

Laboratorio de Computación IV

Clase 16

Andrés Fortier

Repaso

- Seeds.
- Relaciones 1 a N.
- Restringir el contenido del indice al usuario logueado.

Roles




- *Role-based access control (RBAC)*
 - Se crean roles para distintas funciones.
 - Se asignan permisos a los roles.
 - Se asignan roles a las personas.
- Antes de realizar una acción se verifica que el usuario tenga un rol que permita llevarla a cabo.

Roles




- No se asignan permisos a personas
 - Fácil de modificar políticas a nivel global.
- Variantes
 - Un rol por usuario vs. listas de roles.
 - Herencia de roles.

Implementación de roles



- Se puede separar en dos grandes etapas
 - Definición y asignación de roles.
 - Control de roles antes de llevar adelante una acción.
- Se pueden implementar desde cero, aunque hay muchas librerías que lo hacen.
- Mayor flexibilidad implica mayor complejidad.

Implementación de roles



- En ruby / rails
 - rolify, role_model, etc.
 - Cancan (cancancan), pundit, authority, etc.
- Nosotros vamos a trabajar con
 - rolify - <https://github.com/RolifyCommunity/rolify>
 - pundit - <https://github.com/elabs/pundit>

rolify - tablas

users

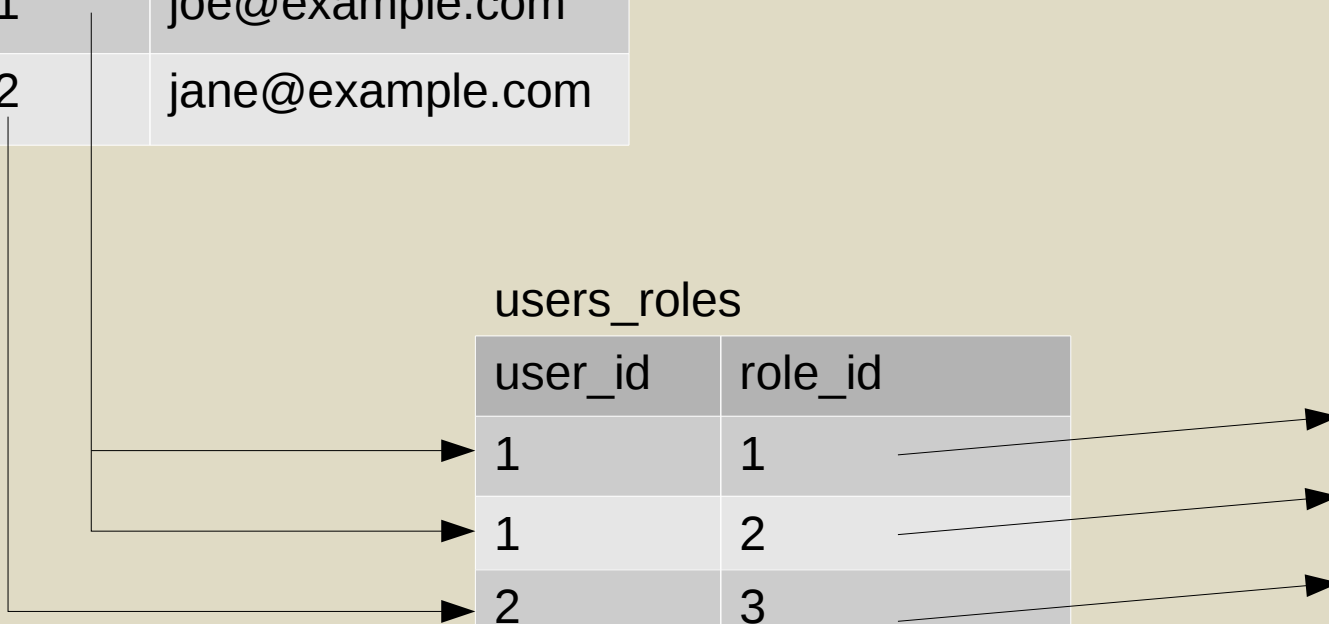
id	email
1	joe@example.com
2	jane@example.com

users_roles

user_id	role_id
1	1
1	2
2	3

roles

id	name
1	admin
2	manager
3	developer



rolify



- Hagan un commit de su repo.
- Y un `db:reset`.

```
$ bin/rails server
```


rolify

- Agreguemos la gema

```
/Gemfile
```

```
..  
gem 'rolify', '~> 5.1'  
...
```

```
$ bundle install
```

rolify

```
$ rails generate rolify Role User
  invoke  active_record
  create   app/models/role.rb
  insert   app/models/role.rb
  create   db/migrate/20150516205715_rolify_create_roles.rb
  insert   app/models/user.rb
  create   config/initializers/rolify.rb
```

=====

An initializer file has been created here:
config/initializers/rolify.rb, you can change rolify settings
to match your needs. Defaults values are commented out.

A Role class has been created in app/models (with the name
you gave as argument otherwise the default is role.rb), you
can add your own business logic inside.

Inside your User class (or the name you gave as argument
otherwise the default is user.rb), rolify method has been
inserted to provide rolify methods.

rolify

```
$ git status
```

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add <file>..." to update what will be committed)
```

```
  (use "git checkout -- <file>..." to discard changes in  
working directory)
```

```
    modified:   Gemfile
```

```
    modified:   Gemfile.lock
```

```
    modified:   app/models/user.rb
```

```
Untracked files:
```

```
  (use "git add <file>..." to include in what will be  
committed)
```

```
    app/models/role.rb
```

```
    config/initializers/rolify.rb
```

```
    db/migrate/20150516205715_rolify_create_roles.rb
```

rolify

```
$ bin/rake db:migrate
== 20150516205715 RolifyCreateRoles: migrating =====
-- create_table(:roles)
  -> 0.0141s
-- create_table(:users_roles, {:id=>false})
  -> 0.0004s
-- add_index(:roles, :name)
  -> 0.0007s
-- add_index(:roles, [:name, :resource_type, :resource_id])
  -> 0.0009s
-- add_index(:users_roles, [:user_id, :role_id])
  -> 0.0004s
== 20150516205715 RolifyCreateRoles: migrated (0.0169s) =====
```

rolify

- En la clase Role cambiar:

```
belongs_to :resource,  
           :polymorphic => true,  
           :optional => true
```

a:

```
belongs_to :resource,  
           :polymorphic => true
```

rolify

```
$ bin/rails console
> user = User.find(1)
> user.roles.empty?
  Role Exists (0.2ms)  SELECT  1 AS one FROM "roles" INNER
JOIN "users_roles" ON "roles"."id" = "users_roles"."role_id"
WHERE "users_roles"."user_id" = ? LIMIT 1  [["user_id", 1]]
=> true
```

rolify

```
> user.add_role :admin
  Role Load (0.1ms)  SELECT  "roles".* FROM "roles"  WHERE
"roles"."name" = 'admin' AND "roles"."resource_type" IS NULL
AND "roles"."resource_id" IS NULL  ORDER BY "roles"."id" ASC
LIMIT 1
  (0.1ms)  begin transaction
  SQL (1.0ms)  INSERT INTO "roles" ("created_at", "name",
"updated_at") VALUES (?, ?, ?)  [["created_at", "2015-05-16
21:33:57.681132"], ["name", "admin"], ["updated_at", "2015-
05-16 21:33:57.681132"]]
  (189.2ms)  commit transaction
  Role Load (0.2ms)  SELECT  "roles".* FROM "roles"  WHERE
"roles"."id" = ? LIMIT 1  [["id", 1]]
  (0.1ms)  begin transaction
  SQL (0.8ms)  INSERT INTO "users_roles" ("role_id",
"user_id") VALUES (?, ?)  [["role_id", 1], ["user_id", 1]]
  (171.4ms)  commit transaction
=> #<Role id: 1, name: "admin", resource_id: nil,
resource_type: nil, created_at: "2015-05-16 21:33:57",
updated_at: "2015-05-16 21:33:57">
```

rolify

```
> user.has_role? :admin
Role Load (0.2ms)  SELECT "roles".* FROM "roles" INNER JOIN
"users_roles" ON "roles"."id" = "users_roles"."role_id" WHERE
"users_roles"."user_id" = ? AND (((roles.name = 'admin') AND
(roles.resource_type IS NULL) AND (roles.resource_id IS
NULL)))  [["user_id", 1]]
=> true
```


- Modifiquemos la configuración de rails_admin

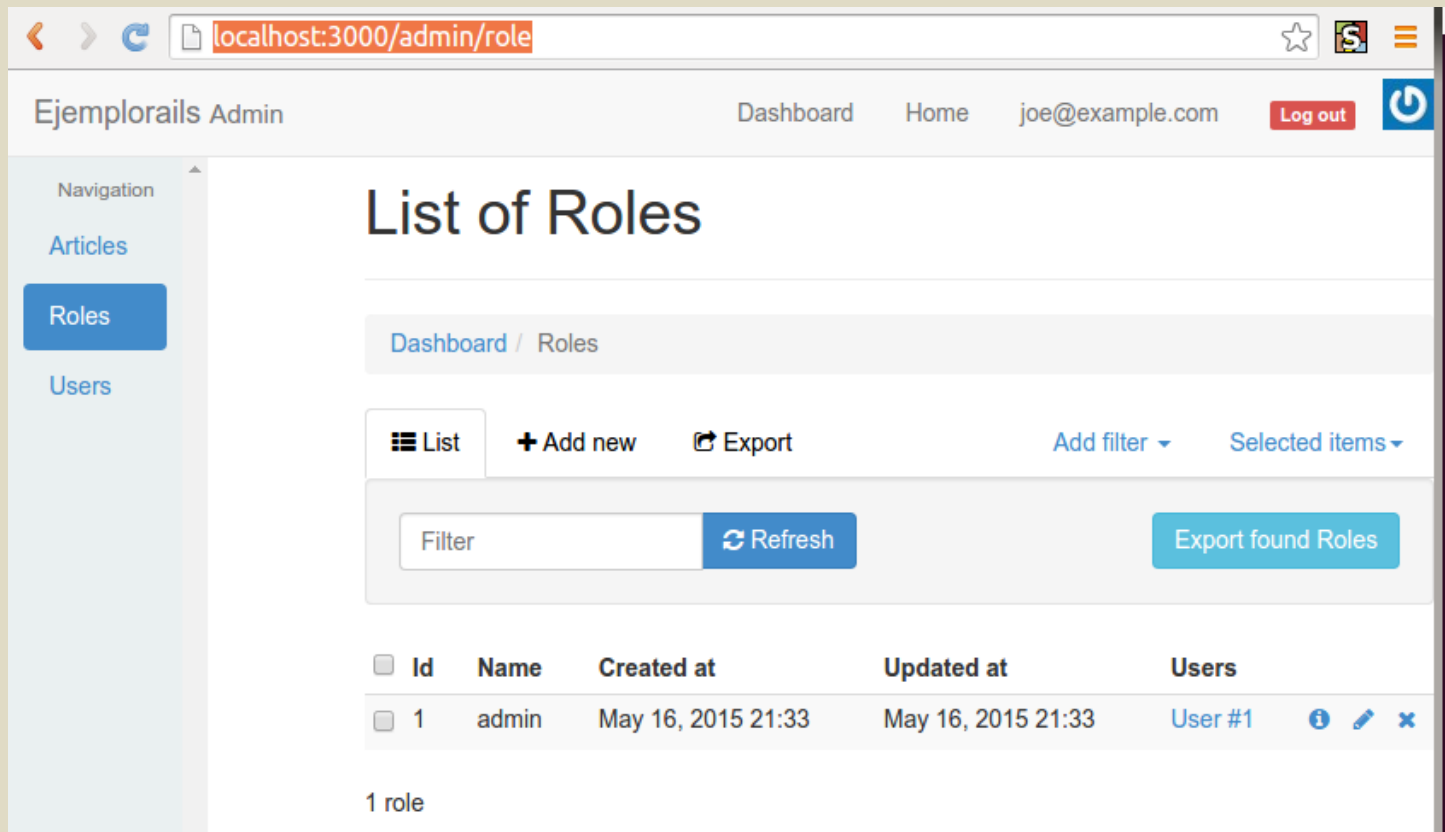
```
config/initializers/rails_admin.rb
```

```
..  
  config.authorize_with do |controller|  
    unless current_user.has_role? :admin  
      redirect_to main_app.root_path  
    end  
  end  
...  

```

rolify

- Abran dos navegadores, uno con cada usuario logueado.
- Vayan a <http://localhost:3000/admin/role>



The screenshot shows a web browser at the URL `localhost:3000/admin/role`. The page is titled "Ejemplorails Admin" and has a navigation bar with links for "Dashboard", "Home", and the user "joe@example.com" with a "Log out" button. A left sidebar contains "Navigation" links for "Articles", "Roles" (highlighted), and "Users". The main content area is titled "List of Roles" and includes a breadcrumb "Dashboard / Roles". Below this are controls for "List", "Add new", and "Export", along with "Add filter" and "Selected items" dropdowns. A search filter box with a "Refresh" button and an "Export found Roles" button are also present. A table lists the roles with columns: Id, Name, Created at, Updated at, and Users. The table contains one row for an "admin" role created on May 16, 2015, assigned to "User #1".

<input type="checkbox"/>	Id	Name	Created at	Updated at	Users
<input type="checkbox"/>	1	admin	May 16, 2015 21:33	May 16, 2015 21:33	User #1

1 role

- Agreguen al usuario 2 al rol y guarden

Navigation

Articles

Roles

Users

Edit Role 'admin'

Dashboard / Roles / Admin / Edit

Show **Edit** **Delete**

Name
Optional. Length up to 255.

Users

User #2

User #1

Choose all **Clear all**

+ Add a new User

Optional.

Save **Save and add another** **Save and edit** **Cancel**

Navigation

Articles

Roles

Users

List of Roles

Role successfully updated

Dashboard / Roles

List

+ Add new

Export

Add filter

Selected items

Filter

Refresh

Export found Roles

<input type="checkbox"/>	Id	Name	Created at	Updated at	Users	
<input type="checkbox"/>	1	admin	May 16, 2015 21:33	May 16, 2015 21:33	User #1 and User #2	<div><div></div><div></div><div></div></div>

1 role

rolify

- Prueben de ingresar nuevamente a la página de admin con el usuario 2.
- La clase que viene nos vamos a centrar en la autorización de acciones basado en roles.

Estilos (CSS)



- CSS - Separar la presentación del contenido.
- No vamos a entrar en detalle ahora respecto de cómo funciona, simplemente lo vamos a usar.
- Varias librerías
 - Bootstrap
 - Foundation
 - Zimit
 - lnk
 - ...

Estilos (CSS)



- Bootstrap (<http://getbootstrap.com/>)
 - Framework HTML/CSS/Javascript.
 - Mobile first.
 - Grid system.
 - Estilos para los componentes HTML.
 - Conjunto de clases CSS para tener una página consistente.
 - Componentes dinámicos que utilizan javascript.

CSS y layouts



- Vamos a
 - Instalar una gema para incorporar bootstrap.
 - Instalar una gema para generar los layouts y páginas de devise con el estilo de bootstrap.

bootstrap-sass

- Incorpora los estilos de bootstrap a una aplicación rails

```
/Gemfile
```

```
..  
gem 'bootstrap-sass'  
...
```

```
$ bundle install
```

rails_layout

- Genera layouts para distintas librerías
 - Puede generar las vistas de devise

```
/Gemfile
```

```
..  
group :development do  
  gem 'rails_layout'  
end  
...
```

```
$ bundle install
```

rails_layout

```
$ bundle install
$ rails generate layout:install bootstrap3 --force
    remove    app/assets/stylesheets/application.css
    create     app/assets/stylesheets/application.css.scss
    create
app/assets/stylesheets/framework_and_overrides.css.scss
    force     app/assets/javascripts/application.js
    remove    app/assets/stylesheets/simple.css
    remove
app/assets/stylesheets/foundation_and_overrides.css.scss
    append
app/assets/stylesheets/framework_and_overrides.css.scss
    remove    app/views/layouts/application.html.erb
    create    app/views/layouts/application.html.erb
    create    app/views/layouts/_messages.html.erb
    create    app/views/layouts/_navigation.html.erb
    create    app/views/layouts/_navigation_links.html.erb
```

rails_layout

```
$ cat app/views/layouts/_messages.html.erb
<%= Rails flash messages styled for Bootstrap 3.0 %>
<% flash.each do |name, msg| %>
  <% if msg.is_a?(String) %>
    <div class="alert alert-<%= name.to_s == 'notice' ?
'success' : 'danger' %>">
      <button type="button" class="close" data-
dismiss="alert" aria-hidden="true">&times;</button>
      <%= content_tag :div, msg, :id => "flash_#{name}" %>
    </div>
  <% end %>
<% end %>
```

rails_layout

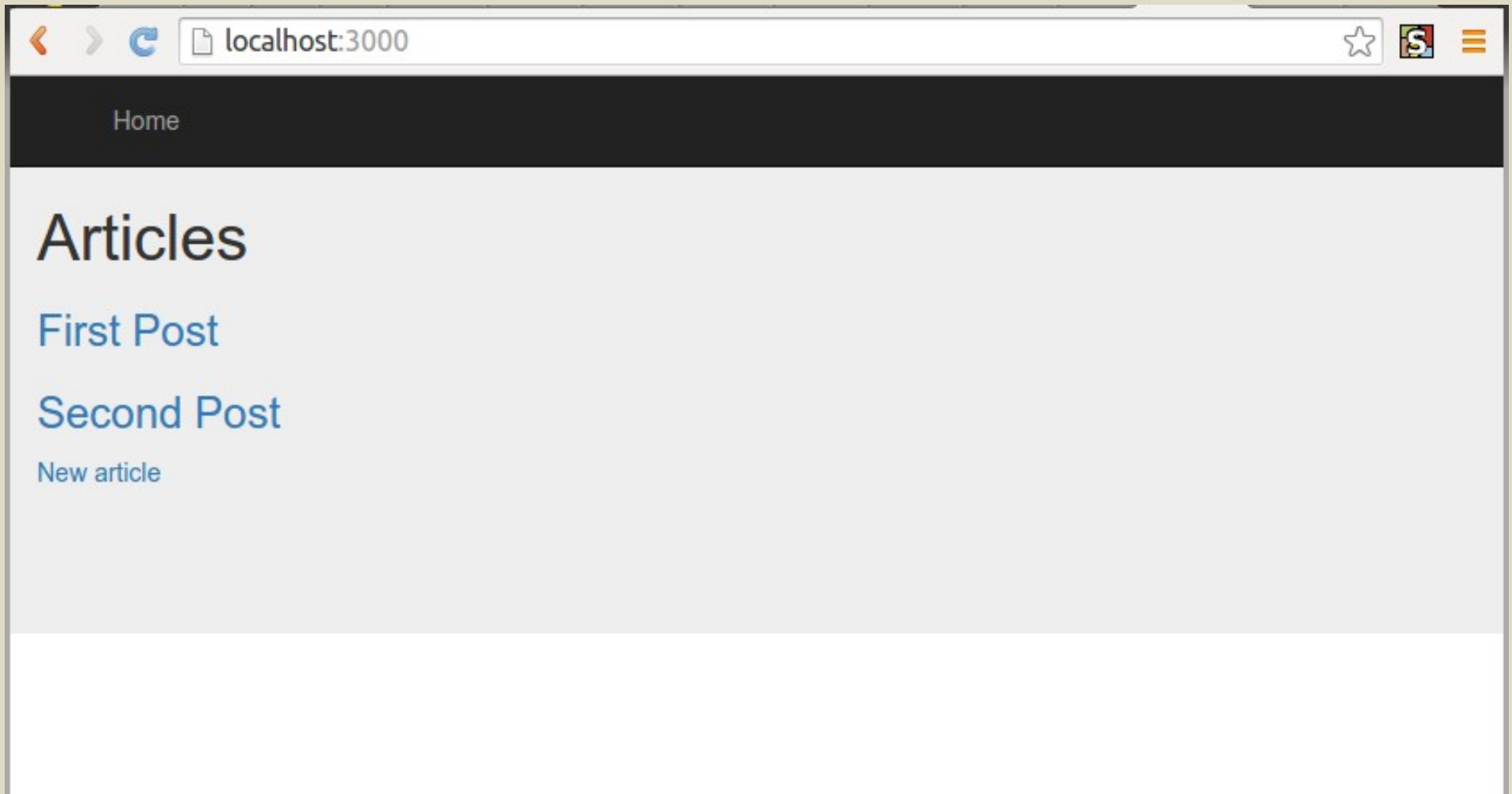
```
$ cat app/views/layouts/application.html.erb
<!DOCTYPE html>
<html>
  <head>
    ...
  </head>
  <body>
    <header>
      <%= render 'layouts/navigation' %>
    </header>
    <main role="main">
      <%= render 'layouts/messages' %>
      <%= yield %>
    </main>
  </body>
</html>
```

rails_layout

```
$ rails generate layout:devise bootstrap3  
  create  app/views/devise/sessions/new.html.erb  
  create  app/views/devise/passwords/new.html.erb  
  create  app/views/devise/passwords/edit.html.erb  
  create  app/views/devise/registrations/new.html.erb  
  create  app/views/devise/registrations/edit.html.erb
```

rails_layout

```
$ bin/rails server
```



rails_layout

[Home](#) [Login](#)

Sign in

Email


[Sign up](#)

Password

[Forgot password?](#)

☐ Remember me

Entrega 31/05/2016



- Código y ejemplo en Openshift.
- Manejo de usuarios
 - Con devise.
 - Login/Logout/Registración.
 - Links acordes en el layout.
- Roles
 - rolify
 - Al menos separar entre usuarios “normales” y administradores.

Entrega 31/05/2016

- Consola de administrador
 - rails_admin
 - Integrada con devise y rolify.
 - Accesible sólo para los que tengan el rol “admin”.
- Layout con bootstrap 3 como vimos hoy.
- Página de índice pública
 - Puede haber otras dependiendo de la aplicación.
- Página(s) que requieren un usuario logueado.

Entrega 31/05/2016

- Cosas que suman
 - Cuanta mas funcionalidad de su dominio de problema, mejor.
 - Restringir acciones en base cierto criterio
 - Ej. Sólo el creador de un elemento puede modificarlo.
 - Set de ejemplos pre-cargados.
 - Estilos (bootstrap).

Tarea para el hogar



- Desde la página de administración editen el rol admin y quiten a todos los usuarios. Prueben de re-establecer a **joe@example.com** como admin desde:
 - Una consola de rails.
 - Una consola de base de datos.
- Modificar su archivo de seeds para que **joe@example.com** sea admin.
- Coloquen la página de artículos como el root.

Tarea para el hogar

- Agreguen link al nav bar:
 - Si están logueados, “Logout”
 - Si no están logueados, “Login”

rails_layout

