Tarea 2 - Primera sesión
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

24 de mayo de 2016

Objetivos de aprendizaje

- **Objetivo mayor**: Primer acercamiento a MVC.
- Entender generalmente organización de archivos
- Comenzar a trabajar con MVC
- Utilización de linea de comandos
- Entender migraciones

Objetivos Practicos

- Creación e inicialización de repositorio Git [**10 pts**]
- Creación de Proyecto rails
- Subir el proyecto al git para que ambos puedan trabajar en él.
- Iniciar el servidor y acceder a través del navegador.
- Crear modelo de tienda: La tienda debe tener **Nombre** y **Descripción**. [**10 pts**]
- Crear modelo de producto: El producto debe tener **Nombre**, **Descripción**, **Precio de venta** y **stock**.[**10 pts**]
- Correr migración.
- Ingresar registros a los modelos recientemente creados utilizando la consola, al menos 5 por cada modelo.[**10 pts**][1]
- Crear un controlador y una vista por cada modelo. [**10 pts**]
- Obtener los registros del modelo en su controlador para poder mostrarlos en la vista.[**10 pts**][2]

[1] http://guides.rubyonrails.org/active_record_basics.html#create
[2] http://guides.rubyonrails.org/getting_started.html#listing-all-articles

- Modificar las rutas para poder acceder a las vistas.[**10 pts**]
- Mostrar los registros de cada uno de los modelos en diferentes vistas. [**20 pts**]
- Commit final. [**10 pts**]

Recomendación de trabajo en equipo

- Mientras un integrante crea el repositorio GIT, el otro crea el proyecto de RoR
- Cada integrante del grupo preocúpese de una entidad para el desarrollo de la sesión. Es decir, uno enfóquese en la creación del modelo **Tienda** y otro con el modelo **Producto**.
- Para ingresar los registros, cada uno puede ingresarlos independiente del otro.
- Recuerde ser ordenado al momento de hacer commit de los cambios. Si se encuentra con un merge problem, la consola le dirá claramente donde está el problema. Póngase de acuerdo con su compañero para solucionar el problema de ser necesario.

¿Qué se hará ?

Inicialmente se crearán dos modelos de la tarea, que son la tienda y el producto. Luego de crearlos, se procederán a crear vistas asociadas a estos modelos para poder verlos dentro del navegador. Este paso incluye la interacción modelo-vista-controlador en su forma básica 1.

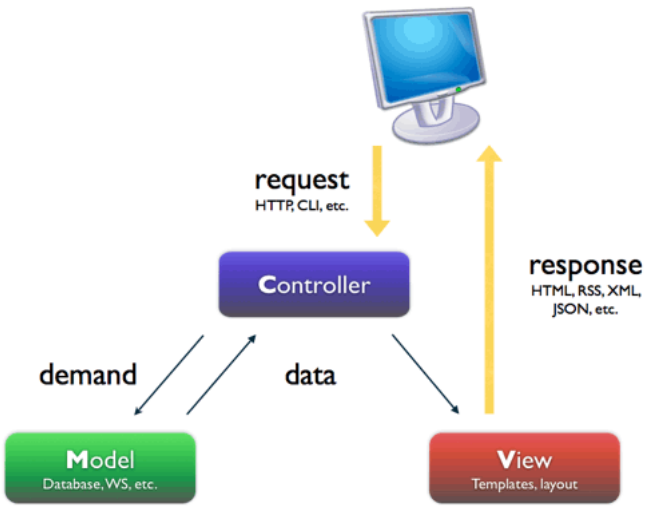

Figura 1: Esquema de MVC.

El procedimiento que hay que entender es: El usuario pide al navegador encontrar cierta página, este request llega al controlador, quién consulta al modelo los datos necesarios para mostrar una vista. El modelo retorna al controlador la información necesaria. El controlador, por último, envía la información a mostrar a la vista, que luego es entregada al usuario.

Un elemento importante a destacar es que un controlador puede estar asociado a múltiples vistas. Generalmente, se hace un grupo de vistas por cada modelo. Por ejemplo, se tiene el modelo **Usuario**, y el controlador se encarga de las vistas **Editar Usuario**, **Lista de Usuarios**, etc., ya que todas esas vistas consultan el mismo modelo.

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.

Comandos útiles para la sesión

- Crear un proyecto RoR

  ```
  rails new NombreProyecto
  ```

  Después de crear el proyecto editar el archivo `Gemfile`, luego deben descomentar la linea `gem 'therubyracer'`, luego reinstalar las gemas, proceso que se debe efectuar cada vez que se edite este archivo en cada uno de los computadores donde se quiera correr el servidor.

- Instalar gemas

  ```
  bundle install
  ```

  Este comando se debe utilizar cada vez que se edite el archivo de gemas, también cuando se baje el proyecto desde cero, en cada uno de los computadores que se utilice.

- Iniciar el servidor

  ```
  rails server
  ```

  Luego de iniciado pueden acceder a la página en su navegador a través de `http://localhost:3000`

- Crear un modelo

  ```
  rails g model Modelo atributo1:tipo1 atributo2:tipo2 ...
  ```

- Ingresar a la consola

  ```
  rails console
  ```

- Crear un controlador y sus vistas asociadas:

  ```
  rails g controller NombreControlador vista1 vista2 ...
  ```

- Crear una nueva migración

```
rails g migration NombreDeLaMigración
```

- Correr migraciones

```
rake db:migrate
```

  Este comando se debe utilizar cada vez que se cree, edite o borre un modelo, también cuando se baje el proyecto desde cero, en cada uno de los computadores que se utilice.

- Consola Rails

```
rails c
```

  Optionalmente, se puede correr con la opción '–sandbox', donde los cambios efectuados no se verán reflejados en la base de datos.

Creación de git (control de versiones)

Para poder manejar su proyecto se utilizará Git[3], para crear e inicializar su proyecto debe seguir los siguientes pasos:

- Ingresar a la plataforma de git otorgada en el departamento de informática `https://gitlab.labcomp.cl` e ingresar con su cuenta de informática.
- Crear un nuevo proyecto presionando el botón superior derecho **New Project**, rellenar los datos, dejandolo privado.
- Luego en el menú a la izquierda ingresar a **Configuración**, de ahí en el menú a la izquierda ingresar a **Members**.
- Agregar a su compañero de tarea en la opción **Add members**, luego ingresar el nombre y en *Project Access* darle la opción **Master**. Repetir para agregar a su ayudante corrector.
- Ahora deben crear sus llaves, **esto lo deben hacer todos los integrantes del grupo**, también es necesario repetirlo nuevamente cuando se quiera acceder desde otro computador. Para crear su llave personal, en la terminal ingresar el comando: `ssh-keygen`, les pedirá el directorio a guardar la llave, presionar enter para el determinado, luego le pedirá una *passphrase* la cual es una clave personal preferentemente distinta a su clave normal de usuario.
- Una vez creada la llave, con el comando `cat .ssh/id_rsa.pub` podrán ver la llave, todo este texto lo deben agregar como su llave en `https://gitlab.labcomp.cl/profile/keys` presionando **Add SSH key**.

[3]Software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

- Una vez agregado su compañero y las llaves de ambos acceder a la terminal para la inicialización del git, para eso ingresar los siguientes comandos, **esto es necesario hacerlo una sola vez.** Reemplazar **USERNAME** con su nombre de usuario del DI y **NOMBRE_REPO** con el nombre de su repositorio.
  - `mkdir test`
  - `cd test`
  - `git init`
  - `touch README.md`
  - `git add README.md`
  - `git commit -m "first commit"`
  - `git remote add origin git@gitlab.labcomp.cl:`**USERNAME/NOMBRE_REPO**.git
  - `git push -u origin master`
- Ahora usted o su compañero pueden clonar el repositorio con el comando: `git clone` **URL_GIT**
- 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: `git add .`
  - Para subir los cambios realizados en el proyecto: `git commit -am` "Descripción"
  - Luego del commit es necesario un `git push` para empujar los cambios al servidor.
- Para más información dirigirse a documentación[4].

Links útiles

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

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