PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

ESCUELA DE INGENIERÍA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IIC-2113 DISEÑO DETALLADO DE SOFTWARE

Entrega-3

Diseño Detallado del Software

30 de Octubre de 2014

Tomás Gunther

Pablo Messina

Cambios en el diseño

Básicamente el esqueleto general del modelo se mantuvo muy parecido, excepto que se eliminaron una clases (que pasaron a formar parte del frontend) estas son las clases que manejan la sesión del usuario: session_controller, session_facebook_controller y session_normal_controller. La relación entre las clases se mantuvo, por lo que el acoplamiento no aumento y la cohesión se mantuvo en el mismo nivel. Sin embargo fue necesario recurrir a la creación de nuevos métodos en las clases User y Mashup. Esto fue debido a que hubo ciertas cosas que no se previeron y para facilitar la creación del código se hicieron unos métodos para reutilizarlo. Por ejemplo se hizo un metodo para resetear el mashup temporal de un usuario, también se tuvo que crear un método para clonar un mashup, el cual se usa para pasar de un mashup temporal a uno definitivo.

NOTA:La implementación de user_facebook_controller no se realizó y se dejará para la próxima entrega, a no ser que se decida que no es necesaria su implementación.

Los cambios (reflejados en el diagrama UML adjunto) son:

Mashup_controller.rb: index_total y new

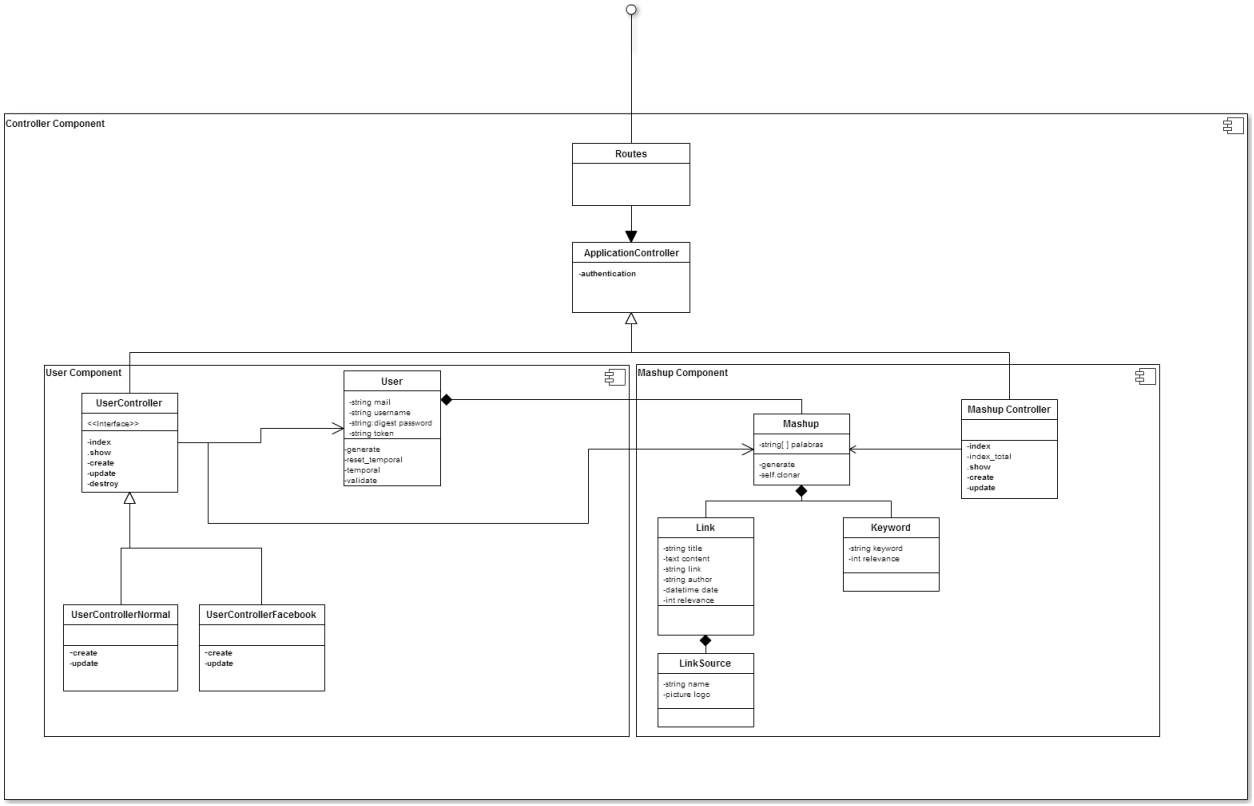
User.rb: reset_temporal, regenerate_temporal, temporal= y temporal

Mashup.rb: generate y clonar

User_facebook_controller: no implementado

Session_controller, session_normal_controller y session_facebook_controller: se borran.

Diagrama UML (Actualizado)



Interpretación de los resultados de Metric_Fu

Para poder ver los puntos vulnerables del código, se utilizaron las métricas de la gema Metric_Fu. La métricas que destacamos son las siguientes:

Cane:
Existe violación del umbral de calidad de código en los métodos MashupsController#update y ApplicationController#authenticate, lo cual es consistente con el hecho de que estos dos métodos son los más complicados en cuanto a la cantidad y forma del código. Probablemente sea necesario hacer alguna refactorización para simplificarlos y hacer el código más mantenible.

Churn:
Básicamente arrojó valores parejos para los números de modificaciones de archivos que suelen ser modificados, como routes.rb, Gemfile, Gemfile.lock, mashups_controller.rb y user.rb. Los primeros 3 son archivos que suelen ser modificados al trabajar con rails, y los últimos dos son archivos de componentes críticos del software y por lo tanto es comprensible que estén dentro de los más modificados.

Flay:
Las similaridades estructurales encontradas son mínimas. Esto es un punto favorable del código, ya que no se están copiando códigos, sino que se están utilizando los métodos de una forma adecuada.

Flog:
Los 3 archivos con mayores valores de complejidad promedio fueron application_controller.rb (29.8), mashup.rb (16.9) y user.rb (13.2), valores relativamente bajos para ser los más altos de la aplicación. Esto nos muestra que la complejidad del código es más bien baja.

Stats:
Líneas de código: 585.
Líneas de test: 214.
Razón código a test: 1 : 0.4

Saikuro:
Las clases con mayores complejidad ciclomática son Mashup (27), ApplicationController (18) y User (12). Esto significa que sólo la primera se considera una clase compleja, de mayor riesgo, mientras que las otras 2 sólo son de complejidad y riesgo moderados. Esto consistente con el hecho de que Mashup es la clase más crítica de la aplicación, y las otras dos clases mencionadas también desempeñan roles importantes en la aplicación. Además, los valores de complejidad ciclomática de la aplicación en general resultaron bastante bajos.

Reek:
Se puede resaltar que dentro de los mayores code smells detectados destacan UnusedParameteres (61) y IrreponsibleModule (38). Quizás se deba realizar un estudio de qué parámetros no se ocupan y realizar una refactorización del código. Además sería util echarle una mirada a los métodos que tienen modulos irresponsables y ver si esto podría provocar un problema en el futuro.

NOTA: los test no funcionan (debido a las gemas que hay en el proyecto). No pudimos hacerlos funcionar y se nos acabo el tiempo. Dejamos un par de test de ejemplos en la rama controller_master.