Ingeniería de Software II

Sistema "Twitteando para Ahorrar"

Grupo 6

1º Cuatrimestre 2013

LU	Nombre	email
667/06	Daniel Foguelman	dfoguelman@dc.uba.ar
767/03	Hernán Modrow	hmodrow@gmail.com
511/00	Leonardo Tilli	leotilli@gmail.com

1. Parte II

En este informe presentamos el análisis realizado al respecto de los distintos atributos de calidad identificados, exponiendo varios escenarios para cada uno de éstos. También mostramos la arquitectura definida para la aplicación, utilizando distintos diagramas (de conectores y componentes, y de alocación), y complementamos estos últimos mediante una descripción detallada de la interacción entre los distintos componentes y artefactos. También explicamos las decisiones que nos llevaron a definir la arquitectura, y cómo ésta cubre los distintos escenarios de calidad.

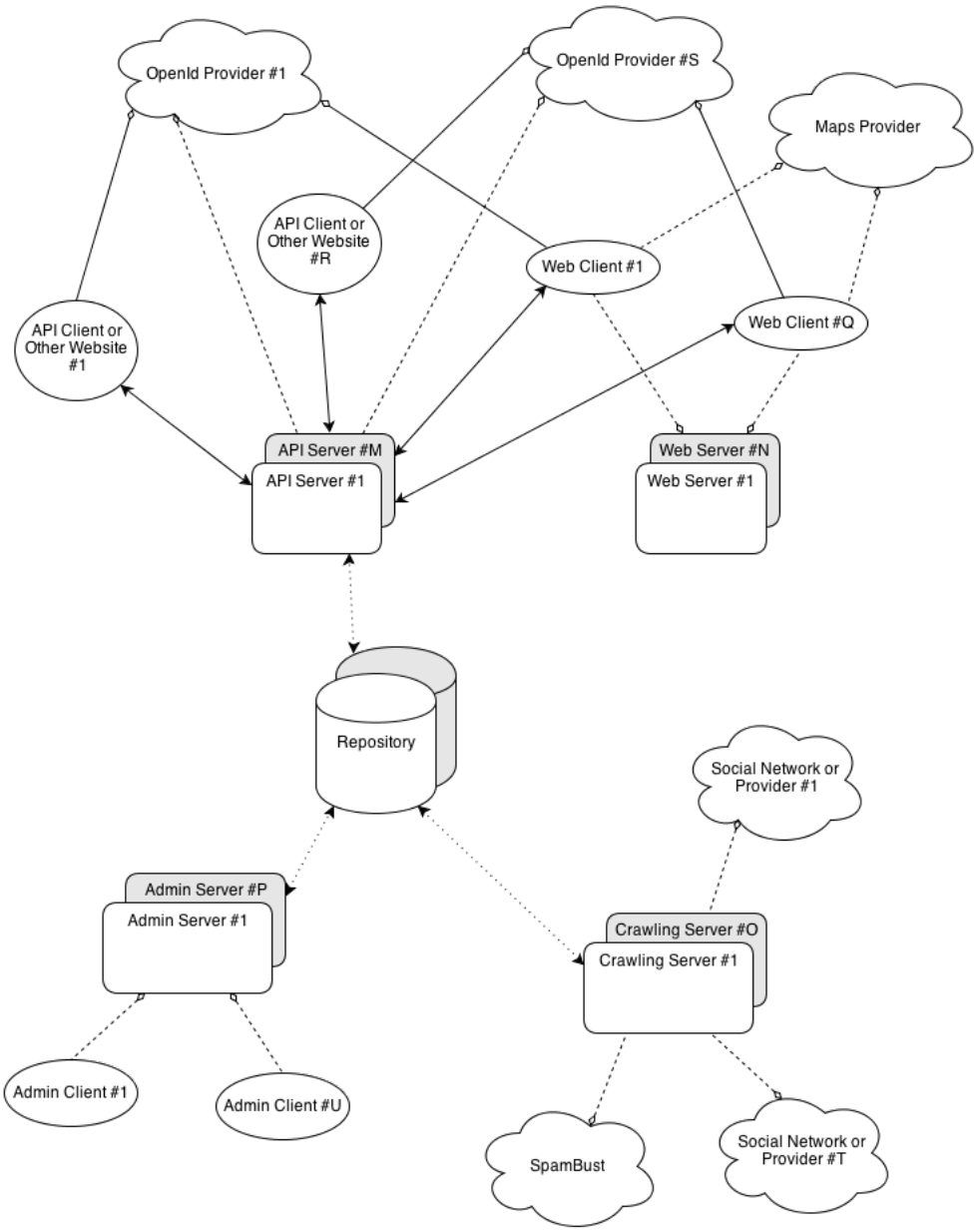
Por último, compartimos nuestra apreciación al respecto de las similitudes y diferencias entre la modalidad de trabajo para el primer trabajo práctico y éste, comparando metodologías y alcances.

Atributos de Calidad y Escenarios

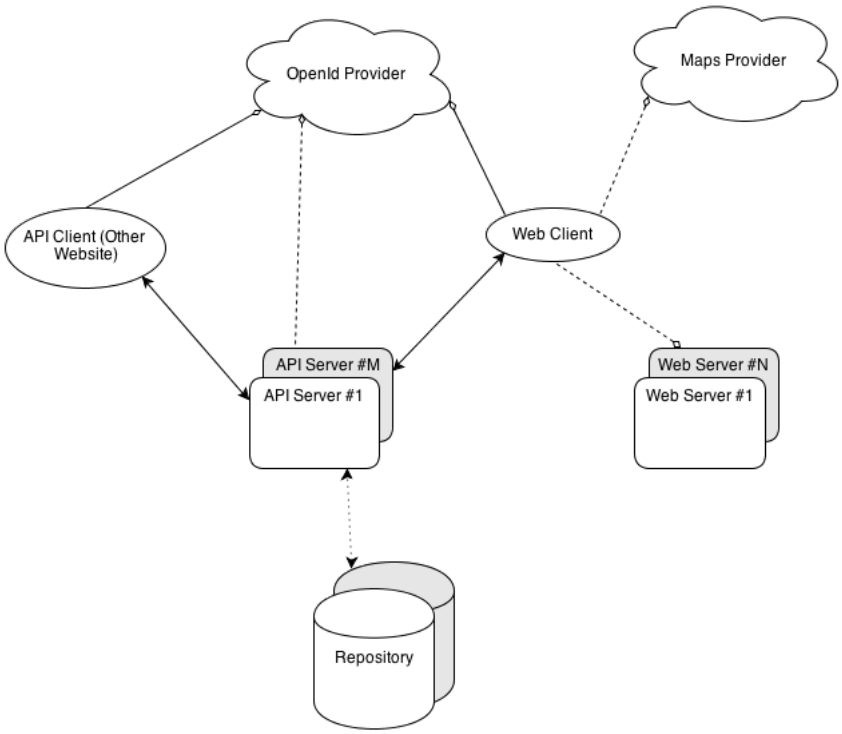
Explicación de la Arquitectura

Diagrama de Componentes y Conectores

Arquitectura Global



Consulta Web o API



En el diagrama de la consulta web se identifican dos tipos de clientes, que son los clientes del sitio Web y los que consumen únicamente la API web. Tenemos también dos tipos de componenetes servidor propios, que son los servidores del sitio Web y el servidores de la API web. Además, el diagrama nos muestra el repositorio de datos de configuración y consulta. Por último, tenemos distintos tipos de servicios externos que son consumidos por los componentes anteriores, como son los proveedores de OpenId y el proveedor de mapas.

Los clientes Web consumen el contenido estático directamente de los servidores del sitio Web, como puede ser el contenido HTML, JS, CSS e imágenes; se puede ver que el conector es de tipo cliente/servidor, lo que representa la sesión que arma el navegador con el servidor, y el bloqueo del thread de UI principal entre sucesivos requests (GET principalmente). En principio, este tipo de interacción es lo más simple posible, sin nigún tipo de autenticación; la mayor parte del contenido es cacheable y compone la UI de la aplicación (no tiene datos sensibles).

Tanto los clientes Web como los clientes de la API (como pueden ser otros sitios web) interactúan con el componente servidor API.

La autenticación es resuelta como parte de esta interacción, de acuerdo a la especificación de OpenId (`http://openid.net/specs/openid-authentication-2_0.html`):

- el cliente informa al componente servidor API el proveedor de OpenId que va a utilizar
- el componente API establece una sesión con el proveedor elgido (representamos esta sesión con un conector de tipo cliente/servidor) que podrá ser reutilizada para subsiguientes validaciones o para obtener datos extra del mismo cliente
- el cliente es redireccionado con el proveedor elegido para realizar la autenticación (representamos esta interacción como un envío de un mensaje a través de una conexión sincrónica)

- el componente API reutiliza la sesión establecida con el proveedor para verificar la información de autenticación y para obtener datos adicionales

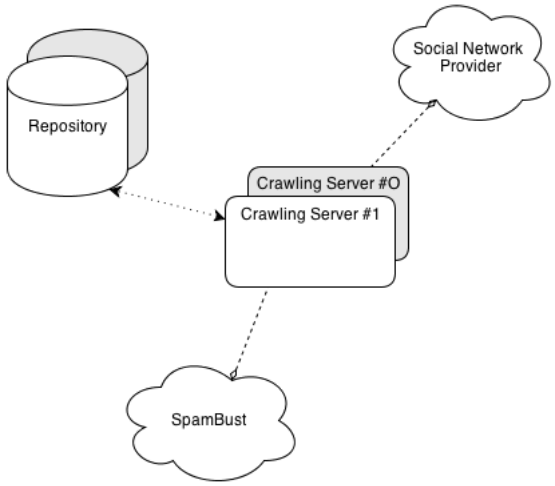
Más allá de manejar la autenticación, el componente servidor API responde a todos los pedidos de búsqueda o datos de configuración, de manera indistinta entre clientes Web o de API (usarían el mismo puerto); el tipo de conexión en este caso es asincrónica en ambos sentidos, para representar llamados tipo AJAX.

La interacción con el proveedor de mapas asumimos que se puede resolver directamente desde el cliente, mediante algún tipo de plugin descargado estáticamente desde el sitio Web.

El acceso al repositorio de datos y configuración lo hace únicamente el componente servidor API, para poder satisfacer los distintos requests, que agruparían todo el contenido dinámico de la aplicación.

Por último vale destacar que la cardinalidad del componente servidor API puede ser distinta a la del componente servidor Web, y dado que el primero atiende a consultas dinámicas, con workflows más complejos (como el de autenticación), y con mayor utilización de recursos, podría ser necesario escalarlo en mayor medida que al segundo, que sólo responde a requests de contenido estático. Además sólo es necesario implementar el proceso de autenticación en el componente servidor API, sustentado por el hecho de que sólo éste tiene acceso al repositorio, simplificando la arquitectura.

Crawling de Redes Sociales y Sitios Web



En el diagrama de crawling se identifica el componente de Crawling, el servicio de SpamBust, y el proveedor de red social (o sitio web de proveedor). Nuevamente, el diagrama nos muestra el repositorio de datos de configuración y consulta.

Los componentes de Crawling utilizan las distintas APIs provistas por las redes sociales para hacer las búsquedas de productos y ofertas; representamos esta interacción mediante un conector de tipo cliente/servidor, para generalizar el comportamiento de las distintas APIs. Podemos tener distintos componentes de Crawling, trabajando de manera independiente, cada uno realizando búsquedas distintas sobre la misma red social, o a distintas redes sociales.

El servicio de SpamBust es utilizado por los componentes de Crawling para filtrar ofertas de dudosa integridad, o que provienen de usuarios con baja reputación. Esta interacción también es representada mediante un conector tipo cliente/servidor, para generalizar las posibles implementaciones.

Por último, el repositorio es utilizado para consultar configuración de proveedores, usuarios, y redes sociales,

y también para persistir todos los datos de ofertas obtenidos en el proceso de crawling.

Diagrama de Alocación

Arquitectura y Calidad

Conclusión