Plantilla de parametrización de Consultas de Mashup

Graciela Vidal y Sandra Casas

GISP - Instituto de Tecnología Aplicada
Universidad Nacional de la Patagonia Austral
Campus universitario Piloto Lero Rivera s/nro. Río Gallegos Santa Cruz
vidalgracielaunpa@gmail.com, scasas@unpa.edu.ar

Resumen El acceso a la información en la Web así como el formato de los contenidos ha cambiado. Por esta razón el proceso de desarrollo también se ha modificado. Esto ha dado lugar al surgimiento de nuevas prácticas como Mashup, que utiliza la web como plataforma de desarrollo. Las APIs cumplen un rol fundamental en este tipo de desarrollo. A su vez el cúmulo de recursos disponibles dificulta la selección adecuada de los mismos. El trabajo presenta una plantilla preliminar para seleccionar y generar código para la ejecución de consultas simples y complejas de videos utilizando las APIs de Datos de Youtube y Vimeo.

Palabras clave: APIs, mashup, Youtube, Vídeo

1 Motivaciones

La forma de acceder a la información y el formato en el cual está disponible a través de la web ha cambiado. En consecuencia las aplicaciones web y su proceso de desarrollo también se han modificado. Las aplicaciones actuales son el resultado de la integración de múltiples recursos disponibles en la web, tales como APIs, contenidos y componentes, más precisamente consiste en extraer información de diferentes fuentes, combinarla y presentarla en otras formas [1]. Esta joven práctica de desarrollo se denomina Mashup, la misma utiliza como plataforma de desarrollo a la Web y además combina técnicas ya establecidas como incrustación, agregación e integración para generar nuevas aplicaciones [2]. Esta nueva forma de desarrollo permite implementar nuevas aplicaciones y servicios web completos [3] , además posibilita la incorporación de contenidos actualizados en sitios web ya desarrollados.

El video es un recurso audiovisual con amplia aceptación, si bien son populares desde hace tiempo en la industria de la música, resultan un medio muy utilizado para difundir información de todo tipo en la actualidad. Las plataformas web de vídeos tales como Youtube y Vimeo entre otras, representan un claro ejemplo de la evolución y crecimiento de este tipo de contenidos.

Ante estos cambios, los sitios basados en videos no sólo evolucionan en cuanto al volumen de información y la cantidad de usuarios, sino que también disponen de nuevas herramientas para desarrolladores. Esto promueve las tendencias actuales como mashup y facilita la distribución y el acceso a videos desde otras aplicaciones.

Las APIs (Applications Programming Interfaces) cumplen un rol fundamental en el proceso de desarrollo mashup. El surgimiento de estas prácticas también ha provocado el crecimiento de sitios web como ProgrammableWeb [4], Google developers [5] y Yahoo Developer Network [6], cuyo principal propósito es ofrecer colecciones de APIs pertenecientes a diferentes dominios como Fotos, Videos, Social, eCommerce, Música, Tools, entre otros. Las APIs presentan sus recursos, especificaciones, referencias y ejemplos en cada sitio, los mismos están destinados a desarrolladores experimentados. Sin embargo, el volumen de recursos disponibles en cada sitio es elevado por lo tanto resulta complejo seleccionar los adecuados.

El trabajo, presenta una plantilla preliminar que permite generar automáticamente el código para ejecutar consultas simples y complejas sobre videos, basadas en diferentes parámetros (categoría, términos y fecha), para ser integrado a diferentes aplicaciones. Esta herramienta contribuye a la selección de recursos mashup desde una API, más precisamente en la generación de consultas.

Este trabajo está organizado de la siguiente manera en la Sección 2 se exponen conceptos básicos sobre Mashup y se mencionan trabajos relacionados al desarrollo en esta práctica, en la Sección 3 el trabajo muestra una exploración de las APIs de Datos de los sitios Youtube y Vimeo, en la Sección 4 se presenta una Plantilla que permite parametrizar consultas y resultados para mashup, en la Sección 5 se exponen las conclusiones.

2 Mashup

Mashup se define como una práctica de desarrollo joven que utiliza la Web como plataforma y técnicas como incrustación, agregación e integración para la manipulación de recursos y datos con el fin de obtener aplicaciones complejas o incorporar funcionalidad en otras.

Un mashup (o Web mashup) es una aplicación que integra dos o más componentes mashup en cualquier capa de la aplicación, por ejemplo, en la capa de datos, lógica de aplicación y presentación, además pueden o no estar en comunicación cada entre ellos [7].

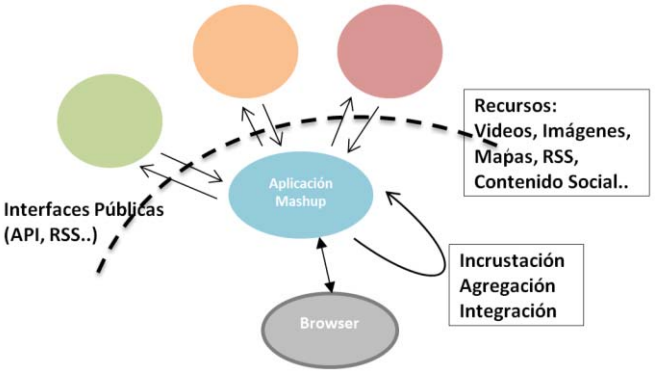

Figura 1. Arquitectura de Mashup

En el proceso de mashup se identifican los siguientes roles [8][9]:

Desarrollador de componente: es quien selecciona recursos (arquitecturas, bases de datos, estilos, formatos, etc.) luego desarrolla un componente mashup, describe su especificación y lo publica en la web.

Creador mashup (composer): es quien descubre, selecciona los componentes mashup y lleva a cabo la "mezcla", esto puede realizarse utilizando herramientas mashup o manualmente.

Usuario mashup: es quien configura y utiliza la aplicación final.

Mashup es considerado un paradigma impulsado por el usuario, los desarrolladores mashup desean crear soluciones que satisfagan sus necesidades situacionales**.**

El proceso de desarrollo mashup puede llevarse a cabo de dos formas: manual o asistido. En el primer escenario no existen APIs, el desarrollador debe analizar los componentes existentes, seleccionar el adecuado e implementar la aplicación. El problema que presenta esta forma es que las interfaces no son estables y que está orientado sólo a programadores expertos. El segundo escenario propone herramientas y plataformas, las cuales simplifican el proceso de desarrollo a usuarios menos experimentados.

Se han llevado a cabo estudios [10] para asistir en la selección de componentes, tal vez la tarea más compleja de este proceso, los cuales proponen un framework basado en la calidad de los componentes. El mismo consiste en la definición de un conjunto pasos para analizar y seleccionar componentes candidatos.

En [11] se aborda el desarrollo Mashup como una metodología simple basada en componentes con la cual usuarios finales pueden crear sus propias aplicaciones convirtiéndolos en desarrolladores e innovadores.

3 Exploración y análisis de APIs de Datos de sitios web de videos

Youtube posee una amplia sección destinada a los desarrolladores, la misma se organiza de la siguiente manera: Documentación, Widgets y herramientas, API de análisis, API de datos, Recursos para Android y Recursos para iOS. Estas herramientas permiten agregar funcionalidad de Youtube a todo tipo de aplicaciones, las operaciones son aplicables a diferentes recursos: Activities, Channels, Playlist, GuideCategories, Search, Videos, entre otros. La API se encuentra organizada por recursos y algunas de las funciones disponibles para cada recurso son list, insert, delete y update.

Los requisitos para utilizar los métodos disponibles en esta API son:

Obtener una clave API o un token OAuth 2.0, esto depende del tipo de solicitud que se envíe y del alcance de los datos a los cuales se pretende acceder (público o privado). Tanto para obtener una clave API como para un token es necesario acceder con una cuenta Gmail y registrar la aplicación a desarrollar, luego se deben seleccionar y habilitar las APIs adecuadas para ser invocadas en la aplicación.

La API de Datos de Youtube (v3) se encuentra organizada de la siguiente manera:

Inicio: breve descripción de los recursos y lenguajes de programación para los cuales está compatible la API de datos. Además de presentar Blogs de desarrolladores, repositorios y asistencia ante problemas en la implementación.

Guías: esta opción permite acceder a : 1) pasos requeridos para comenzar a utilizar la API, 2) los recursos y operaciones disponibles, 3) cuotas la API de datos utiliza una cuota para garantizar que los programadores utilicen el servicio según lo previsto, sin afectar la calidad del mismo o que limiten el acceso de las demás personas, 4) bibliotecas, 5) solicitudes autorizadas, 6) guías y tutoriales, 7) guía de implementación y migración

Referencias: en esta opción se describen en detalle cada uno de los recursos y los métodos disponibles para cada uno. Algunos de los recursos de la API de datos son: Activities, ChannelBanners, Channels, GuideCategories, PlaylistItems, Playlist, Search, Suscriptions, entre otros. Cada uno de los métodos se identifica por nombre, Solicitud HTTP y descripción.

Muestras: se presentan fragmentos de código implementados en diferentes lenguajes de programación, tales como Go, Java, Javascript, .NET, PHP, Python y Ruby.

Asistencia: permite realizar preguntas relacionadas al uso de la API de datos de Youtube en todas sus versiones.

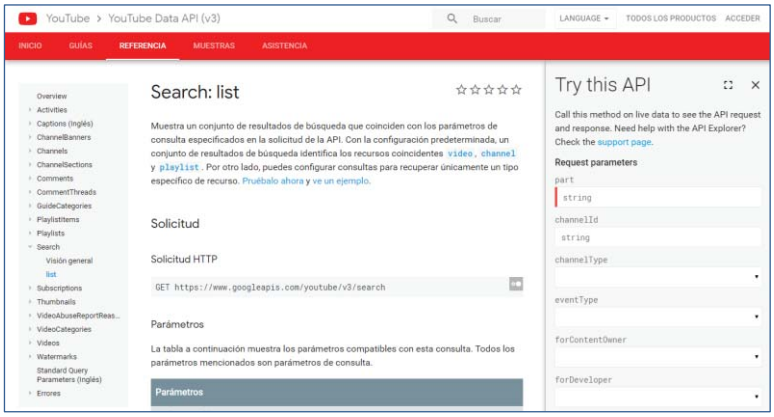

Figura 2. Distribución y especificación de recursos de la API de Datos de Youtube (v3)

La API de Vimeo fue construida para cumplir con los principios y prácticas REST (Representación de Estado de Transferencia), es un estilo arquitectónico de diseño para interactuar con recursos en línea, en este caso videos, utilizando métodos HTTP y estándar como GET, POST y PATCH.

El primer paso para utilizar la API de Vimeo consiste en registrar la aplicación, esta puede referirse a una aplicación móvil completa, una página web dinámica o un script de pocas líneas.

La API de Vimeo está disponible para ser integrada en diferentes lenguajes de programación como PHP, Python, Java y otros. Para realizar llamadas a la API se requiere un token de acceso, el cual consiste en una cadena de caracteres que cumple funciones de identificar a la aplicación y determinar un rango de acciones disponibles o ámbitos a los cuales puede acceder la aplicación.

El sitio web de la API de Vimeo, presenta la información de diferentes formas, tanto para desarrolladores nuevos o avanzados, incluye tutoriales, bibliotecas para

diferentes lenguajes de programación y referencias a la API, en la cual se describen todas las solicitudes y la estructura de los resultados.

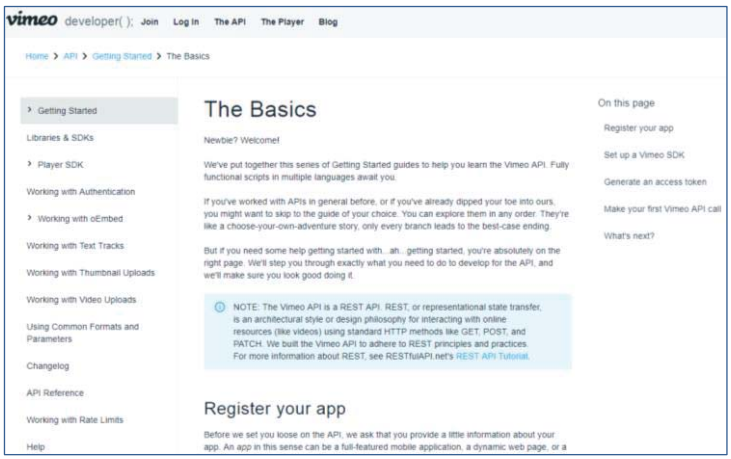

Figura 3. Recursos destinados a desarrolladores en el sitio Vimeo.

En la Tabla 1 se resumen diversos aspectos analizados sobre cada una de las API, los mismos están relacionados a las herramientas que proponen cada una para implementar consultas basadas en diferentes criterios.

Tabla 1. Análisis de API de Datos de YouTube y Vimeo.

	API de Datos de Youtube	API de Datos de Vimeo
Método de autenticación	`gapi.client.setApiKey(..)`	`Authorization:bearer {token}` ó `var vimeoToken = "xxxxxx";`
Método para cargar la API	`gapi.client.load(..)`	No requiere
Método o Solicitud para consultas	Métodos `gapi.client.youtube.search.list(..)` `gapi.client.youtube.videos.list(..)`	Solicitud `endpoint` `https://api.vimeo.com/{endpoint}?campo1=hola;…Access token=vimeoToken;`
Formato de los resultados	JSON Estructura `ítems`	JSON Estructura `data`
Permite parametrizar los resultados?	Si `part: snippet, player,..`	Si `fields=name, description, embed,..`

4 Plantilla de parametrización de consultas con APIs de Datos

El trabajo propone una plantilla preliminar en la cual es posible generar fragmentos de código parametrizado de consultas, utilizando las API de Youtube y Vimeo. El lenguaje utilizado para ambas APIs es Javascript, en el proceso de consulta se distinguen tres instancias:

Autenticación: el desarrollador debe registrarse y generar las claves de acceso para realizar las llamadas.
Consulta: se define una consulta parametrizada a partir de la API y del criterio seleccionado, además es posible parametrizar los resultados.
Resultados: los resultados son presentados un archivo JSON en el cual se describe los recursos que responden a la consulta.

La Figura 4 muestra la plantilla de configuración de consultas parametrizadas, el usuario debe seleccionar la API, el tipo de consulta y el criterio en el cual se basará la consulta.

Figura 4. Plantilla diseñada para generar código para consultas.

En la Tabla 2 se detallan los tipos de consulta que genera la plantilla definida, las mismas están basadas en tres parámetros: categoría, término y fecha.

Tabla 2. Tipos de consultas

		Categoría	Término	Fecha
Consulta	Simple	X		
			X	
				X
	Compleja	X	X	
		X	X	
			X	X

Consultas Simples

El desarrollador selecciona una API, luego el tipo de consulta que desea obtener, a partir del criterio seleccionado se habilita el campo correspondiente, en el caso de categoría, se despliega un listado con todas las categorías de cada sitio. Si la consulta estará basada en un término se habilita un campo para ingresar el mismo, en el caso de fecha es posible seleccionar la misma desde un date. En la Figura 5 se muestra el código generado utilizando la API de Datos de Youtube, basada en fecha.

Figura 5. Código de consulta de API YouTube.

El código generado podrá ser integrado a la aplicación que lo requiera, en este trabajo se utilizó el lenguaje Javascript, la ejecución del mismo por medio de un objeto request, obtiene como resultado un archivo JSON. El mismo consiste en una estructura array denominada ítems, cada ítems, a su vez puede contener otra estructura de ítems, lo que se define como propiedades anidadas. El desarrollador puede especificar en la consulta, los ítems y propiedades anidadas que incluirá la estructura resultante. Si no se especifican ítems en la consulta se incluirán todas las propiedades en el resultado. En la Figura 6 se presenta la estructura utilizada por la API de Datos de Youtube, en la misma se observan los siguientes ítems: kind y etag cuyos valores son únicos, por otro lado los ítems id y snippet contienen otros ítems.


```
"items": [
 {
  "kind": "youtube#searchResult",
  "etag": "\"Bdx4f4ps3xCOOo1WZ91nTLkRZ_c/-_HH2WqkK2AteMCoG7LGhwtqxzI\"",
  "id": {
   "kind": "youtube#video",
   "videoId": "7hPMmzKs62w"
  },
  "snippet": {
   "publishedAt": "2015-06-18T07:00:01.000Z",
   "channelId": "UCmQmc-Y4N9btVMUS1BfnyDw",
   "title": "Madonna - Bitch I&#39;m Madonna ft. Nicki Minaj",
   "description": "You're watching the official music video for “Bitch I'm Madonna”
   "thumbnails": {
    "default": {
     "url": "https://i.ytimg.com/vi/7hPMmzKs62w/default.jpg",
     "width": 120,
     "height": 90
    },
```


Figura 6. Estructura de resultados con API de Datos de Youtube

El desarrollador que utilice la plantilla propuesta deberá previamente, obtener una clave API en el sitio correspondiente. En el caso de Youtube deberá invocar el método `setApiKey(..)` para establecer su clave y el método `load(..)` para incorporar la API a la aplicación. El ejemplo 1 muestra el código generado para ejecutar una consulta de videos por categoría, invocando el método `gapi.client.youtube.videos.list(..)`, la lista de parámetros incluye: part, en el cual se indica qué subestructuras son requeridas en los resultados, en este caso snippet y player, maxResults permite definir la cantidad de recursos que incluirá cada página, chart: incluye los videos más populares y sus categorías, order: ofrece diferentes criterios para ordenar los resultados y por último el campo `videoCategoryId` en el cual se indica la categoría seleccionada para la consulta.

```
gapi.client.youtube.videos.list({part: "snippet,player", maxResults: 5,
chart:”mostPopular” order: "viewCount",videoCategoryId: mySearch });
```

Ejemplo 1. Consulta por categoría en Youtube.

En ejemplo 2, se presenta un fragmento de código para una consulta de videos por categoría en el sitio Vimeo, en este ejemplo la categoría es *animación* y se trata de un parámetro seleccionado en la plantilla, además se detalla la lista de campos que incluirá la estructura del resultado, en este caso se requiere: name, description, embed, created_time, modified_time. El desarrollador deberá proporcionar su token de acceso para ejecutar la consulta.

```
https://api.vimeo.com/categories/animation/videos?name,description,embed,
created_time,modified_time&access_token={API_KEY}
```

Ejemplo 2. Consulta por categoría en Vimeo.

Los resultados obtenidos utilizando la API de Vimeo son presentados en una estructura similar a la que propone la API de Youtube, excepto en la denominación de algunos campos. En Figura 7 se muestra la estructura se denomina data, de igual forma existen campos de la estructura que contienen otros campos, como por ejemplo embed, que a su vez contiene otras estructuras que incluyen más campos, como badges y live.


```
"data": [
  {
    "uri": "/videos/89797442",
    "name": "RUDENS X MADONA X ŠKODEK",
    "description": "I spent 1 week in my hometown called Madona together ..
    "type": "video",
    "link": "https://vimeo.com/user10987738/rudensxmadonaxskodek"
    "duration": 110,
    "width": 1280,
    "language": null,
    "height": 720,
    "embed": {
      "html": "<iframe src=\"https://player.vimeo.com/video/89797442?title=..."
      "badges": {
        "hdr": false,
        "live": {
          "streaming": false,
          "archived": false
        },
        "staff_pick": {
          "normal": false,
          "best_of_the_month": false,
          "best_of_the_year": false,
          "premiere": false
        },
        "vod": false,
        "weekend_challenge": false
      }
    },
    "created_time": "2014-03-22T18:47:33+00:00",
    "modified_time": "2018-02-06T00:17:21+00:00",
    "release_time": "2014-03-22T18:47:33+00:00",
```


Figura 7. Estructura de resultados con API de Datos de Vimeo

Consultas complejas

La flexibilidad de ambas APIs en cuanto a los parámetros admitidos en la consulta permitió establecer solicitudes basadas en varios criterios. En el ejemplo 3 se define una consulta utilizando la API de Youtube basada en término *q* y categoría (`videoCategoryId`), además se especifica el tipo de recurso *video*, se indica un criterio de ordenamiento y se restringe a los publicados desde el año 2015. Como parte del resultado se requiere el ítem snippet.

```
gapi.client.youtube.search.list({part: "snippet",type: "video",q:
encodeURIComponent($("#search").val()).replace(/%20/g, "+"),maxResults: 5,order:
"viewCount",publishedAfter: "2015-01-01T00:00:00Z", videoCategoryId:"1" });
```

Ejemplo 3. Consulta compleja basada en término y categoría

El análisis sobre la taxonomía de los resultados en ambas APIs, permitió identificar un conjunto de propiedades equivalentes, es decir que se denominan de diferente forma pero representan la misma información. Esto permitió parametrizar la información que será incluida en la respuesta.

5 Conclusiones

El trabajo propone una plantilla preliminar que permite generar código para integrar y ejecutar consultas simples y complejas utilizando las API de Datos de Youtube y Vimeo. Se realizó un análisis de cada API, enfocado en dos instancias, en primer lugar en el proceso que proponen para definir las consultas, en el cuál se identificaron tres pasos autenticación, consulta y resultados. La API Youtube utiliza métodos, mientras que Vimeo utiliza endpoint para realizar solicitudes de consulta.

Luego se realizó una exploración de la taxonomía definida por cada APIs para describir los recursos y presentar los resultados. Se observó una gran similitud en la estructura utilizada para describir los recursos, se identificaron propiedades equivalentes de manera que las respuestas a las consultas presenten la misma información, para cualquiera de las dos APIs.

La plantilla permite generar, a partir de una simple selección de API, criterios y tipo de consulta (simple o compleja), fragmentos de código parametrizados, para ejecutar consultas en lenguaje Javascript.

El enfoque de plantillas puede ser aplicado a consultas de otros dominios. El trabajo futuro está dirigido a desarrollar generadores de código enfocados en otras operaciones disponibles en las APIs mencionadas, posibilitar que el desarrollador seleccione desde la plantilla qué campos desea incluir en los resultados, e incluso incorporar nuevas APIs, para luego ser utilizado en proceso de desarrollo mashup.

Referencias

1. Makki S. K. & Sangtani J. Data Mashups & Their Applications in Enterprises. Internet and Web Applications and Services, ICIW '08. Third International Conference Athens: IEEE, pp 445-450. (2008)
2. Matera M., & Picozzi M. Next Generation Web Apps. Towards Transformative UX. Politecnico de Milano Dpto.Electronica e Informáatica (2015)
3. Yu J., Benatallah B., Casari F. & Daniel F. Understanding Mashup Development. Service Mashup. IEEE Computer Society. ISSN: 1089-7801.(2008)
4. ProgramableWeb https://www.programmableweb.com/ (2019)
5. Google Developer https://developers.google.com/ (2019)
6. Yahoo Developer Network https://developer.yahoo.com/api/ (2019)
7. Daniel, F., & Matera M.. Mashups Concepts, Models and Architectures. Milano, Italy: ISBN: 978-3-642-55049-2. Springer. (2014).
8. T. Janner et al., Patterns for Enterprise Mashups in B2B Collaborations to Foster Lightweight Composition and End User Development. Proc. IEEE Int'l Conf. Web Services (ICWS 09) pp. 976–983 (2009)
9. Gamble T.& Gamble R. Monoliths to Mashups: Increasing Opportunistic Assets. IEEE Software. pp 9-24 (2008)
10. Saeed A.A Quality-based Framework for Leveraging the Process of Mashup Component Selection. ISSN: 1651-4769. Department of Applied Information Technology. University of Gothenburg Sweden (2009)
11. Daniel F., Matera M.& Weiss M. Next in Mashup Development: User-Created Apps on the Web. IEEE Computer Society. ISSN: 1520-9202 (2011)