

Laboratorio de Computación IV

Clase 13

Andrés Fortier

Repaso

- Validaciones.
 - *save* vs *save!*
- *Partials*.
- *Layouts*.
- *Flash messages*.
- Devise Gem.

Blog - Relaciones

- Comencemos con la misma BD
 - Ya veremos mas adelante que esto se puede automatizar

```
$ bin/rails console
> User.delete_all
  SQL (182.3ms)  DELETE FROM "users"
=> 1
> User.count
  (0.3ms)  SELECT COUNT(*) FROM "users"
=> 0
> Article.delete_all
...
> Article.count
  (0.3ms)  SELECT COUNT(*) FROM "articles"
=> 0
```

Blog - Relaciones

```
> Article.new({title: 'First Post', text: 'Hi'}).save!  
...  
> Article.new({title: 'Second Post', text: 'Ho'}).save!  
...  
> Article.new({title: 'Third Post', text: 'Hu'}).save!
```

- Vayan a http://localhost:3000/users/sign_up y creen dos usuarios
 - [usr1@test.com](#)
 - [usr2@test.com](#)

Blog - Relaciones

- ¿Cómo se guardan las relaciones (1-1 y 1-N) en una BD relacional?

cuentas

id	id_titular	saldo
1	1	\$1.500,00
2	15	\$300.000,00
3	23	\$12,50

titulares

id	nombre
1	"Juan Perez"
15	"Jose Julio"
23	"Anastacio Ponce"

Blog - Relaciones



- Queremos que nuestros artículos tengan un autor
 - Crear una migration para agregar la columna de id de usuario a la tabla de artículos.
 - Indicarle al modelo (*ActiveRecord*) que existe esa relación.

Blog - Relaciones

- Crear la *migration*

```
$ bin/rails generate migration AddAuthorToArticles author:references
  invoke  active_record
  create  db/migrate/20150509203054_add_author_to_articles.rb
```

```
$ cat db/migrate/20150509203054_add_author_to_articles.rb
class AddAuthorToArticles < ActiveRecord::Migration
  def change
    add_reference :articles, :author, index: true
  end
end
```

Blog - Relaciones

- Ejecutemos la *migration*

```
$ bin/rake db:migrate
== 20150509203054 AddAuthorToArticles: migrating =====
-- add_reference(:articles, :author, {:index=>true})
   -> 0.0133s
== 20150509203054 AddAuthorToArticles: migrated (0.0134s) ==
```


Blog - Relaciones

- Veamos que se modificó en nuestro schema

```
$ git diff db/schema.rb
...
-ActiveRecord::Schema.define(version: 20150503182243) do
+ActiveRecord::Schema.define(version: 20150509203054) do

  create_table "articles", force: true do |t|
    t.string    "title"
    t.text      "text"
    t.datetime  "created_at"
    t.datetime  "updated_at"
+   t.integer   "author_id"
  end

+  add_index "articles", ["author_id"], name:
+  "index_articles_on_author_id"
+
  ...
```

Blog - Relaciones

- En una nueva consola

```
$ bin/rails console
> Article.all
  Article Load (2.9ms)  SELECT "articles".* FROM "articles"
=> #<ActiveRecord::Relation [
#<Article id: 10, title: "First Post", text: "Hi",
created_at: "2015-05-09 20:00:51", updated_at: "2015-05-09
20:00:51", author_id: nil>,
#<Article id: 11, title: "Second Post", text: "Ho",
created_at: "2015-05-09 20:01:55", updated_at: "2015-05-09
20:01:55", author_id: nil>,
#<Article id: 12, title: "Third Post", text: "Hu",
created_at: "2015-05-09 20:02:06", updated_at: "2015-05-09
20:02:06", author_id: nil>]>
```

Blog - Relaciones

```
> article = Article.first
Article Load (0.4ms)  SELECT  "articles".* FROM "articles"
ORDER BY "articles"."id" ASC LIMIT 1
=> #<Article id: 10, title: "First Post", text: "Hi",
created_at: "2015-05-09 20:00:51", updated_at: "2015-05-09
20:00:51", author_id: nil>
> article.author
NoMethodError: undefined method `author' for
#<Article:0x000000041a3840>
```

Blog - Relaciones

- Nos falta indicarle al modelo que existe una relación con la clase `User`

```
app/models/article.rb
```

```
class Article < ActiveRecord::Base
  validates :title,
            presence: true,
            length: { minimum: 5 }
  belongs_to :author, class_name: 'User'
end
```

- *Nota: Si la relación se llamara `user` no necesitaríamos especificar la clase.*

Blog - Relaciones

- En una consola

```
$ bin/rails console
> article = Article.first
  Article Load (0.4ms)  SELECT  "articles".* FROM "articles"
  ORDER BY "articles"."id" ASC LIMIT 1
=> #<Article id: 10, title: "First Post", text: "Hi",
created_at: "2015-05-09 20:00:51", updated_at: "2015-05-09
20:00:51", author_id: nil>
> article.author
=> nil
```

Blog - Relaciones

- Asignemos un autor al artículo

```
> usr1 = User.where({email: 'usr1@test.com'}).first
User Load (0.5ms)  SELECT "users".* FROM "users"  WHERE
"users"."email" = 'usr1@test.com'  ORDER BY "users"."id" ASC
LIMIT 1
=> #<User id: 3, email: "usr1@test.com",...>
> article.author = usr1
=> #<User id: 3, email: "usr1@test.com",...>
> article.save!
(0.2ms)  begin transaction
SQL (1.4ms)  UPDATE "articles" SET "author_id" = ?,
"updated_at" = ? WHERE "articles"."id" = 10  [["author_id",
3], ["updated_at", "2015-05-09 20:53:12.487093"]]
(201.4ms)  commit transaction
=> true
```

Blog - Relaciones

- Asignemos un autor a todos los artículos

```
> Article.all.each do | article |
>   article.author = usr1
?>   article.save!
?>   end
Article Load (0.5ms)  SELECT "articles".* FROM "articles"
(0.1ms)  begin transaction
(0.1ms)  commit transaction
(0.1ms)  begin transaction
SQL (1.1ms)  UPDATE "articles" SET "author_id" = ?...
(204.3ms)  commit transaction
(0.1ms)  begin transaction
SQL (1.2ms)  UPDATE "articles" SET "author_id" = ?...
(204.1ms)  commit transaction
=> [
#<Article id: 10, title: "First Post",..., author_id: 3>,
#<Article id: 11, title: "Second Post",..., author_id: 3>,
#<Article id: 12, title: "Third Post",..., author_id: 3>]
```

Blog - Relaciones

- Visualicemos el autor relacionado al artículo

```
app/views/articles/show.html.erb
```

```
<h1>
  <%= @article.title %>
</h1>

<h3>
  By <%= @article.author.email %>
</h3>

<p>
  <strong>Text:</strong>
  <%= @article.text %>
</p>

...
```


Blog - Relaciones

- Nos falta asociar al usuario logueado al artículo al crearlo

```
app/controllers/articles_controller.rb
```

```
class ArticlesController < ApplicationController
  skip_before_action :authenticate_user!, only: [:index]

  ...

  def create
    @article = Article.new(article_params)
    @article.author = current_user

    begin
      @article.save!
    ...
  end
end
```

Blog - Relaciones

- Vayamos a una consola de BD

```
$ bin/rails dbconsole
sqlite> select * from articles;
10|First Post|Hi|2015-05-09 22:06:03.139767|2015-05-09
22:06:03.139767|3
11|Second Post|Ho|2015-05-09 22:07:32.559730|2015-05-09
22:07:32.559730|3
12|Third Post|Hu|2015-05-09 22:07:53.091840|2015-05-09
22:07:53.091840|3
sqlite> update articles set author_id=4 where title="First
Post";
sqlite> select * from articles;
10|First Post|Hi|2015-05-09 22:06:03.139767|2015-05-09
22:06:03.139767|4
11|Second Post|Ho|2015-05-09 22:07:32.559730|2015-05-09
22:07:32.559730|3
12|Third Post|Hu|2015-05-09 22:07:53.091840|2015-05-09
22:07:53.091840|3
```

Blog - Relaciones

- Vayamos a una consola de rails

```
$ bin/rails console
> article = Article.where({title:'First Post'}).first
  Article Load (0.4ms)  SELECT  "articles".* FROM "articles"
WHERE "articles"."title" = 'First Post' ORDER BY
"articles"."id" ASC LIMIT 1
=> #<Article id: 10, title: "First Post", text: "Hi",
created_at: "2015-05-09 22:06:03", updated_at: "2015-05-09
22:06:03", author_id: 4>
> article.author.email
  User Load (0.2ms)  SELECT  "users".* FROM "users" WHERE
"users"."id" = ? LIMIT 1  [["id", 4]]
=> "usr2@test.com"
```

rails_admin

- Agreguemos la gema

```
/Gemfile
```

```
..  
gem 'rails_admin', '~> 0.6.7'  
...
```

```
$ bundle install
```

```
$ rails generate rails_admin:install  
...
```

rails_admin

- Modifiquemos la configuración

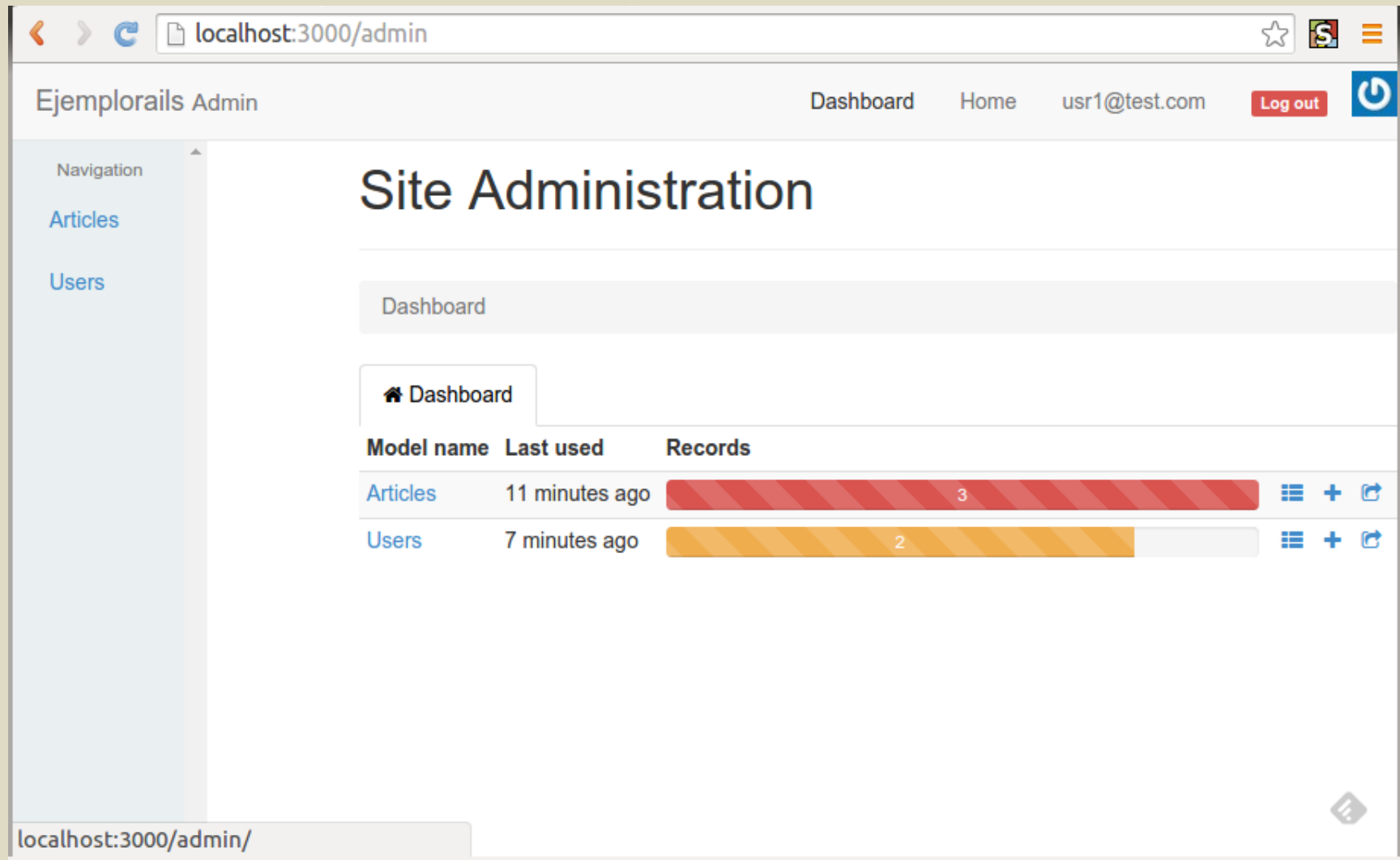
```
config/initializers/rails_admin.rb
```

```
..  
  ## == Devise ==  
  config.authenticate_with do  
    warden.authenticate! scope: :user  
  end  
  config.current_user_method(&:current_user)  
...
```







```
$ bin/rails server
```

rails_admin

- Vayan a <http://localhost:3000/admin>



The screenshot shows the Rails Admin web interface in a browser window. The address bar displays `localhost:3000/admin`. The page title is "Ejemplorails Admin". The top navigation bar includes links for "Dashboard", "Home", the user "usr1@test.com", and a "Log out" button. A left sidebar contains a "Navigation" menu with links for "Articles" and "Users". The main content area is titled "Site Administration" and features a "Dashboard" tab. Below the tab is a table with the following data:

Model name	Last used	Records	
Articles	11 minutes ago	3	  
Users	7 minutes ago	2	  

The browser's status bar at the bottom shows `localhost:3000/admin/`.

rails_admin

The screenshot displays the rails_admin web interface in a browser window. The address bar shows `localhost:3000/admin/article`. The page title is "Ejemplorails Admin". The top navigation bar includes links for "Dashboard", "Home", the user "usr1@test.com", a "Log out" button, and a power icon. A left sidebar contains a "Navigation" menu with "Articles" (selected) and "Users". The main content area is titled "List of Articles" and shows a breadcrumb "Dashboard / Articles". Below this, there are controls for "List", "Add new", and "Export", along with "Add filter" and "Selected items" dropdowns. A search bar with a "Filter" label and a "Refresh" button is present, followed by an "Export found Articles" button. The table below lists three articles with columns for Id, Title, Text, Created at, Updated at, and Author. Each row has a checkbox and a set of action icons (info, edit, delete, view). At the bottom left, it says "3 articles".

localhost:3000/admin/article

Ejemplorails Admin

Dashboard Home usr1@test.com Log out

Navigation

Articles

Users

List of Articles

Dashboard / Articles

List + Add new Export Add filter Selected items

Filter Refresh Export found Articles

<input type="checkbox"/>	Id	Title	Text	Created at	Updated at	Author	
<input type="checkbox"/>	3	Third Post	Hu	May 09, 2015 22:07	May 09, 2015 22:07	User #1	i pencil x eye
<input type="checkbox"/>	2	Second Post	Ho	May 09, 2015 22:07	May 09, 2015 22:07	User #1	i pencil x eye
<input type="checkbox"/>	1	First Post	Hi	May 09, 2015 22:06	May 09, 2015 22:06		i pencil x eye

3 articles

rails_admin

The screenshot displays the rails_admin web interface in a browser window. The address bar shows the URL `localhost:3000/admin/article/new`. The page header includes the title "Ejemplorails Admin" and navigation links for "Dashboard", "Home", "usr1@test.com", and a "Log out" button. A left sidebar contains a "Navigation" menu with "Articles" (selected) and "Users". The main content area is titled "New Article" and features a breadcrumb trail: "Dashboard / Articles / New". Below the breadcrumb, there are three tabs: "List", "Add new" (active), and "Export". The "Add new" tab contains a form with three fields: "Title" (required, length 5-255), "Text" (optional), and "Author" (a dropdown menu). The "Author" dropdown is open, showing "User #2" and "User #1". At the bottom of the form, there are four buttons: "Save", "Save and add another", "Save and edit", and "Cancel".

localhost:3000/admin/article/new

Ejemplorails Admin

Dashboard Home usr1@test.com Log out

Navigation

Articles

Users

New Article

Dashboard / Articles / New

List + Add new Export

Title

Required. Length of 5-255.

Text

Optional.

Author

User #2

User #1

+ Add a new User Edit this User

Save Save and add another Save and edit Cancel

Importante

- Saber manejar diversas herramientas
 - Backend de administración (ej. rails_admin)
 - Consola de Rails.
 - Consola de BD.

Tarea para el hogar

- Agregar una validación para que el autor de un artículo no pueda ser nulo.
- Familiarizarse con la interfaz de admin.