Aplicación para la calidad de anunciantes

Proyecto Final. Ruby on Rails

Sara Cortés Carnero
Ana de la Cuesta Torre
Francisco Manuel Pérez Frías
Marino Tejedor González

Repositorio GitHub: https://github.com/equipo5RoR/AppAnunciantes

Marzo 2016

1. DESCRIPCIÓN Y OBJETIVOS

Desde la empresa “BeRuby” solicitan una aplicación que sea capaz de leer una base de datos proporcionada, de forma que extraiga información de reclamaciones. Dicha información debe procesarse y utilizarse o bien en una página web o bien en un informe que se envía por email.

Para ello se fijan los siguientes objetivos:

- Leer de una base de datos ya existente
- Mandar emails semanales
- Calcular y almacenar contenido con tareas en background
- Montar una vista web con un look & feel similar al de la ficha de anunciante.

2. CONEXIÓN CON UNA BASE DE DATOS EXTERNA

La base de datos con la que se trabaja está en formato MySQL, y contiene tres tablas: una tabla de anunciantes, otra con las comisiones de cada venta y una última con las reclamaciones. Para la conexión hay que instalar la gema mysql2 y configurar los parámetros necesarios para la conexión con ella, estos son: adaptador, codificación, nombre de la base de datos, usuario y contraseña de acceso, el protocolo de red y la conexión. Tras esta configuración ya se puede conectar con la base de datos, por lo que se crea el campo “puntuación” dentro de la tabla de anunciantes.

Para poder extraer los datos de dicha base de datos, se utiliza la clase ActiveRecord que viene definida en Rails. A través de ella se crean los modelos, uno por cada tabla, con el mismo nombre pero en singular de la tabla de la base de datos. Tras crear los modelos se definen las relaciones entre ellos y ya se puede acceder a los datos de la base a través de la aplicación.

3. CREACIÓN DEL ALGORITMO

La puntuación de cada anunciante depende de unos criterios establecidos en la toma de decisiones con el cliente. Estos son: fecha del último reembolso, fecha del último pago, admisión de reclamaciones, porcentaje de reclamaciones sobre el total de ventas y porcentaje de reclamaciones aceptadas sobre el total de reclamaciones.

Para otorgar la puntuación hay que recorrer las tablas buscando los datos necesarios para posteriormente realizar el cálculo. Para recorrer las tablas se han utilizado las funciones SQL que incluye la clase ActiveRecord, lo que hace que las búsquedas sean más rápidas y con un código más limpio. Por ejemplo: para obtener el porcentaje de reclamaciones sobre el total de ventas hay que recorrer primero la tabla de reclamaciones extrayendo las de un mismo anunciante, también hay que recorrer la tabla de comisiones obteniendo las del mismo anunciante y, finalmente, calcular el porcentaje entre ellas.

```
total_claims = UserClaim.where("advertiser_id=?",advertiser_id).count
total_sales = UserCommission.where("advertiser_id=?",advertiser_id).count
claims_percent = (total_claims.to_f / total_sales.to_f * 100)
```

Finalmente se calcula la puntuación a partir de los cinco valores obtenidos.

4. ENVÍO DEL EMAIL

Para el envío del mail se va a utilizar la clase ActionMailer definida en Rails. Primero hay que configurar los datos del servidor de correo que se vaya a utilizar, en este caso Gmail. Estos son: dirección, puerto, dominio, usuario y contraseña de la cuenta remitente y la autenticación. Tras esto se genera el modelo de mail, en el que se define la cuenta remitente y el método para el envío del mail, a partir de la sentencia "**`mail :to =>`**". Por otro lado, se crea el template html del contenido del mail y se define el método que que lanzará el envío adjuntando el archivo de resumen de las puntuaciones. Este archivo debe ir en formato csv, por lo que utilizaremos la clase CSV definida en Rails para convertir la tabla de anunciantes, en la que se incluye la puntuación, y adjuntarla al email:

```
csv = Advertiser.to_csv

attachments["resumen.csv"] = csv

ResumeMail.resume_email(email,csv).deliver_now
```

5. CREACIÓN DE TAREAS

Para la actualización de la puntuación y el envío del mail se definen dos tares que lancen dichas funciones de manera automática y que se ejecuten en segundo plano. Para esto es necesario instalar las gemas "whenever" y "crontab", que se encargan de la automatización de tareas. Primero hay que configurar el proyecto, ejecutando el comando **wheneverize**, que crea un archivo sobre el que se definen los tiempos de ejecución de las tareas (schedule.rb). Por otro lado se crea un archivo **.rake** en el que se definen las tareas, una con la llamada al método que actualiza la puntuación y otra con la llamada al método que envía el mail. Por último hay que transformar el archivo Schedule a crontab, gracias a la gema, ya que sino las tareas no serán lanzadas en los periodos que se hayan escrito en dicho archivo.

6. CREACIÓN DE LAS VISTAS HTML

Para la creación de las vistas se ha utilizado el framework "Bootstrap", que permite crear interfaz web responsive con CSS y Javascript. Este framework es muy útil cuando se quiere crear páginas web que sean responsive en todos los tamaños de dispositivos.

Gracias al controlador se accede a la tabla de anunciantes para obtener tanto los detalles del anunciante como las puntuaciones. Estos detalles se mandan a la vista a través de una variable de instancia, y se muestran a partir de estrellas para que sea más visual.

Por otra parte se generan los iconos de cada anunciante gracias a un script que crea un icono con la inicial de cada anunciante, y que se implementa al algoritmo a través de un nuevo modelo y un nuevo controlador.

7. CONCLUSIONES

El principal problema que se encontró fue la refactorización del código, ya que el primer algoritmo estaba escrito sobre el controlador y sin utilizar los métodos de ActiveRecord. La diferencia es sustancial, ya que sin la refactorización el método tenía un tiempo de ejecución de 10 minutos que se redujo a menos de 1 minuto tras la refactorización.

Afrontar el proyecto en equipo ha sido determinante tanto para la resolución del proyecto como para nuestro crecimiento como programadores. En este proceso hemos aprendido los unos de los otros.