IIC2513 Tecnología y Aplicaciones de la WWW
Semestre 1-2015 Prof. J. Navón
Solución Interrogación 2

Problema 1
Manejo de sesión

a) En la aplicación Depot que hemos estudiando en clases explique la estrategia que se usa para "recordar" los productos que se han comprado

- se crea en el modelo un carro que tiene los lineItems
- el id del carro se guarda en el hash de sesión
- al agregar un producto, si la sesión tiene un carro se recupera el id y se agrega a ese carro; en caso contrario se crea un carro y se guarda el id en la sesión

b) ¿Que relación existe entre el objeto o hash de sesión de Rails y las cookies ? Describa la situación si en un instante la sesión contiene lo siguiente {:code =>"0132", :description = >"hammer"}

- Existe una cookie con el id de la sesión y una cookie por cada item guardado en el hash En el caso dado hay una cookie con la sesión, una con el código y una con la descripción. Rails puede operar con el estado en el server (se usa solo el cookie de sesión) o con el estado en el cliente (se usan todas las cookies)

c) En la solución vista en clases si dejo libros en el carrito y vuelvo después de algunos días al sitio, ¿va a "recordar" lo que tenía en el carrito o no? Justifique.

- No, no lo va a recordar porque el id del carro que está en la sesión se pierde. El objeto carro con sus items queda en la BD pero se pierde la asociación con el usuario.

Los problemas 2 al 4 de esta interrogación se basan en una aplicación similar a un Twitter (simplificada) en que el modelo solo incluye: users(login, name, number_of _tweets) y tweets(date, status). El atributo number_of_tweets mantiene la cuenta del número de mensajes posteados por el usuario. El atributo status contiene el cuerpo del mensaje (140 caracteres)

Problema 2
Generación del Modelo mediante Scafoldings

a) Escribir un comando rails que permita generar automáticamente un modelo para el usuario con atributos: login, nombre, número de tweets y además la conexión con sus tweets

- rails generate model User login: string name: string, number_of_tweets: integer tweets: has_many

b) Escribir un comando rails que permita generar automáticamente un modelo para los tweets con atributos: status y date pero además la conexión con su autor

- rails generate model Tweet date: date status: string user: belongs_to

c) Explique la forma mas sencilla de lograr que al eliminar un usuario se eliminen automáticamente sus tweets

- en modelo de User, cambiar *has_many :tweets* por *has_many :tweets, dependent: :destroy*

d) Explique de que forma podemos lograr que no se pueda eliminar un usuario si tiene a lo menos un tweet

- agregar una entrada before_destroy :ensure_can_do en el modelo de User. El método (privado) ensure_can_do devuelve true si no hay tweets para el usuario (tweets.empty?)

Problema 3

Controlador, Vistas, Layouts

a) Escriba el método show del controlador de tweets que recibe como parámetro el id de un tweet y carga la variable de instancia @tweet con el tweet correspondiente a ese id y finalmente invoca la vista en /app/views/tweets/status.html.erb

```
class TweetsController < ApplicationController
 def show
  @tweet = Tweet.find(params[:id])
  render action: 'status'
 end
end
```

b) Escriba el código de la vista status.html.erb que solo incluye un título de primer nivel (H2) con el contenido del tweet (status) y a continuación un párrafo (p) que diga "publicado por <nombre del autor del tweet> el <fecha del tweet>" (ver figura)


```
<h2><%= @tweet.status %></h2>
<p>publicado por <%= @tweet.user.name %> el  <%= @tweet.date %></p>
```

c) Escriba ahora el contenido del layout que sirve de marco para toda la aplicación que incluye un título, un logo del sitio (logo.png en assets/images) un espacio reservado para lo que generan las vistas y un footer que debe aparecer al final de la página. Indique también el nombre (path completo) de este archivo. Escriba además el archivo de estilos necesario y referéncielo desde el archivo de layout para producir el resultado de la figura.

Nombre del Archivo: app/views/layouts/application.html.erb


```
<html>
  <head><title>A better twitter</title>
    <%= stylesheet_link_tag    "application" %>
  </head>
  <body>
    <div id="container">
      <div id="header">
        <%= image_tag("logo.png") %>
        <h1>IIC2513 Tecnología y Aplicaciones de la WWW</h1>
      </div>
      <div id="body">
        <%= yield %>
      </div>
      <div id="footer">
        <h3>Copyright IIC2513 </h3>
      </div>
    </div>
</html>
```

Nombre de Archivo de Estilos: ../stylesheets/application.css.scss

```
body {
   height:100%;
}
img {float:right; width:80px;}

#container {
   min-height:100%;
   position:relative;
}
#header {
   background:#cfcfcf;
   padding:10px;
}
#body {
   padding:10px;
```

```
  padding-bottom:60px;
}
#footer {
  position:absolute;
  bottom:0;
  width:100%;
  height:60px;
  background:#cfcfcf;
}
```

Problema 4
Se pide escribir tests en RSpec para verificar la validación incorporada al modelo de la aplicación. Los tests a escribir son los siguientes:

a) El número de tweets asociado a un nuevo usuario debe ser cero

```
describe User do
 it "has no tweets if new user´"do
  user = User.new(login: "jnavon", name:"Jaime Navon");
  user.number_of_tweets.should == 0
 end
end
```

b) Debe verificarse que un tweet creado con "bla, bla, bla, web, bla, bla, bla" incluye la palabra "web"

```
describe Tweet do
 it "has a status that matches 'web´"do
  tweet = Tweet.new(status: " bla, bla, bla, web, bla, bla, bla ")
  tweet.name.should match(/web/)
 end
end
```

c) El número de usuarios debe aumentar al crear y guardar un nuevo usuario

```
describe User do
 it "increments number of users"do
   user = User.new(login: "jnavon", name:"Jaime Navon");
   expect { user.save }.to change {User.count ] by(1)
 end
end
```

d) El modelo debe tener el método que permite acceder al status

```
describe Tweet do
 it 'is invalid without a status do
  tweet = Tweet.new
  tweet.should_not be_valid
 end
end
```