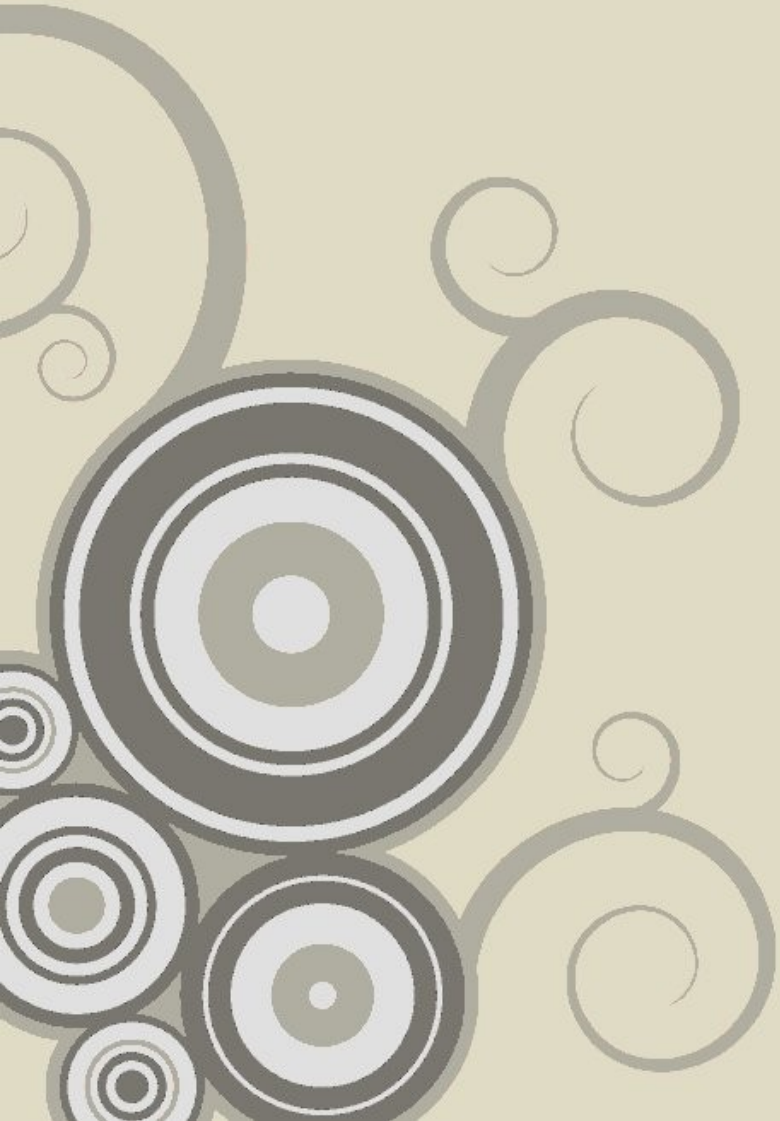
Laboratorio de Computación IV



Clase 14

Andrés Fortier

Repaso

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

Autenticación vs. Autorización

- Autenticación: saber quién es el usuario.
- Autorización: saber si el usuario tiene permisos para hacer algo.

Blog – Autenticación

- Agreguemos la gema

```
/Gemfile
..
gem 'devise', '~> 3.5.6'
...
```

```
$ bundle install
```

```
$ rails generate devise:install
   create  config/initializers/devise.rb
   create  config/locales/devise.en.yml
   ...
```

Blog – Autenticación

- Agregar configuración para el mailer

config/environments/development.rb

```
...

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

...
```

```
$ rails generate devise User
   invoke  active_record
   create    db/migrate/20150503182243_devise_create_users.rb
   create    app/models/user.rb
   insert    app/models/user.rb
    route  devise_for :users
```

Blog – Autenticación

```
$ bin/rake db:migrate
== 20150503182243 DeviseCreateUsers: migrating ==============
-- create_table(:users)
   -> 0.0171s
-- add_index(:users, :email, {:unique=>true})
   -> 0.0007s
-- add_index(:users, :reset_password_token, {:unique=>true})
   -> 0.0008s
== 20150503182243 DeviseCreateUsers: migrated (0.0189s) =====
```

Blog – Autenticación

```
$ cat db/schema.rb
ActiveRecord::Schema.define(version: 20150503182243) do
  create_table "articles", force: true do |t|
    ...
  end
  create_table "users", force: true do |t|
    t.string   "email",                  default: "", null: false
    t.string   "encrypted_password", default: "", null: false
    t.string   "reset_password_token"
    t.datetime "reset_password_sent_at"
    t.datetime "remember_created_at"
    t.integer  "sign_in_count",          default: 0,  null: false
    t.datetime "current_sign_in_at"
    t.datetime "last_sign_in_at"
    t.string   "current_sign_in_ip"
    t.string   "last_sign_in_ip"
    t.datetime "created_at"
    t.datetime "updated_at"
  end
    ...
end
```

Blog – Autenticación

- Reinicien el servidor.
- Vayan al índice.
  - ¡No cambió nada!
- Debemos indicar explícitamente que páginas queremos proteger
- Recordemos que nuestro controller es (y en general todos los que creemos serán) subclase de *ApplicationController*.

Blog – Autenticación

- Agreguemos autenticación

/app/controllers/application_controller.rb

```
class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  protect_from_forgery with: :exception

  before_action :authenticate_user!

end
```

- Vayan nuevamente al home
- Creen un usuario

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 usuario
  - 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.
- Hagan un commit para tener el working directory limpio.

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!
    ...
```

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', '~> 0.8.1'
...
```

```
$ bundle install
```

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

rails_admin

- Modifiquemos la configuración

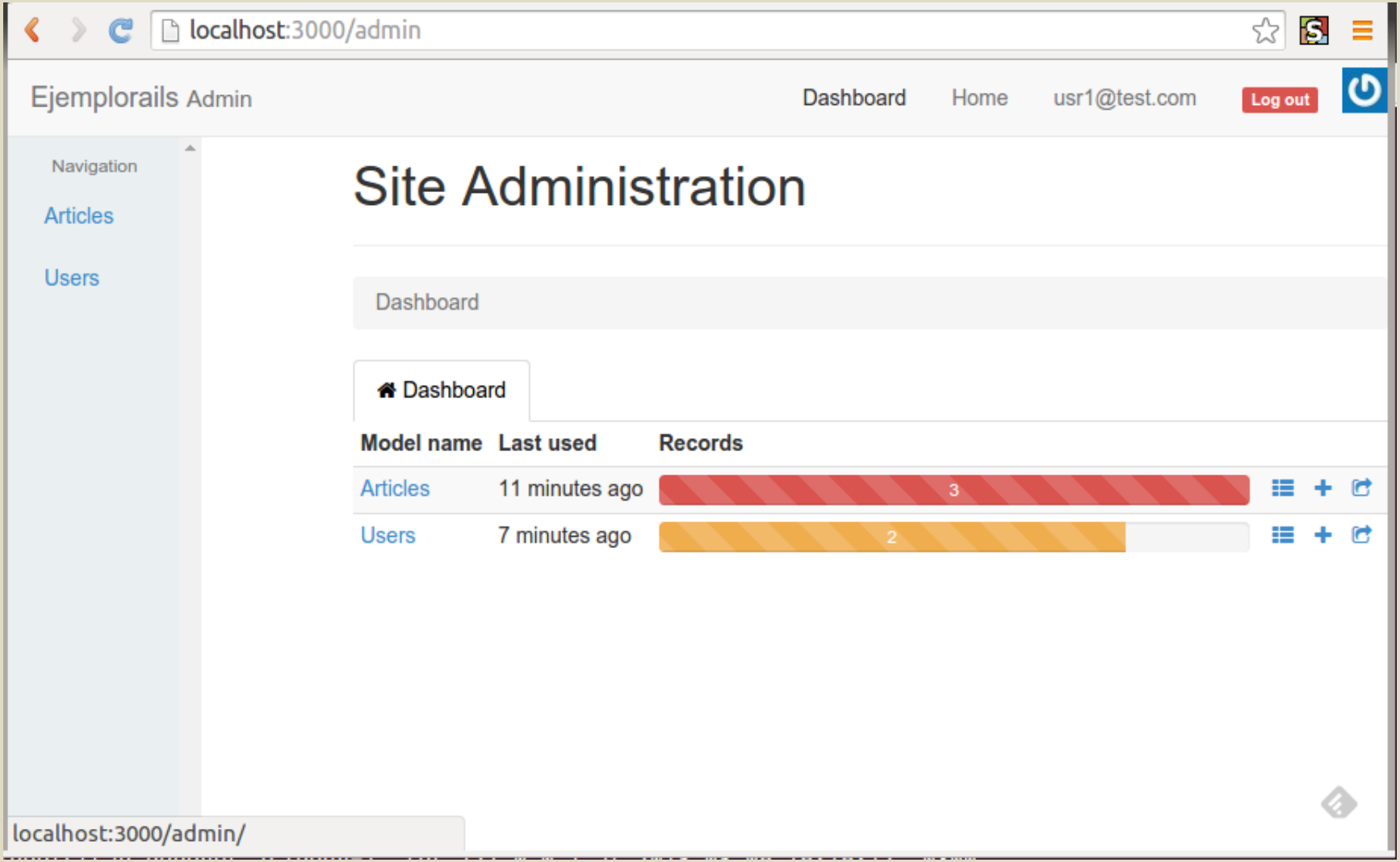
```
config/initializers/rails_admin.rb
```

```
..
  ## == 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



rails_admin

localhost:3000/admin/article

Ejemplorails Admin

Dashboard Home usr1@test.com Log out

Navigation

Articles

Users

List of Articles

Dashboard / Articles

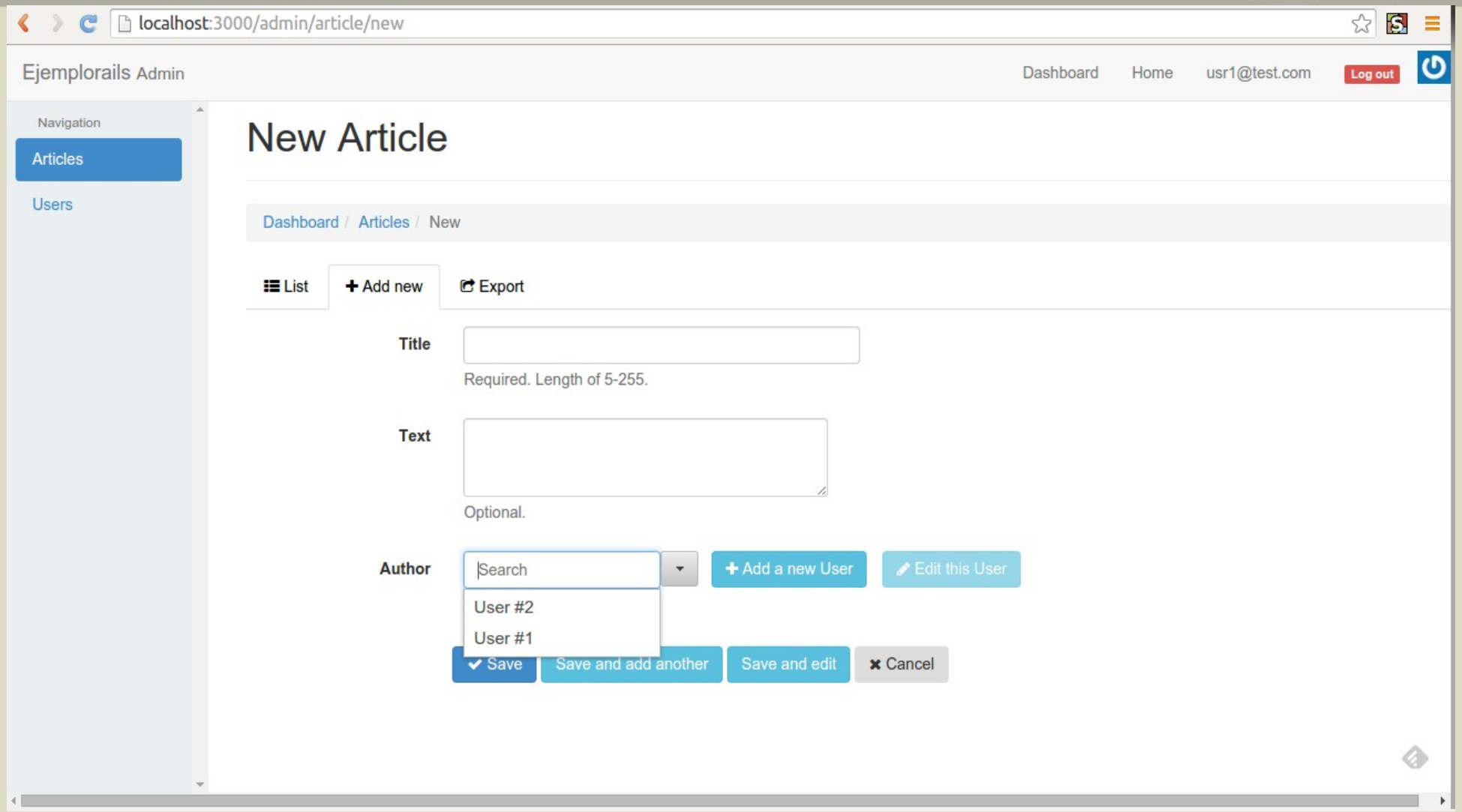
List + Add new Export Add filter Selected items

Filter Refresh Export found Articles

Id	Title	Text	Created at	Updated at	Author
3	Third Post	Hu	May 09, 2015 22:07	May 09, 2015 22:07	User #1
2	Second Post	Ho	May 09, 2015 22:07	May 09, 2015 22:07	User #1
1	First Post	Hi	May 09, 2015 22:06	May 09, 2015 22:06	

3 articles

rails_admin


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
Search
Add a new User
Edit this User
User #2
User #1
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.