacenar el usuario actual o su id.

Luego se pueden observar solo los recursos asociados a ese usuario.