Clase 11:

Usuarios y sesiones en Ruby on Rails

profesor: **Patricio López Juri** { patricio@lopezjuri.com }
créditos: 10
horario: J:7-8
sala: H3

Supuesto que hago yo en este curso:

Ustedes son personas inteligentes y pueden aprender y deducir reglas lógicas en base a ejemplos.

Repaso

- ¿Cómo se usaba `belongs_to` y `has_many`?

HTTP es *Stateless*, es decir “sin estado”.

Esto significa que el servidor no retiene información de la *request* pasada.

Analogía para entender esto

Comprar en el McDonalds físicamente como si fuéramos “sin estado”.

- Imaginen que van al McDonalds y están haciendo su orden en la caja.
- Supongamos que la acción de pedir la hamburguesa, bebida, etc… cada una es una *request* independiente.
- Para hacer cada *request* debemos salir del local y volver a entrar.

Analogía para entender esto

Comprar en el McDonalds físicamente como si fuéramos “sin estado”:

- Imaginen que van al McDonalds y están haciendo su orden en la caja.
- Supongamos que la acción de pedir la hamburguesa, bebida, etc… cada una es una *request* independiente.
- Para hacer cada *request* debemos salir del local y volver a entrar.

¿Suena medio tonto, o no? ¿No sería mejor simplemente esperar en la caja y sin salir del local?

Analogía, ahora somos Amazon.com

Comprando en Amazon.com, donde usamos la web (HTTP) y sí es “sin estado”:

- La acción de vitrinear, “agregar al carro” y comprar son *requests* HTTP.
- Amazon tiene millones de *requests* cada segundo.
- Pueden pasar días entre cada acción HTTP.
    - Ej: agregamos muchas cosas al carro durante la semana y hacemos la compra el domingo.

Analogía, ahora somos Amazon.com

Comprando en Amazon.com, donde usamos la web (HTTP) y sí es “sin estado”:

- La acción de vitrinear, “agregar al carro” y comprar son *requests* HTTP.
- Amazon tiene millones de *requests* cada segundo.
- Pueden pasar días entre cada acción HTTP.
    - Ej: agregamos muchas cosas al carro durante la semana y hacemos la compra el domingo.

Ya no parece buena idea quedarnos en la caja porque podemos hacer una cola inmensa.

Entonces en el caso del McDonalds si fuera state-less,

¿Cómo nos pueden reconocer entre cada visita?

(cómo pueden saber qué estábamos pidiendo antes)

Las *cookies* sirven para identificarnos entre cada request

- Las *cookies* suelen ser un *string* de números y letras aleatorios que el servidor usa para identificarnos.
- Con esto podemos llevar un “estado” entre distintas peticiones
- ¿Qué pasa si alguien nos roba las *cookies*?
- ¿Qué pasa si “borramos las *cookies*”?

¿Qué pasa si usamos otro computador?

Ya no tenemos las *cookies* del otro dispositivo. Además estas pueden “vencer” y dejan de ser válidas.

¿Cómo hacemos perdurar datos de la sesión a lo largo del tiempo?

Cuentas de usuario

Debemos permitir a los usuarios poder crear un perfil permanente en nuestra aplicación web.

Este perfil debe:

- Poder ser accedido solo por la persona que conozca su nombre o correo público y la contraseña privada.
- Poder ser accedido en distintos computadores.
- Poder recuperar la cuenta si se pierden u olvidan las credenciales.

Pero si HTTP es *stateless*,
cómo hacemos para implementar sesiones de usuario?

Podemos mandar el usuario y contraseña en cada *request* HTTP, ¿o no?

Mandar las credenciales una y otra vez es inseguro.

Lo correcto sería:

1. Ingresar las credenciales válidas.
2. Recibir una *cookie* a cambio.
3. Utilizar esta *cookie* hasta que caduque.
4. Volver al paso 1.

¿y si quiero cerrar una sesión?

¿y si quiero cerrar una sesión?

Le decimos al server que la *cookie* ya no es válida y la destruímos.

¿Cómo hacemos todo esto en Ruby on Rails?

¿Cómo hacemos todo esto en Ruby on Rails?

Podemos hacer todo manualmente con *cookies* y todo, pero en Rails existen gemas que nos ahorran trabajo!

Gema Devise

La gema más conocida para ayudarnos en esto es Devise:

https://github.com/plataformatec/devise

```
# en Gemfile
gem 'devise'

# en la terminal
bundle install
rails generate devise:install

# en config/environments/development.rb
config.action_mailer.default_url_options = { host: 'proyecto-rails2-lopezjuri.c9users.io', port: 3000 }

# en config/routes.rb
root to: "welcome#index"

# en app/views/layouts/application.html.erb
<p class="alert"><%= alert %></p>

# en la terminal
rails generate devise User
rake db:migrate
```

Ver proyecto en Github

https://github.com/INP3440-2017-2/proyecto-rails/commits/master

Desde "`Add devise`" hacia arriba.