



Autenticación

Sesión Conceptual 1





Inicio

{desafío}
latam_



15 minutos

- Crear formularios de login y acciones de salida para un usuario y utilizar la sesión para almacenar valores asociados al usuario actual.
- Limitar acceso a secciones con base en el rol del usuario y seleccionar una página de inicio en función del rol del usuario.

Objetivo



Desarrollo

{desafío}
latam_



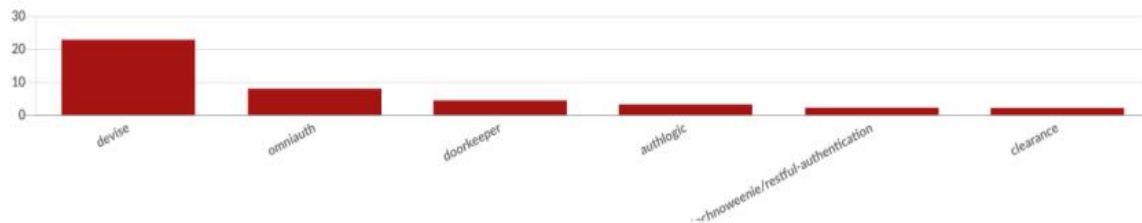
150 minutos

CATEGORY

[Edit this category](#)

Web Authentication

Handle the authentication of users in web applications written in frameworks like Rails

☒ Full☐ Compact☐ Table

Sesiones y Cookies

Sesiones

- Una **Sesión** es un dato (variable auxiliar) guardado en alguna parte (hay varias formas de almacenar la Sesión, según su contenido se define el lugar de almacenamiento) que nos permite almacenar información acerca de **quién** nos hace la solicitud.

- Una **Cookie** es el valor de una variable, que se almacena en la memoria temporal del navegador.
- En la Sesión podríamos guardar otras cosas y no solo una referencia al usuario actual. También podemos guardar objetos completos como podría ser la información de los carros de compra.

Ejercicio

```
$ rails new cookies_jam  
[...]  
$ cd cookies_jam /  
$ rails db:migrate  
$ rails g controller cookies set show delete  
[...]
```

Completamos el controller y revisamos las rutas

```
cookies_controller.rb
1 class CookiesController < ApplicationController
2   def set
3     cookies[:user_alias] = 'CookieJammer'
4     cookies[:client_id] = '874325gqwhwq343242'
5   end
6
7   def show
8     @user_alias = cookies[:user_alias]
9     @client_id = cookies[:client_id]
10  end
11
12  def delete
13    cookies.delete :user_alias
14    cookies.delete :client_id
15  end
16 end
17
```

```
routes.rb
1 Rails.application.routes.draw do
2   get 'cookies/set'
3   get 'cookies/show'
4   get 'cookies/delete'
5 end
6
```

Show me the Cookies!

Usamos variables de instancia para mostrar las cookies.

```
show.html.erb
1 <table>
2   <tr>
3     <td>User Alias:</td>
4     <td><%= @user_alias %></td>
5   </tr>
6   <tr>
7     <td>Client Id:</td>
8     <td><%= @client_id %></td>
9   </tr>
10 </table>
11
```

Primer current_user

Ahora modifiquemos el controlador para complementar las sesiones con las cookies y tener una implementación del famoso current_user.

```
cookies_controller.rb
1 class CookiesController < ApplicationController
2   def set
3     session[:current_user] = {
4       test_id: {
5         name: 'CookieJammer',
6         client_id: '874325gqwhwq343242'
7       }
8     }
9     cookies[:session_id] = 'test_id'
10  end
11
12  def show
13    current_user = session[:current_user][cookies[:session_id]]
14    @user_alias = current_user[:name.to_s]
15    @client_id = current_user[:client_id.to_s]
16  end
17
18  def delete
19    session.delete :user_alias
20    session.delete :client_id
21  end
22 end
```

Sistema de autenticación manual

Es poco frecuente programar un sistema de autenticación completamente desde cero

**Existen varios sistemas de autenticación que
han sido
extraídos en formas de gemas**

**Pero es necesario hacer uno en nuestro
proceso de formación como Developers**

```
$ rails new auth_jam  
[...]  
$ cd auth_jam/  
$ rails db:migrate  
$ rails g model User email:string password_digest :string  
[...]
```

Desde la versión 3.1 tenemos disponible el método *has_secure_password* en nuestros modelos. Esta función agrega métodos a las instancias de nuestro modelo para verificar sus passwords almacenadas mediante la función BCrypt.

Descomentar esta línea del Gemfile

Bundle install en la terminal

```
# Use ActiveRecord has_secure_password  
gem 'bcrypt' , '~> 3.1.7'
```

Revisar migración y agregar índice en el campo email

Luego ejecutar la migración

```
20190213152556_create_users.rb
1  class CreateUsers < ActiveRecord::Migration[5.1]
2    def change
3      create_table :users do |t|
4        t.string :email
5        t.string :password_digest
6        t.timestamps
7        t.index :email, unique: true
8      end
9    end
10  end
11
```

Configuración del modelo

```
user.rb
1  class User < ApplicationRecord
2    has_secure_password
3    validates :email, presence: true, uniqueness: { case_sensitive: false }
4  end
5  |
```

Probando en la consola

```
$ rails console  
  
> user = User.create(email: 'test@dsl.com', password: 'Contodo2019',  
password_confirmation: 'Contodo2019')  
[...]  
  
> user.authenticate('Fakepass')  
=> false  
  
> user.authenticate('Contodo2019')  
  
> #<User id: 1, email: "test@dsl.com", password_digest:  
"$2a$10$pBi/pbBRza290q59aGnTo0YpT/k..." , created_at: "2019 -02 -13 17:36:21" ,  
updated_at: "2019-02-13 17:36:21">  
  
" 2019-02-13 1 7: 3 6: 2 1 ">
```

Continuamos con controladores y rutas

```
$ rails generate controller sessions new create destroy  
[...]  
$ rails generate controller home index  
[...]
```

```
routes.rb  
1  Rails.application.routes.draw do  
2    resources :sessions, only: [:new, :create, :destroy]  
3    root 'home#index'  
4  end  
5
```

Método `current_user`

```
application_controller.rb
1 class ApplicationController < ActionController::Base
2   protect_from_forgery with: :exception
3
4   helper_method :current_user
5
6   protected
7   def current_user
8     @current_user ||= User.find(session[:user_id]) if session[:user_id]
9   end
10 end
11
```

Un buen lugar para implementar el método `current_user` es el `application_controller.rb` para que esté disponible en todos los controladores y lo podemos disponer en las vistas agregando el método `helper_method`.

Método current_user en la vista del home#index

```
index.html.erb
1  <%= current_user ? "Hola user: #{current_user.email}" : 'Hola extraño' %>
2  <br>
3
4  <% if current_user %>
5    <%= link_to('Log out', session_path(current_user), method: :delete) %>
6  <% else %>
7    <%= link_to('Log in', new_session_path) %>
8  <% end %>
9
```

Formulario para crear una nueva sesión

```
new.html.erb x
1 <h1>Sign in</h1>
2 <%= flash.now[:notice] if flash.now[:notice] %>
3 <%= form_with url: sessions_path, local:true do %>
4   <div class="field">
5     <%= label_tag :email %>
6     <%= email_field_tag :email,
7       params[:email],
8       placeholder: 'Enter your email address',
9       required: true %>
10   </div>
11   <div class="field">
12     <%= label_tag :password %>
13     <%= password_field_tag :password,
14       params[:password],
15       placeholder: 'Enter your password',
16       required: true %>
17   </div>
18   <%= submit_tag 'Sign in' %>
19 <% end %>
20
```

El controlador de sesiones

```
sessions_controller.rb
1  class SessionsController < ApplicationController
2    def new
3    end
4
5    def create
6      user = User.find_by_email(params[:email])
7      if user && user.authenticate(params[:password])
8        session[:user_id] = user.id
9        redirect_to root_path, notice: 'Logeado correctamente'
10     else
11       flash.now[:notice] = 'Email o password inválida'
12       render action: :new
13     end
14   end
15
16   def destroy
17     session[:user_id] = nil
18     redirect_to root_url, notice: 'Signed out successfully.'
19   end
20 end
21
```

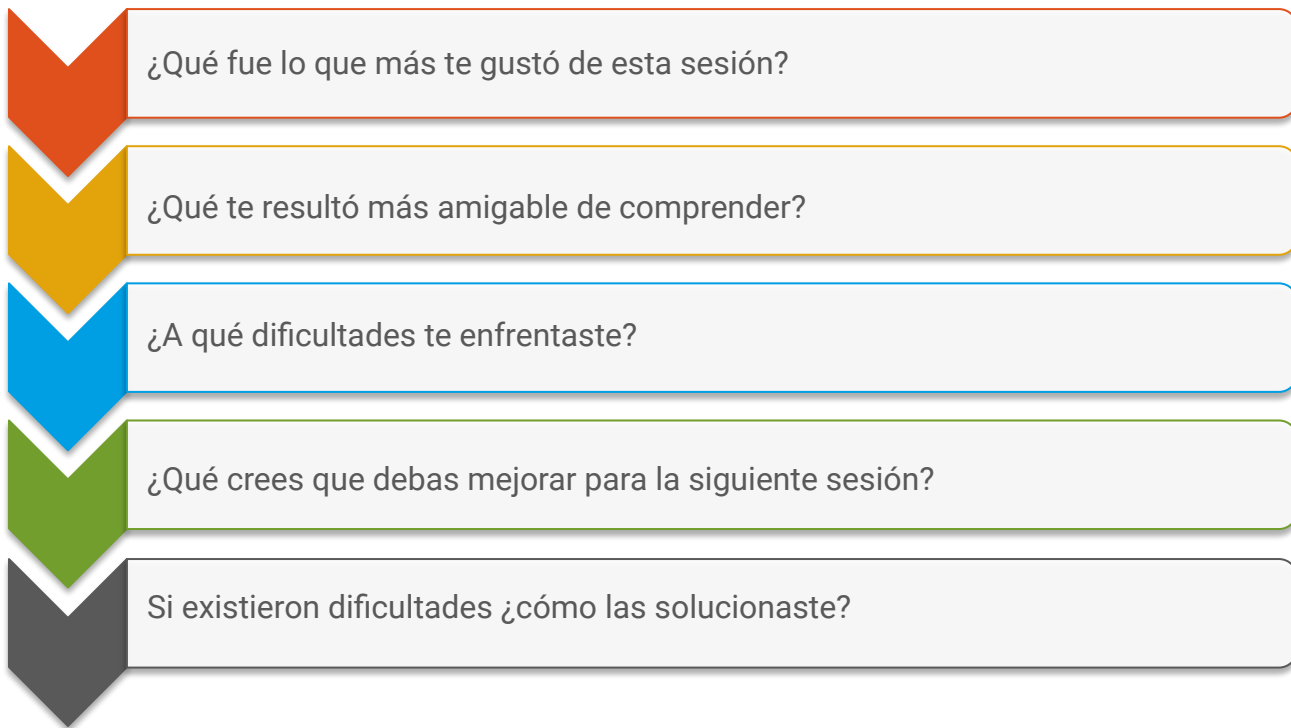


Cierre

{desafío}
latam_



15 minutos



¿Qué fue lo que más te gustó de esta sesión?

¿Qué te resultó más amigable de comprender?

¿A qué dificultades te enfrentaste?

¿Qué crees que debas mejorar para la siguiente sesión?

Si existieron dificultades ¿cómo las solucionaste?



*Academia de
talentos digitales*

www.desafiolatam.com



/DesafioLatam



/DesafioLatam



/DesafioLatam



/DesafioLatam