Tarea 2 - Sesión # 2
Bases de Datos

Andrea Figueroa R.
abfiguer@alumnos.inf.utfsm.cl

Camilo Rivas F.
camilo.rivas@alumnos.usm.cl

Camilo Valenzuela C.
camilo.valenzuela@alumnos.usm.cl

Javier Jeria M.
javier.jeria.13@sansano.usm.cl

Cecilia Reyes C.
reyes@inf.utfsm.cl

30 de mayo de 2016

Preámbulo

Repaso Git (control de versiones)

Para ir subiendo los cambios realizados a su trabajo los comandos a usar son:

- Para traer los nuevos cambios del proyecto: `git pull`
- Para agregar los nuevos archivos y cambios del repositorio: `git add .`

  Con el '.' le decimos a git que busque los cambios en esa carpeta y en todas las subcarpetas. También se puede hacer git add ¡nombre de archivo¿para agregar los cambios sólo de un archivo.
- Para guardar los cambios realizados en el proyecto: `git commit -am` "Descripción"

  Para mayor legibilidad del repositorio tratar de que la descripción del comentario sea significativo, y no poner palabras al azar.
- Luego del commit es necesario un `git push` para empujar los cambios al servidor.
- **De no haber terminado la sesión uno correctamente se sugiere utilizar la base proporcionada por los ayudantes.**
  Para esto clonar el proyecto: `git@gitlab.labcomp.cl:cvalenzu/Tarea2-ayubd.git` y copiar los archivos necesarios dentro de su carpeta de git, debe seguir utilizando un proyecto de git propio, no trabajar sobre éste.

Para más información dirigirse a documentación[1].

Links útiles

- Getting started RoR:
  `http://guides.rubyonrails.org/getting_started.html`
- Modelos:
  `http://guides.rubyonrails.org/active_record_basics.html`

[1] https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html

- Insertar registros:
  `http://guides.rubyonrails.org/active_record_basics.html#create`
- Listar registros:
  `http://guides.rubyonrails.org/getting_started.html#listing-all-articles`
- Curso interactivo de Git:
  `https://www.codeschool.com/learn/git`
- Curso interactivo de Ruby y RoR:
  `https://www.codeschool.com/learn/ruby`
- Tutorial básico **Devise** en español:
  `http://www.peoplecancode.com/es/paths/4/tutorials/13`
- Agregar campos a **Usuario Devise**:
  `http://www.peoplecancode.com/es/tutorials/adding-custom-fields-to-devise`

Dentro del Proyecto RoR

Archivos importantes en el directorio de Rails

- **/app/models**: Acá están los modelos, generalmente un archivo por cada modelo.
- **/app/controllers**: Acá están los controladores, hay al menos uno por cada vista, y pueden existir otros no relacionados a un modelo.
- **/app/views**: Acá están todas las vistas, por cada controlador hay una carpeta, dentro de éstas están las vistas asociadas a ese controlador.
- **/config/routes.rb**: Acá se definirán las rutas.
- **/db/migrations**: En esta carpeta se encuentran las migraciones creadas tanto por la generación de un modelo como las generadas manualmente. Pueden ser editadas, pero si se ejecuta el comando rake db:migrate", las modificaciones que se le realicen a estos archivos no tendrán efecto sobre la base de datos.
- **/db/seed.rb**: Este archivo es útil para poblar inicialmente la base de datos, se puede realizar un script insertando datos y relaciones, para luego correr el comando rake db:seed".

  `http://www.xyzpub.com/en/ruby-on-rails/3.2/seed_rb.html`

¿Qué se hará ?

En esta sesión se verá el concepto de Gemas de Ruby, que son y para que sirven, luego se utilizará como ejemplo la Gema Devise, para crear los distintos Usuarios que utilizarán la plataforma que se esta creando.

Gemas de Ruby

Una **gema** es, básicamente, la manera en que Ruby permite distribuir programas, módulos o librerías que extienden funcionalidad, casi siempre específica, y que hacen **no tener que repetirnos (Don't Repeat Yourself)** volviendo más fácil nuestro flujo de desarrollo. Por ejemplo, hay gemas para Rails que se encargan de la autenticación de usuarios como es el caso de Devise, o Carrierwave, una gema que se encarga de subidas de archivos.

Para casi cada acción que se realiza repetitivamente a la hora de programar, existe una gema que lo vuelve todo más fácil.

Cabe decir que las gemas pueden depender de otras gemas para funcionar.

Devise

Como se mencionó anteriormente **Devise** es una gema de Ruby on Rails que se encarga de la autenticación de usuarios, en pocos pasos se puede tener listo el sistema de registro, login y manejo de sesiones para distintos usuarios.

Luego de instalar y configurar la gema se pueden crear modelos de Devise, que vienen por defecto con correo y contraseña entre otros atributos, cada modelo de Devise va a tener sus propias rutas, pero comparten la misma vista, por lo que si se desea tener Usuarios distintos, se tendrá que editar las rutas y vistas que vienen por defecto.

Rutas de Devise

Devise crea varias rutas por defecto, conocerlas sirve para agregar carácteristicas a nuestro sitio, como por ejemplo agregar un campo de login al menu del sitio.

A continuación se detallarán algunas de las rutas que utiliza devise

Método	Ruta	Controlador#Método	Descripción
GET	`/<modelo>/sign_in`	devise/sessions#new	Muestra la vista para el login
POST	`/<modelo>/sign_in`	devise/sessions#create	Realiza lógica del login
DELETE	`/<modelo>/sign_out`	devise/sessions#destroy	Realiza lógica de logout
POST	`/<modelo>s`	devise/registrations#create	Crea nuevo registro de usuario
GET	`/<modelo>/sign_up`	devise/registrations#new	Muestra la vista de registro

Objetivos de aprendizaje

- Entender utilidad de gemas y aplicar el conocimiento con gema devise.
- Aprender a realizar validaciones

Objetivos Practicos

- Instalar y configurar devise[2] **[10 pts]**
- Crear modelo de devise de Usuario y Administrador de tienda[3] **[20 pts]**
- Crear vista 'Dashboard para Usuario' y 'Menú Administrador' **[20 pts]**
  - **Menú de Administrador de Tienda**
    El Administrador de Tienda debe poder:
    1. Crear una nueva tienda que él pueda administrar.
    2. Agregar productos a una tienda que él administre.
  - **Dashboard para Usuario** Un usuario puede:
    1. Un mensaje de bienvenida al mall.
    2. Ver un listado de las tiendas disponibles.

  No se espera que implemente las funcionalidades del Dashboard de Administrador, solo que deje lista la vista para poder agregar estos requerimientos más adelante.
- Cambiar la ruta a la cual se redirecciona al Usuario y administrador una vez logueado **[10 pts]**
- Registrar un Usuario y Administrador **[10 pts]**
- Validar que sólo un administrador logeado pueda ver el menu de administrador **[10 pts]**
- Commit Final y uso de git durante desarrollo **[20 pts]**

[2] `https://github.com/plataformatec/devise#getting-started`

[3] Considerar que hay dos formas de crear el rol de Admin, más información en `https://github.com/plataformatec/devise/wiki/How-To:-Add-an-Admin-Role`

Recomendación de trabajo en equipo

- Cada integrante del grupo preocúpese de una entidad para el desarrollo de la sesión. Es decir, uno enfóquese en **Administrador** y el otro en **Usuario**. Decidir que enfoque de los dos recomendados se usará[4]
- Para ingresar los registros, cada uno puede ingresarlos independiente del otro.
- Recuerde ser ordenado al momento de hacer commit de los cambios.
- Recuerde además, que la instalación de las gemas se hace de forma local, es decir, cada miembro del equipo deberá instalar el bundle luego de agregada la gema devise.

[4] `https://github.com/plataformatec/devise/wiki/How-To:-Add-an-Admin-Role`