PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

ESCUELA DE INGENIERÍA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IIC-2113 DISEÑO DETALLADO DE SOFTWARE

# Entrega-2

## Diseño Detallado del Software

26 de Septiembre del 2014

Tomás Gunther

Pablo Messina

# Definición

Dentro de la aplicación backend, nuestro grupo se encargó de la definición del componente de manejo de usuarios y mashups. Para manejar esto se hizo un componente llamado Controlador y este a su vez esta compuesto por 2 componentes: User y Mashup.

La mayor responsabilidad de este componente es administrador de la aplicación. Se encarga de recibir y procesar las solicitudes del frontend. Para procesar las solicitudes, este componente debe llamar a los componentes de Parser y de IA, los cuales extraerán los datos necesarios para armar el Mashup.Otra labor importante es toda la administración de los usuarios. Esta consiste en el CRUD de los usuarios, a través de Facebook y de una cuenta local.

Una decisión importante de diseño que se tomó es que para poder generar un mashup el usuario debe tener una cuenta creada en la aplicación, ya sea una cuenta local normal o una cuenta en Facebook.

A continuación se realizará un análisis, en base a los 4 principios fundamentales:

**Abstracción**: la abstracción está presente en la definición de las clases UserController y SessionController, las cuales definen métodos estandarizados que posteriormente son implementados de manera concreta por las clases UserControllerNormal y UserControllerFacebook en el primer caso, y SessionControllerNormal y SessionControllerFacebook en el segundo caso. La razón de ser de esta decisión en el diseño radica en que de esta manera se puede obtener diferentes comportamientos para las mismas firmas/métodos dependiendo de la necesidad de la situación concreta particular.

**Ocultamiento**: el ocultamiento de este componente es el adecuado, ya que para el resto de los componentes es transparente el manejo de la información. Cuando la vista solicita un mashup, este componente no tiene idea de como se guarda la información, ni qué parámetros se utilizan. Lo único que necesita saber es que con cierta ruta recibe un mashup con los datos necesarios para que puedan consumirlo.

**Cohesión**: las clases fueron agrupadas buscando maximizar la cohesión entre ellas, y para esto se usó como criterio la consistencia entre el nombre de las componentes y el propósito funcional de las clases encapsuladas por dichas componentes. Así se tiene que User Component incluye todas las clases de control de solicitudes y representación en la base de datos de los usuarios así como las clases que manejan y registran las sesiones que los usuarios pueden abrir durante el uso de la aplicación. Mashup Component, por su parte, incluye las clases que permiten controlar solicitudes relacionadas directamente con los mashups así como las clases necesarias para darles persistencia a los mashups en la base de datos.

**Acoplamiento**: existe acoplamiento entre las componentes User y Mashup debido a que un usuario (representado por la clase User) posee potencialmente muchos mashups, por lo que UserController necesita recurrir a estos mashups en respuesta a consultas relacionadas con los usuarios. Sin embargo, este acoplamiento es necesario por la naturaleza misma de la aplicación y es el mínimo posible, ya que todas las demás clases están debidamente agrupadas en sus respectivas componentes con el fin de mantener la cohesión interna de las mismas. Tratar de fusionar las componentes User y Mashup reduciría el acoplamiento pero a costa de perjudicar seriamente la cohesión y haría menos intuitiva la compresión del código pensando en su mantenibilidad futura.

# Definición de los elementos del componente

En los Anexos se adjunta el diagrama de clases de este componente. A continuación se explican los componentes y las clases que tiene cada componente.

Además en los anexos se adjunta 2 diagramas de secuencias, esto para explicar mejor y más claramente el funcionamiento de este componente.

## 1. Componente Controlador

Este componente se encuentra construido con 2 archivos: el archivo routes y ApplicationController. Además dentro de este componente existen 2 componentes más: el componente User y el componente Mashup.

### 1.1.- Routes:

Este elemento es fundamental. Es el encargado de reenviar el request que recibe en la API[1] al controlador que es capaz de resolverlo. Es la entrada directa de este componente.

### 1.2.- ApplicationController

Este elemento define el comportamiento de los otros 3 controladores (UsersController, SessionsController y MashupsController). Tiene un método:

- Authenticate: valida el token para validar un usuario y para que este usuario pueda realizar acciones con sus mashups.

## 2. Componente User

Este componente tiene la responsabilidad de manejar todo el sistema de los usuarios. Esta compuesto por 1 modelo, 2 clases abstractas y 4 subclases.

### 2.1.- User

Este modelo hereda de ActiveRecord::Base, encargada de modelar los datos asociados a un usuario de la aplicación para su persistencia en la base de datos (capa Modelo). Sus atributos son:

- string mail
- string username

[1] En los anexos se adjunta la API de este componente

- string:digest password
- string token

Además incluye un método generate, el cual genera un Mashup temporal (leer componente mashup)

### 2.2.- UserController

Esta interfaz hereda de ApplicationController, se encarga de definir los distintos métodos de los controladores de usuario. Los métodos son:

- index: devuelve todos los usuarios.
- show: devuelve la información de un usuario
- create: crea un usuario (se sobrescribe para Facebook o Normal)
- update: actualiza un usuario (se sobrescribe para Facebook o Normal)
- destroy: elimina un usuario

### 2.3.- UserControllerNormal y UserControllerFacebook

Estos controladores redefinen los métodos create y update para manejar la creación y la actualización de los usuarios normales y los de Facebook, respectivamente.

### 2.4.- SessionController

Maneja la sesión de los usuarios, generando un token de autorización.

- create: genera el token de autorización
- destroy: destruye el token de autorización

### 2.5.- SessionControllerNormal y SessionControllerFacebook

Sobrescriben los métodos de SessionController.

## 3. Componente Mashup

Este componente tiene 4 modelos y un controlador.

### 3.1.- MashupController

Este controlador es el encargado de realizar todas las acciones correspondiente a los mashups. Todas las acciones sobre un Mashup requiere que se tenga identificado al usuario, por lo que se llama al método authenticate de ApplicationController para verificar la identidad del usuario.

Una decisión importante que se tomó es el funcionamiento de este controlador. Cada usuario tendrá asociado un Mashup, llamado Mashup temporal, este se autogenerará cuando el usuario cree una cuenta.

Cada vez que el usuario navegue y edite un mashup, se guardaran todos los parámetros (palabras, fuentes, valoración, etc) en el mashup temporal, utilizando el update definido en la API (ver anexos para más detalles).

Luego, cuando se quiera guardar un Mashup, se define el método create, el cual recibirá un nombre y clonará el Mashup temporal en un nuevo Mashup, utilizando el nombre recibido en los parámetros.

Cada vez que se guarde un Mashup se generará una nueva versión, dejando intactas las versiones anteriores. Si se desea se podrá borrar alguna versión más antigua.

En resumen las 5 acciones que tendrá el Mashup son:
- index: retorna un .json con todos los mashups de un usuario
- show: retorna un .json con un mashup (es el unico metodo que no necesita estar identificado el usuario, ya que cualquier persona puede ver un Mashup). Cabe notar que si no esta autentificado no podrá agregar términos en el Mashup. En caso contrario, se puede grabar el Mashup y pasa a ser parte de los Mashup del otro usuario.
- create: clona el Mashup temporal en un nuevo Mashup
- update: actualiza el Mashup temporal. **Nota:** esta acción se encargará de comunicarse con el componente de Parser e IA.
- destroy: elimina un Mashup de un usuario

### 3.2.- Mashup

Modelo que guarda la información básica de un Mashup, sus atributos son:
- string[ ] parameters

### 3.3.- Keyword

Este modelo guarda las palabras clave de un Mashup, sus atributos son:
- string keyword
- int value

### 3.4.- Link

Se guardan los vínculos que se adjuntan en un Mashup, sus atributos son:
- string title
- string link

- int value

### 3.5.- LinkSource

Este modelo guarda todas las fuentes, sus atributos son:

- string name
- picture logo

## Patrones de diseño

En este diseño reconocemos los siguientes patrones de diseño:

### 1. Bridge

Este patrón se ve claramente en 2 partes del diseño. En la herencia que existe entre UserController y sus clases hijas UserControllerNormal y UserControllerFacebook. Tambien con SessionController y sus clases hijas.

Este patrón se ve en este caso debido a que tenemos una clase abstracta que define los métodos y separamos 2 implementaciones distintas que se ejecutan para los 2 casos, usuario local y usuario de Facebook.

### 2. Chain of responsability

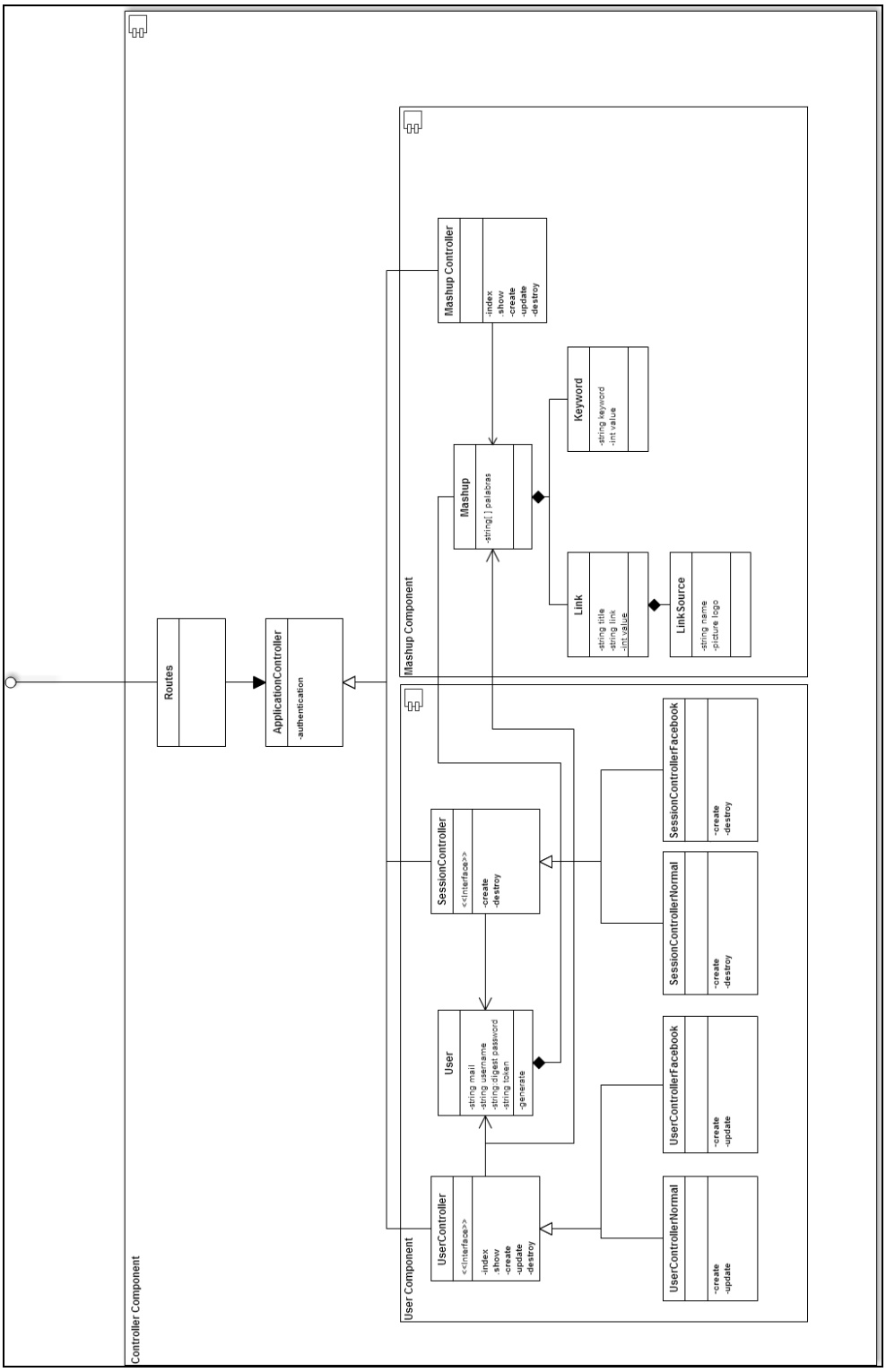
Este patrón se ve cuando el route.rb recibe un request, tiene que elegir cual es el controlador adecuado para responder el request. Luego de que el controlador recibe el request, este se comunica con alguna clase modelo que a su vez, al heredar de ActiveRecord::base, se comunica con la base de datos subyacente para recuperar los datos buscados y responder así la solicitud inicial.

### 3. Facade

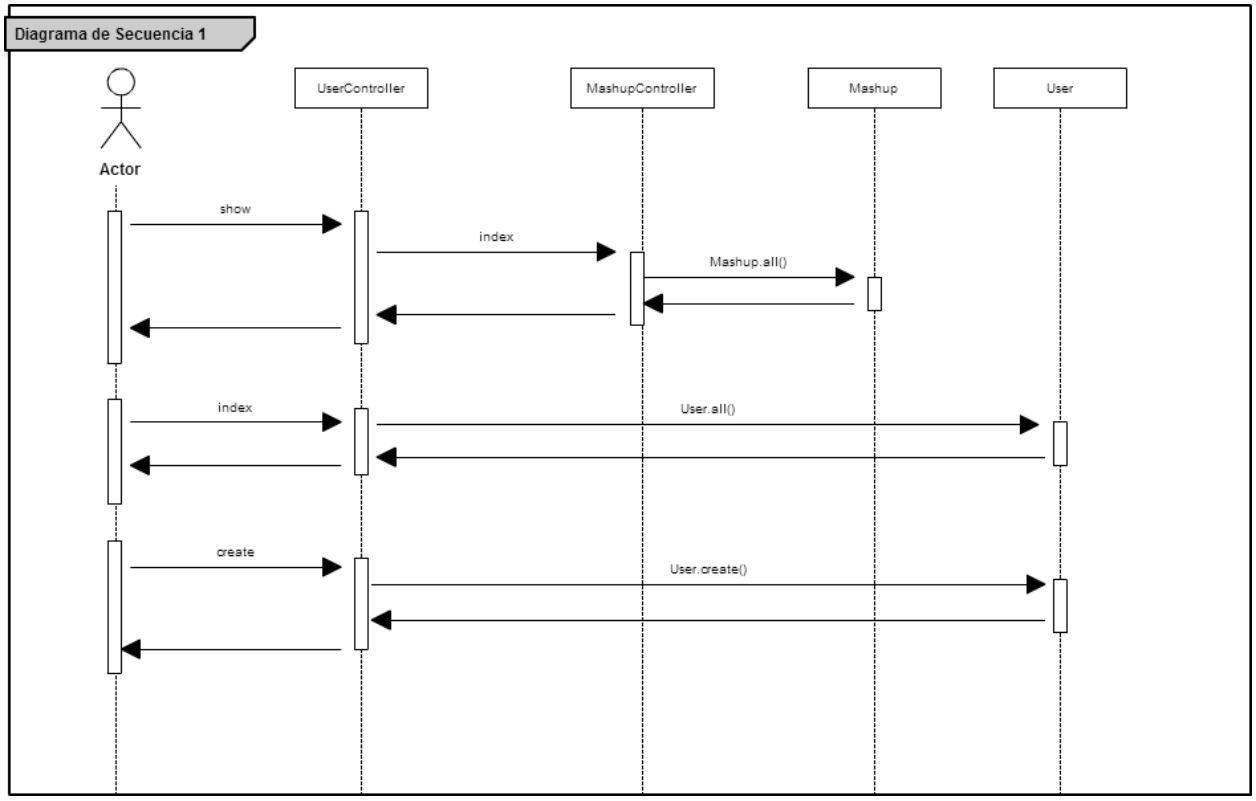
El patrón facade está presente de cierta manera en route.rb, dado que ésta parte de la aplicación hace las veces de main invocando los métodos de los controladores según el request recibido desde el frontend.

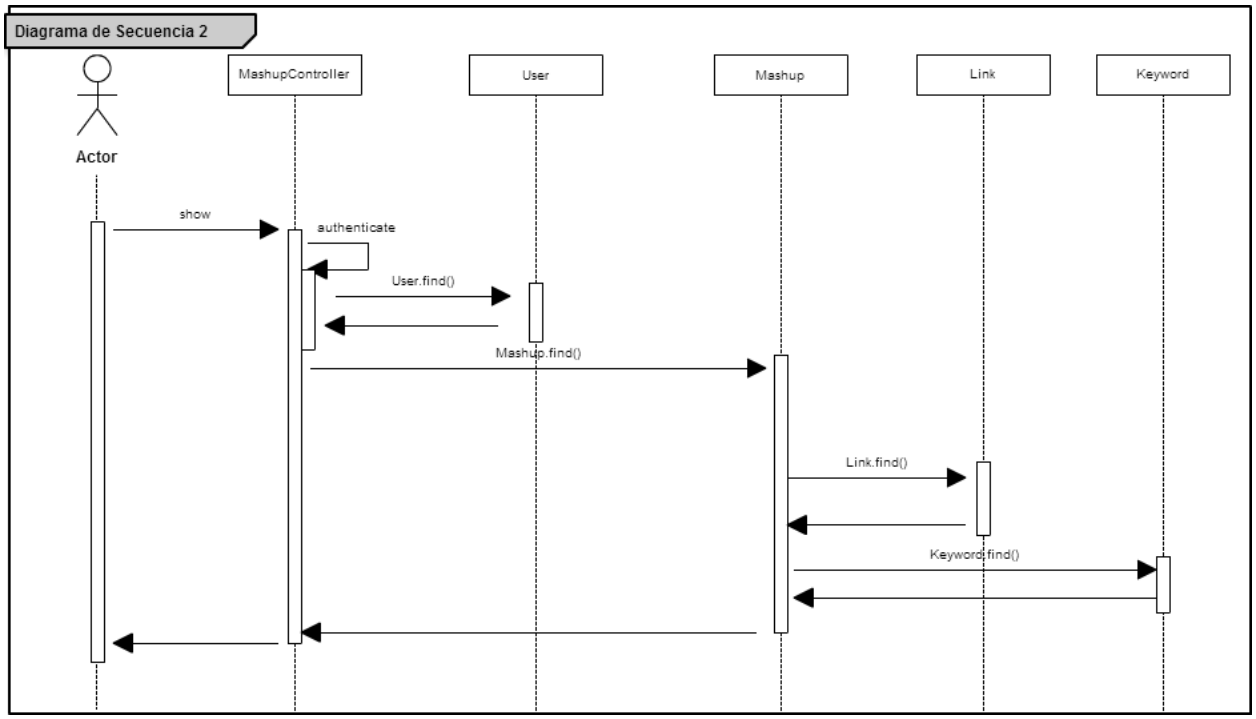
# Anexos

## Diagrama de Clases



# Diagramas de Secuencias






# API

## Usuario

**index**: GET http://ruta/users/
**show:** GET http://ruta/users/:id
**destroy:** DELETE http://ruta/users/

### Usuario Normal

**create:** POST http://ruta/users/
**update:** PUT http://ruta/users/

### Usuario Facebook

**create:** POST http://ruta/users/facebook
**update:** PUT http://ruta/users/facebook

## Sesión:

### Usuario Normal

**create:** POST http://ruta/session/
**delete:** DELETE http://ruta/session/

### Usuario Facebook

**create:** POST http://ruta/session/facebook
**delete:** DELETE http://ruta/session/facebook

## Mashup:

**update:** PUT http://ruta/mashup/
Se requiere en el header el token del usuario y los parámetros en el body. Este método actualiza el Mashup temporal del usuario.

**create:** POST http://ruta/mashup/
Se requiere en el header el token del usuario y un nombre en el body. Este método guarda el mashup temporal en el usuario asociado, con el nombre indicado.

**delete:** DELETE http://ruta/mashup/
Se requiere en el header el token del usuario y el id del mashup.

**index:** GET http://ruta/mashup/:id_user
Se requiere el id del usuario. Entrega todos los mashup del usuario en un .json.

**show:** GET http://ruta/mashup/:id
Se requiere el id del mashup. Entrega el .json del mashup