_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

AGREGADOR/GESTOR DE NOTICIAS Y CLIENTE PARA IPHONE

Francisco Javier Saorín

Tutor: José Enrique López Patiño

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2013-14

Valencia, 3 de Julio de 2014

Resumen

La presente memoria abarca la descripción y desarrollo del Trabajo Fin de Grado "Agregador/gestor de noticias y cliente para iPhone", consistente en un servicio web cuya función es la de recopilar periódicamente la información de noticias RSS de diferentes sitios web con ayuda de APIs de terceros como Google y la librería SimplePie para llevar a cabo el parseado de las fuentes RSS. Dichas fuentes son previamente añadidas al servicio a través de una aplicación móvil (app cliente) para iPhone en la que los usuarios consumen las noticias recogidas por el servicio y pueden guardar y compartir las noticias en las redes sociales u otras aplicaciones. Para la comunicación entre servicio y cliente se ha desarrollado una serie de scripts PHP que llevan a cabo la comunicación del servicio con una base de datos dónde se encuentran almacenados los usuarios, noticias y fuentes RSS utilizados en el servicio y aplicación.

Resum

La present memòria comprén la descripció i desenvolupament del Treball Fi de Grau "Agregador/gestor de notícies i client per a iPhone", consistent en un servici web, la funció del qual és la de recopilar periòdicament la informació de notícies RSS de diferents llocs web amb ajuda d'APIs de tercers com Google i la llibreria SimplePie per a dur a terme el parseig de les fonts RSS. Les dites fonts són prèviament afegides al servici a través d'una aplicació mòbil (app client) per a iPhone en la que els usuaris consumixen les notícies arreplegades pel servici i poden guardar i compartir les notícies en les xarxes socials o altres aplicacions. Per a la comunicació entre servici i client s'ha desenvolupat una sèrie de scripts PHP que duen a terme la comunicació del servici amb una base de dades on es troben emmagatzemats els usuaris, notícies i fonts RSS utilitzats en el servici i l'aplicació

Abstract

The present inform covers the description and development of the End of Degree Project "News feed/manager and iPhone's client", which consists in a web service with the function of collect periodically the information of RSS news feeds from different sites using third party APIs like Google and the library SimplePie to parse their data. This feeds are previously added to the web service via mobile application (client app) designed for iPhone. In this application, the users can read the news collected by the web service and additionally save and share them on the social network and other applications. A series of php scripts have been development for the purpose of the communication between the service and the client app. This scripts are in charge of quering the database with the feeds data and return it to the client.

Índice

Capítulo 1. Introducción 3

Capítulo 2. RSS (Really Simple Syndication) 5

2.1 Concepto 5

2.2 Historia 5

Capítulo 3. Objetivos del TFG 8

Capítulo 4. Metodología de trabajo 9

4.1 Gestión del proyecto 9

4.2 Distribución en tareas 9

4.2.1 Investigación 9

4.2.2 Desarrollo de backend (servidor) 9

4.2.3 Desarrollo de frontend (cliente) 9

4.2.4 Comprobación y testeo 10

4.3 Diagrama temporal 10

Capítulo 5. Desarrollo del trabajo 11

5.1 Servidor 11

5.1.1 Base de datos 11

5.1.2 Arquitectura 13

5.1.3 Usuarios 15

5.1.4 Suscripciones 16

5.1.5 Artículos 18

5.1.6 Servicio 19

5.1.7 Mantenimiento 20

5.2 Cliente 21

5.2.1 Interfaz de usuario 21

5.2.2 Desarrollo 23

Capítulo 6. Conclusiones y propuesta de trabajo futuro 27
6.1 Conclusiones 27
6.2 Espresso Reader 28
Capítulo 7. Anexos 29
7.1 SimplePie 29
7.1.1 Descripción 29
7.1.2 Compatibilidad 29
Capítulo 8. Bibliografía 30

Capítulo 1. Introducción

El crecimiento exponencial de las nuevas conexiones a Internet, tanto a través del equipo informático tradicional como de nuevos dispositivos móviles como smartphones y tablets, ha convertido a la sociedad tradicional en una sociedad continuamente conectada, dotada de innumerables posibilidades para acceder a todo tipo de información, en cualquier momento y en cualquier lugar.

Correo electrónico, redes sociales y noticias son algunos de los ejemplos de información a los que cada persona accede a lo largo de día, en numerosas ocasiones. Estas últimas, han conseguido aumentar su difusión a través de una gran cantidad de software que permite al usuario consumir artículos procedentes de sitios web especializados, canales de noticias o cualquier otro tipo de información que pueda ser consultada desde el primer momento de su publicación.

El formato RSS[1] (Really Simple Syndication), del que se puede obtener más información en el capítulo 2 de la presente memoria, ha permitido simplificar este proceso de difusión de contenido en la web, permitiendo difundir información actualizada a los usuarios que previamente se hayan suscrito a una fuente de contenidos (fuente RSS). Con ayuda de este formato, una gran variedad de aplicaciones web o nativas permiten al usuario recibir las noticias en cuánto son publicadas sin tener que ser él el que acuda a los sitios web para comprobar si existe nuevo contenido.

Un software, denominado "RSS reader" [2], "agregador" o "lector de feeds", puede presentar la información RSS al usuario mediante una interfaz web, de escritorio o móvil. Los usuarios pueden suscribirse a cualquier feed RSS introduciendo el URI del feed en el lector. Esta aplicación comprueba regularmente los feeds del usuario para descargar nueva información.

Uno de los lectores RSS más conocidos fue Google Reader, lanzado por Google el 7 de octubre de 2005. Su interfaz, basada en una bandeja de entrada similar a la de cualquier lector de correo electrónico, proporcionaba una manera rápida y cómoda de consumir las millones de noticias que manejaba día a día. Dichas noticias podían ser compartidas de diferentes formas y podían ser marcadas como destacadas, por lo que quedaban guardadas para futuras lecturas. El 13 de Marzo de 2013, Google anunció su desaparición[3], haciéndola efectiva el 1 de Julio de 2013, con el fin de aumentar la base de contenidos de su nueva red social Google+.

Sin embargo, tras su desaparición y consecuente abandono por parte de todas las aplicaciones que se alimentaban del servicio de Google, fueron muchas las plataformas que nacieron con el fin de proporcionar servicio a aquellas aplicaciones que habían quedado "huérfanas" y que los usuarios ansiaban utilizar de nuevo. Así nació Digg Reader, NetVibes, The Old Reader y la que

parece ser que se ha convertido en la principal alternativa para el público en general, Feedly, que recientemente ha proporcionado una API abierta a desarrolladores para poder dar servicio a una gran variedad de aplicaciones web y móviles existentes en el mercado.

Capítulo 2. RSS (Really Simple Syndication)

2.1 Concepto

RSS (Really Simple Syndication) también conocido como "Rich Site Summary", utiliza un conjunto de formatos web estándar para publicar, de manera frecuente, información actualizada: entradas de blog, noticias, audio, video… Un documento RSS (denominado "feed", "web feed" o "canal") incluye una serie de texto y metadatos como la fecha de publicación y el nombre del autor.

Los feeds RSS permite a las publicaciones gestionar y difundir datos automáticamente. Un formato de fichero XML estándar garantiza la compatibilidad con todo tipo de máquinas y aplicaciones. Estos feeds también benefician a los usuarios, quienes quieren recibir, de manera frecuente, actualizaciones de sus sitios favoritos o agregar datos de diferentes sitios.

La suscripción a un sitio RSS elimina la necesidad de que el usuario tenga que comprobar manualmente el sitio web para recibir nuevo contenido, su aplicación constantemente monitoriza el sitio e informa al usuario de que hay nueva información. La aplicación también puede descargar la información por sí sola para que el usuario ya la tenga disponible.

2.2 Historia

Los formatos RSS fueron precedidos de muchos intentos a través de la sindicación web, que no consiguieron mucha popularidad entonces. La idea básica de reestructurar la información sobre diferentes sitios web comienza en 1995, cuando Ramanathan V. Guha e integrantes del grupo de tecnología avanzada de Apple desarrollaron el "Meta Content Framework".

La primera versión de RSS (RDF Site Summary) fuera creada por Dan Libby y Ramanathan V. Guha en Netscape. Fue lanzada en Marzo de 1999 para su uso en el portal "My.Netscape.Com". Esta versión comenzó a conocerse como RSS 0.9. En Julio de 1999, Dan Libby desarrollo una nueva versión, RSS 0.91, que simplificaba el formato eliminando los elementos RDF e incorporaba nuevos elementos procedentes del formato de noticias de Dave Winer. Libby también renombró el formato de RDF a "RSS Rich Site Summary" y esbozó un desarrollo más completo del formato en una especificación.

Esta sería la última participación de Netscape en el desarrollo de RSS durante 8 años. Ya que RSS fue bien recibido por las publicaciones web, que querían que sus feeds fueran usados en el portal de Netscape y otros portales RSS, Netscape dejó de dar soporte a su portal en Abril de 2001 durante la reestructuración del nuevo propietario de la compañía, AOL, eliminando documentación y herramientas usadas por este formato.

Dos entidades surgieron para llenar el vació, sin la ayuda ni aprobación de Netscape: El "RSS-DEV Working Group" y Dave Winer, cuya compañía (UserLand Software) había publicado algunas de las primeras herramientas de publicación fuera de Netscape que podían leer y escribir RSS.

Winer publicó una versión modificada de RSS 0.91 en el sitio web de la compañía, cubriendo cómo había sido utilizado en los productos de la misma y reclamó el copyright del documento. Unos meses más tarde, UserLand intentó registrar la marca RSS en Estados Unidos pero fue rechazada en Diciembre de 2001.

El "RSS-DEV Working Group" [4], un proyecto cuyos miembros incluían a Guha y representates de O'Reilly Media y otros, desarrollaron RSS 1.0 en Diciembre de 2000. Esta nueva versión, introdujo soporte para RDF y añadió soporte para espacios de nombre XML, adoptando nuevo elementos de metadatos como los procedentes de Dublin Core [5] .

En Diciembre de 2000, Winer publicó RSS 0.92, un conjunto de pequeños cambios relacionados los elementos de cierre, que permitió transportar sonidos de audio sobre feeds RSS y ayudó a asentar las bases del podcasting. También publicó un borrador de RSS 0.93 y RSS 0.94 que fueron finalmente retirados.

En Septiembre de 2002, Winer publicó una importante nueva versión del formato, RSS 2.0, que volvió a renombrarse a "Really Simple Syndication" e eliminó atributos de tipo introducidos en RSS 0.94 y añadió soporte para espacios de nombre. Para preservar la retrocompatibilidad con RSS 0.92, el soporte para espacios de nombres se aplicó únicamente para los contenidos incluidos en RSS 2.0 y no para los elementos en sí. A pesar de que otros estándares como Atom intentaron corregir sus limitaciones, los feeds RSS no han sido desplazados por otros formatos debido al soporte completo para espacios de nombre.

Debido a que ni Winer ni el RSS-DEV Working Group estaban implicados en Netscape, ellos no reivindicaron el nombre y formato RSS. Esto generó alguna que otra controversia sobre la cuestión de que entidad era realmente el padre de RSS.

Un producto en continuo debate fue la creación de un formato alternativo, Atom, que comenzó a desarrollarse en Junio de 2003. Este formato, cuya creación respondía a los deseos de disponer de un formato libre de las limitaciones de RSS, ha sido adoptado por el IETF en el RFC 4287.

En Julio de 2003, Winer y Userland Software atribuyeron el copyright de la especificación RSS 2.0 al "Harvard's Berkman Center for Internet & Society" al mismo tiempo que Winer lanzaba el "RSS Advisory Board" con Brent Simmons y Jon Udell, un grupo cuyo propósito era el de mantener y publicar la especificación y responder a las dudas sobre el formato.

En Septiembre de 2004, Stephen Horlander creó el conocido icono RSS para su uso en el navegador Mozilla Firefox.

Figura 1. Icono representativo del formato RSS

En Diciembre de 2005, el equipo del navegador Microsoft Internet Explorer y el de Microsoft Outlook, anunciaron en sus respectivos blogs que estaban adoptado este icono. En Febrero de 2006, Opera Software siguió los mismo pasos. Esto hizo, del rectángulo naranja con ondas de radio blancas, el estándar de los feeds RSS y Atom, reemplazando a una gran variedad de iconos y textos que habían sido previamente utilizados para identificar al formato.

En Enero de 2006, Rogers Cadenhead relanzó el RSS Advisory Board sin la participación de Dave Winer, con un enfundado deseo de continuar el desarrollo del formato y resolver sus ambigüedades. En Junio de 2007, el equipo revisó su versión de las especificaciones para confirmar que los espacios de nombre podían extenderse a los elementos del núcleo y los atributos, como Microsoft había hecho en Internet Explorer 7. De acuerdo con su punto de vista, una diferencia de la interpretación permitía a las publicaciones no tener claro que estaba permitido o prohibido.

Capítulo 3. Objetivos del TFG

El objetivo de este TFG consiste en el desarrollo de una alternativa real al ya descontinuado agregador de contenidos Google Reader. Una alternativa que, teniendo en cuenta las limitaciones propias del servidor utilizado (lejos de la grandísima cantidad de recursos utilizados por Google en su correspondiente servicio) intentará ofrecer una experiencia al usuario muy similar a la ofrecida por el ya desaparecido servicio.

Dicho objetivo se consigue a través de una tarea programada en el servidor, encargada de acceder periódicamente a todas las fuentes de noticias guardadas en la base de datos, realizar una lectura de las mismas y guardar en el servidor todas las noticias publicadas hasta ese momento, sin diferenciar por usuario, sólo por fuente de contenidos.

Por otro lado, el servidor, además de llevar a cabo esta tarea y almacenar los contenidos en la base de datos, proporciona una API que la aplicación cliente (disponible para iPhone) utiliza para recopilar todas las noticias procedentes de las fuentes a las que el usuario está suscrito. Este usuario, que previamente se habrá registrado en el servicio con una dirección de correo y una contraseña, podrá hacer uso del servicio descargando, consumiendo y marcando como “leída” o “guardada” una o varias noticias a través de la aplicación.

La aplicación móvil, clasifica las noticias por fuente de contenidos y elimina cada una de ellas que es marcada como “leída”. Si el usuario desea consultar cualquiera de estas noticias en el futuro, tiene la opción de guardar la noticia marcándola como “guardada”. Mediante esta opción, la siguiente vez que acceda al servicio desde otro dispositivo, podrá acceder a las que guardó previamente además de las nuevas noticias disponibles.

Capítulo 4. Metodología de trabajo

4.1 Gestión del proyecto

El proyecto es desarrollado teniendo en cuenta las limitaciones existentes del servidor, perteneciente a un servicio de hosting. Es de esperar que, a pesar de obtener la misma experiencia de usuario que Google Reader, es imposible disponer de un producto totalmente comercial ya que la lectura de las diferentes fuentes RSS se realiza de manera periódica, con un intervalo de 10 minutos y no puede ser realizada en tiempo real, menos aún si se dispone de una gran cantidad de fuentes RSS (+1000) que deben ser leídas por una sola máquina, a pesar de que se hace de manera concurrente.

4.2 Distribución en tareas

El desarrollo del trabajo se ha llevado a cabo mediante la realización de 4 grandes tareas o etapas: Investigación, desarrollo de backend o API, desarrollo de frontend o cliente y la fase de comprobación y testeo.

4.2.1 Investigación

En esta primera etapa se realizó un estudio sobre las características ofrecidas por Google Reader y con qué tecnología podría llevarse a cabo su implementación. También se estudió de que manera podría realizarse la lectura o parseo de las fuentes RSS, decidiéndose por utilizar finalmente la librería de código abierto SimplePie[6] sobre PHP.

Adicionalmente, se procede a la contratación de un servicio de hosting profesional proporcionado por la empresa española Dinahosting S.L que sirve como infraestructura para albergar toda los ficheros y recursos utilizados por la parte de backend del proyecto.

4.2.2 Desarrollo de backend (servidor)

Con ayuda de la librería SimplePie, que es tratada en uno de los anexos de esta memoria, y la API de Google Feeds se implementó una serie de scripts PHP que dotan al proyecto de una interfaz de comunicación con la que la aplicación puede hacer login y registro de usuarios, añadir y eliminar suscripciones, descargar noticias y marcarlas con diferentes etiquetas. Adicionalmente, también se desarrolló el servicio encargado de recorrer las fuentes RSS periódicamente para actualizar la base de datos de noticias.

4.2.3 Desarrollo de frontend (cliente)

Tras desarrollar la parte correspondiente del servidor, se comenzó a desarrollar la aplicación para iOS que permite a los usuarios consumir los contenidos de las fuentes a las que está

suscrito. La comunicación queda implementada mediante la llamada a diferentes funciones de la API mediante parámetros POST y la correspondiente respuesta JSON por parte del servidor.

4.2.4 Comprobación y testeo

Finalmente, se realiza una serie de comprobaciones y test sobre el servicio y aplicación mediante la creación de varios usuarios, suscripción a fuentes y desarrollo de diferentes casos de uso descargando noticias, marcando y modificando suscripciones.

4.3 Diagrama temporal

Capítulo 5. Desarrollo del trabajo

5.1 Servidor

5.1.1 Base de datos

El núcleo de almacenamiento del servicio es ofrecido por una base datos relacional MySQL, constituida por una serie de tablas que se describen a continuación. Cada una de estas tablas proporciona toda la información necesaria para el correcto funcionamiento del servicio: usuarios, contenido, fuentes RSS, estado de lectura y guardado, acceso a servicio…

Toda su elaboración se ha llevado a cabo mediante la conocida interfaz web de gestión de bases de datos PHPMyAdmin[7], proporcionada por el proveedor de hosting contratado para este proyecto. Además se han configurado una serie de parámetros de acceso a la base de datos que son utilizados por cada uno de los scripts que conforman el servicio web.

Tabla	Descripción
Items	Almacena los datos de cada una de las noticias que son añadidas desde sus correspondientes fuentes RSS, como pueden ser el título de la noticia, su autor, contenido, URL…
ReadItems	Relaciona cada elemento con un estado de lectura y con cada usuario. Es decir, su información indica qué noticias han sido ya leídas y por qué usuario ha sido leída.
SavedItems	Misma finalidad que *ReadItems* pero para el estado de guardado para futura consulta.
Subscriptions	Almacena la información necesaria para poder acceder a cada una de las fuentes RSS, es decir, el título y la dirección del fichero XML/RSS de la que se pueden extraer las noticias.
Users	Almacena la información correspondiente a cada uno de los usuarios registrados, es decir, su email de acceso, hash de contraseña y token de uso para poder realizar las llamadas al servicio web.
UsersSubscriptions	Relaciona cada usuario con una o varias fuentes RSS, proporcionado información sobre las suscripciones activas que tiene cada uno de los usuarios registrados.

Tabla Items	Descripción de campo	Tipo de dato
ID	ID único de noticia	varchar(255)
title	Título de la noticia	varchar(255)
author	Autor de la noticia	varchar(255)
content	Contenido de la noticia en formato HTML	text
created	Timestamp de la fecha en la que fue publicada	int
subscription	ID de la fuente RSS a la que pertenece	int
url	URL de la noticia para acudir al sitio web original	varchar(255)

Tabla ReadItems	Descripción de campo	Tipo de dato
user	ID único de usuario	int
item	ID único de noticia	varchar(255)

Tabla SavedItems	Descripción de campo	Tipo de dato
user	ID único de usuario	int
item	ID único de noticia	varchar(255)

Tabla Subscriptions	Descripción de campo	Tipo de dato
ID	ID único de fuente RSS	int
title	Título de la fuente	varchar(255)
url	URL del fichero XML/RSS de la fuente	varchar(255)

Tabla Users	Descripción de campo	Tipo de dato
ID	ID único de usuario	int
username	Dirección de correo	varchar(64)
password	Resumen de la contraseña del usuario	varchar(64)
token	Token de uso	varchar(64)

UsersSubscriptions	Descripción de campo	Tipo de dato
user	ID único de usuario	int
subscription	ID único de fuente RSS	int

5.1.2 Arquitectura

La arquitectura del servidor está formado por dos grandes plataformas: la API pública, usada por la aplicación para el intercambio de información, y los scripts privados que conforman el servicio permanente, que recopila las noticias periódicamente, así como otros scripts auxiliares utilizados por la parte pública y el propio servicio.

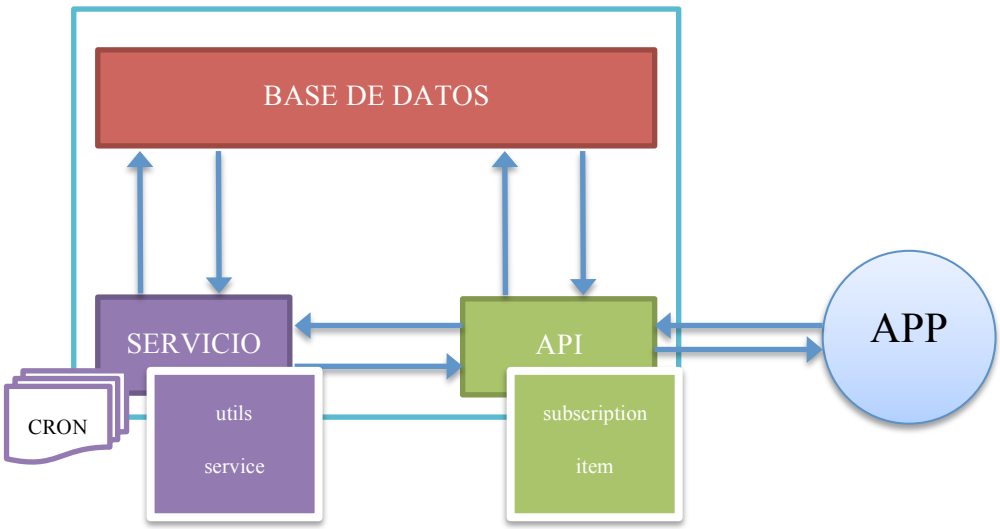


Figura 2. Arquitectura del servidor.

Todo el servicio web, correspondiente a la parte del servidor del proyecto es accesible mediante la siguiente dirección web:

http://reader.fjsaorin.com/api/v1/X/Y

Dónde X e Y definen la categoría de métodos de la API y el método respectivamente. De esta manera se puede encontrar la siguiente estructura de métodos:

X	Y	Descripción
subscription	*get*	Obtención de las suscripciones activas de un usuario.
	update	Permite añadir y eliminar una suscripción de un usuario.
item	*get*	Obtención de todas las nuevas noticias no leídas y guardadas para un usuario, a partir de una fecha determinada.
	update	Permite marcar/desmarcar una noticia como leída/guardada.
user	*new*	Registra un nuevo usuario en el servicio
	login	Permite iniciar sesión en el servicio con unos determinados parámetros de acceso.

A través de esta API se pueden acceder a todos los scripts PHP que conforman todas las tareas y funciones que puede ofrecer el servicio. Estas terminaciones pueden ser llamadas por cualquier aplicación cliente si se dispone de un token de usuario que se asigna una vez la sesión es iniciada en el dispositivo.

Adicionalmente, existen otras terminaciones que conforman métodos internos del servicio, métodos auxiliares, mantenimiento y parámetros de acceso de la base de datos que son utilizados por el resto de scripts del servicio.

Estas funciones privadas son las siguientes:

X	Y	Descripción
utils	*info*	Contiene variables globales para todo el servicio correspondiente al acceso a la base de datos y tiempos de caché.
	util	Incluye funciones auxiliares como la generación de cadenas alfanuméricas aleatorias, comprobación de validez de token…
service	*core*	Realiza el recorrido por todas las suscripciones de la base de datos y llama a "fetch" de manera concurrente para añadir las nuevas noticias a la base de datos.
	fetch	Obtiene el fichero XML/RSS de una determinada fuente para realizar su lectura.
	maintenance	Permite eliminar aquellas noticias con una antigüedad mayor a 1 mes y que no han sido guardadas por los usuarios.
lib	*SimplePie*	Librería encargada de leer los ficheros XML/RSS para obtener toda la información correspondiente a las noticias de una fuente RSS.

5.1.3 Usuarios

Para poder utilizar la API del servicio es necesario haberse registrado previamente en el sistema. Este registro se realiza mediante una dirección de correo electrónico y una contraseña que servirán como datos de acceso.

Dicho registro se realiza mediante la terminación *user / new*. Mediante este método se inserta en la tabla *Users* el nombre de usuario proporcionado y el resumen o hash MD5 de la contraseña para salvaguardar la misma de posibles vulnerabilidades y mantener la privacidad de la información de usuario.

user / new		
Parámetros de entrada	*u*	Dirección de correo
	p	Contraseña

Respuesta JSON con código de error	*0*	Registro realizado correctamente
	11	Falta algún parámetro para completar la operación
	12	No es una dirección de correo válida
	13	Error al operar con la base de datos

Si el registro se realiza correctamente, el usuario ya puede iniciar sesión en el sistema. Esta acción se realiza sobre *user / login*.

Para poder utilizar los servicios de la API es necesario disponer de un token de sesión (cadena alfanúmerica aleatoria) que el servidor genera cada vez que el usuario se registra o inicia sesión. De esta manera se garantiza que las llamadas a la API se realizan de manera controlada y no se permite la ejecución de los scripts fuera de la aplicación cliente.

Internamente, este método, comprueba si el usuario existe comprobando el nombre de usuario y el resumen MD5 de la contraseña y asigna una nuevo token al usuario.

user / login		
Parámetros de entrada	*u*	Dirección de correo
	p	Contraseña
Respuesta JSON con código de error en caso de que no se haya realizado la operación con éxito	*11*	Falta algún parámetro para completar la operación
	14	Login incorrecto
Respuesta JSON con información de usuario si no ha hay problemas	*id*	ID único de usuario
	token	Token de sesión

5.1.4 Suscripciones

Para comenzar a recibir noticias, los usuarios deben suscribirse a sus fuentes RSS favoritas. Estas fuentes lo forman un archivo RSS en formato XML incluido en la mayoría de publicaciones online como blogs, periódicos, revistas…

Mediante la terminación *subscription / update* los usuarios pueden realizar sus modificaciones sobre las suscripciones activas que tiene en ese momento o nuevas suscripciones que desee añadir.

En el caso de añadir una nueva suscripción, este método utiliza la API de Google Feed, que a partir de una dirección web, palabras clave o título es capaz de obtener su fuente RSS correspondiente. Cabe destacar que la librería SimplePie ya incluye la posibilidad de "autodescubrir" las fuentes dada una URL pero finalmente se ha optado por utilizar este desarrollo de Google por cuestión de fiabilidad.

Si la fuente RSS correspondiente es encontrada se procede a la inclusión de la misma en la tabla *Subscriptions* de la base de datos, guardando su título y dirección de fuente. Por otro lado, también se suscribe al usuario a dicha fuente incluyendo los IDs únicos correspondientes en la tabla *UsersSubscriptions*.

En el caso de que se desee elimina una suscripción, simplemente se eliminan estos IDs únicos de la tabla *UsersSubscriptions*. Cabe destacar que, en este caso, la fuente no es eliminada de la base de datos ya que otros usuarios pueden estar utilizándola.

subscription / update		
Parámetros de entrada	*u*	Dirección de correo
	t	Token de sesión
	s	En caso de nueva suscripción: palabras clave, url del sitio web, título de la publicación… En caso de eliminación de suscripción: ID de fuente
	d	0 = Nueva suscripción 1 = Eliminación de suscripción
Respuesta JSON en caso de nueva suscripción	*id*	ID de fuente
	title	Título de fuente
	url	URL de fuente
Respuesta JSON en caso de eliminación o no encontrada	*0*	Fuente no encontrada
	1	Suscripción eliminada

Adicionalmente, también es posible obtener todas las suscripciones activas del usuario mediante la terminación *subscription / get* . Esto se consigue accediendo a la información de todas las fuentes asignadas al usuario en la tabla *UsersSubscriptions* y *Subscriptions*. La principal finalidad de este método es poder presentar en la aplicación una lista de suscripciones activas para poder seleccionar cuál se desea eliminar.

subscription / get		
Parámetros de entrada	*u*	Dirección de correo
	t	Token de sesión
Respuesta JSON con array de todas las suscripciones activas del usuario	*id*	ID de fuente
	title	Título de fuente
	url	URL de fuente

5.1.5 Artículos

Una vez los usuarios se encuentran suscritos a diversas fuentes de noticias de su interés ya es posible comenzar a recibir las noticias de dichas fuentes.

El método encargado de proporcionar las últimas noticias y el que se ha convertido en el método más complejo de desarrollar se denomina *item / get*.

Este método realiza en 3 principales etapas con el fin de poder clasificar las noticias con una o varias etiquetas o tags.

En primer lugar obtiene las últimas noticias registradas en la base de datos que pertenecen a las suscripciones activas del usuario y no han sido leídas aún.

En segundo lugar obtiene, con independencia del tiempo en que fueron publicadas, aquellas que el usuario ha marcado como guardadas.

Finalmente, se obtienen todos las noticias que han sido guardadas y ya han sido leídas por el usuario.

El resultado es una lista ordenada por fecha de publicación, de más reciente a más antiguo de todos los artículos que existen en las fuentes de noticias a las que está suscrito el usuario. Proporcionado toda la información posible sobre cada una de las noticias para que puedan ser mostradas en la aplicación sin problema.

Cabe destacar que uno de los parámetros de entrada es el timestamp de la última petición, de esta manera se ahorra gran cantidad de tiempo en peticiones consecuentes, no teniendo que devolver los artículos de nuevo si ya lo ha recibido anteriormente.

subscription / get		
Parámetros de entrada	*u*	Dirección de correo
	t	Token de sesión
	d	Timestamp de última petición. (Opcional. Si no se incluye se devuelven independientemente de la fecha de publicación)
	n	Número máximo de elementos a descargar. (Opcional. Últimas 1000 noticias por defecto)
Respuesta JSON con array de todas las noticias correspondientes	*id*	ID de noticia
	title	Título de noticia
	url	URL de noticia
	author	Autor de la noticia
	content	Contenido HTML de la noticia
	created	Timestamp de publicación
	tags	Array con etiquetas aplicadas por el usuario a la noticia correspondiente (read, saved)

5.1.6 Servicio

El servicio supone el núcleo de la plataforma ya que es el que dota de la garantía de actualización a la aplicación. Sin él, las fuentes no podrían ser leídas y no se podrían enviar las noticias a los diferentes usuarios.

Técnicamente, se trata de una tarea CRON[8] programada para ejecutar la terminación *service / core* cada 10 minutos.

Este método proporciona una ejecución rápida y sin esperas. Al ejecutarse, fuerza la desconexión con el responsable de la ejecución y lleva a cabo los procesos de lectura internos. Estos procesos consisten en la lectura de todas las fuente RSS registradas en la base de datos sin

excepción y la llamada a *service / fetch* de manera concurrente para mejorar el rendimiento del proceso total. En esta llamada se proporciona la información correspondiente a cada fuente para que se proceda a su lectura.

Este último método viene implementado en gran parte por la librería SimplePie, que a partir de la dirección de la fuente que se le pasa, es capaz de realizar el parseo o lectura del archivo XML correspondiente y obtener una lista de todas las noticias disponibles, que son registradas en la tabla *Items* de la base de datos.

Es importante destacar que, debido al carácter privado de estos métodos, se ha optado por utilizar un secreto compartido a la hora de ejecutar estos métodos, de manera que si no se proporciona la cadena alfanúmerica adecuada, estos métodos no realizarán ningún tipo de acción.

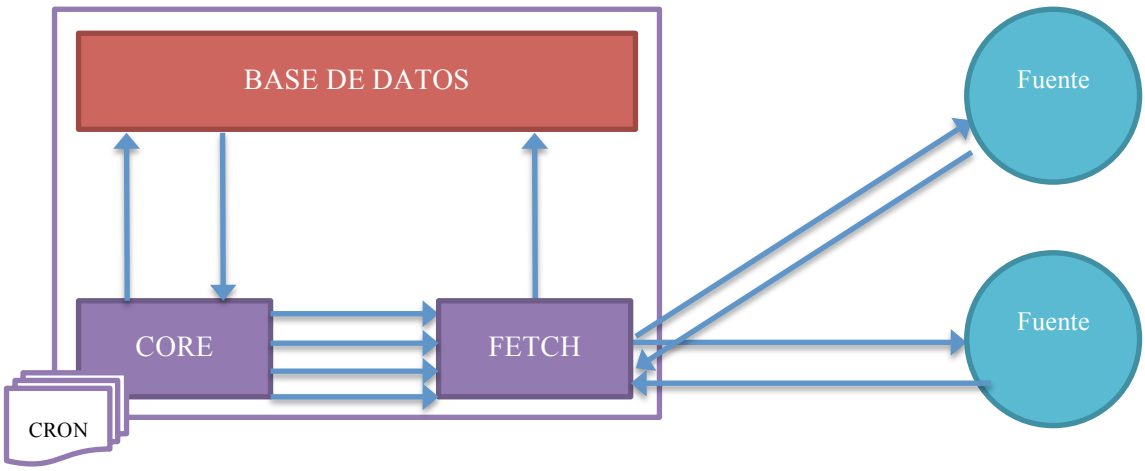


Figura 3. Servicio.

5.1.7 Mantenimiento

Adicionalmente, en *service / maintenance* se puede encontrar un método que puede programarse mediante otra tarea CRON para realizarse una vez al mes. Su finalidad es la de eliminar todos aquellos elementos cuya fecha de publicación es mayor a un mes, que ya se pueden considerar antiguos y es poco probable que los usuarios los lean. Con esto, se consigue un cierto mantenimiento de la base de datos, reduciendo su tamaño y librándola de elementos no necesarios.

5.2 Cliente

5.2.1 Interfaz de usuario

Para que cualquier usuario pueda utilizar la plataforma es necesario disponer de una herramienta con la que poder realizar todo tipo de gestiones, en cuánto a noticias se refiere. Por ello, se ha decidido desarrollar una aplicación para iPhone complementariamente al servicio de backend ya comentado.

Se trata de una aplicación basada en dos pestañas principales, una para las nuevas noticias pendientes por leer y otra para las noticias guardadas por el usuario. Cada una de estas pestañas alberga un control de navegación, con su respectiva pila de vistas, para navegar entre suscripciones y noticias.

Sin embargo, antes de acceder a estas vistas es necesario realizar el inicio de sesión o registro correspondiente para poder disfrutar del servicio.

En cuánto al diseño se refiere, la interfaz de usuario de la aplicación ha sido desarrollada únicamente mediante elementos nativos del sistema, como lo son la mayoría de iconos utilizados en la aplicación, barras de navegación, barras de herramientas, barras de selección de pestañas, tablas, vistas de navegación web… La aplicación se rige por un diseño basado en un único tono de azul y diferentes tonalidades de gris que hacen de la interfaz una salida visual agradable y simple para el usuario incluso mediante realiza una lectura de sus noticias.

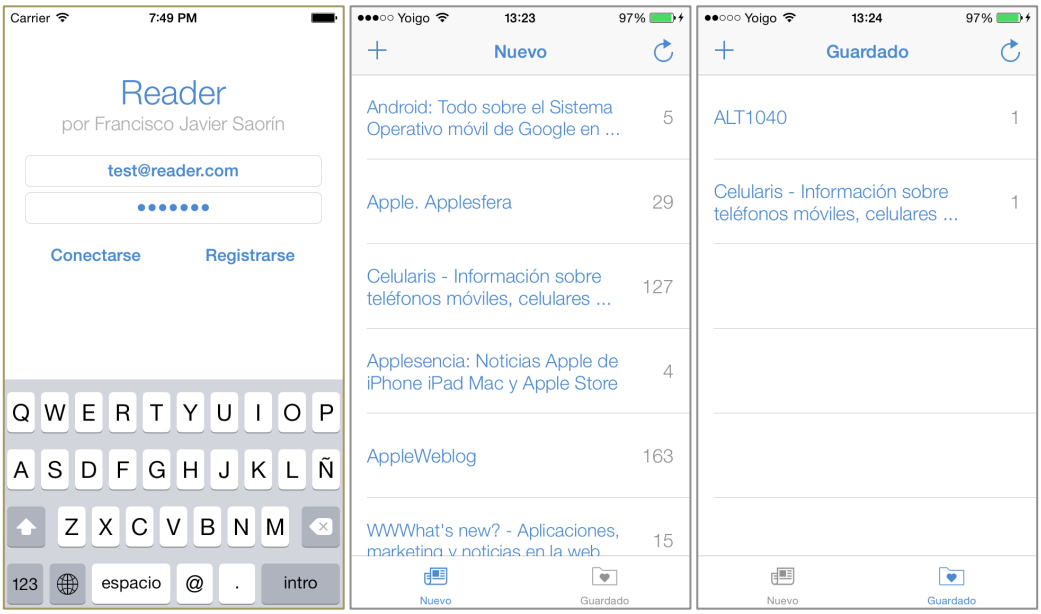


Figura 4. Vista inicio de sesión y suscripciones.

La primera vista en la pila de navegación de cada una de las pestañas se presenta como una lista de suscripciones en las que se indica el nombre de cada suscripción y el número de elementos incluidos en la misma. De esta manera, pulsando sobre cualquiera de ellas se accede a los elementos correspondientes.

Adicionalmente, en la parte superior, se dispone de 2 botones, uno para acceder a una vista de gestión de suscripciones (izquierda) y un botón de refresco (derecha).

Independientemente de la pestaña activa, al pulsar sobre una suscripción se accede a los elementos de dicha suscripción, pero a los elementos pendientes por leer o guardados, dependiendo de la pestaña.

Figura 5. Vista de elementos de una suscripción.

Se dispone así de una lista de los elementos de la suscripción seleccionada con su respectivo título, fecha e imagen destacada. Dicha imagen no es proporcionada por ningún servicio si no que es extraída del contenido HTML del artículo.

Una vez seleccionada una noticia, se dispone del acceso a las acciones relaciones con cada uno de los elementos. Mediante un navegador web integrado permite mostrar el contenido HTML de los mismos.

Por otro lado, con ayuda de los botones superiores permite por un lado marcar los elementos como guardado (botón derecho) y por otro compartir la noticia (botón izquierdo) en otras aplicaciones y redes sociales como pueden ser Twitter y Facebook, compartiendo en este caso el título y la URL de la noticia.

Figura 6. Vista de contenido y de gestión de suscripciones.

Por último, la aplicación dispone de un apartado en el que el usuario tiene a su disposición un listado de las suscripciones activas que tiene en ese momento. Desde esa lista puede eliminar su suscripción a una determinada fuente RSS.

Por otro lado, en la parte superior, dispone de un campo de texto en el que puede introducir determinadas palabras, título o directamente la URL del sitio web al que quiere suscribirse. Si el feed existe, la app añadirá la nueva suscripción a su lista de activas para eliminarla cuando lo desee.

Si se realiza alguna modificación en las suscripciones, la aplicación pedirá al servidor la lista de nuevas noticias para que, en el caso de que la suscripción añadida ya estuviera dada de alta, pueda recibir sus elementos al momento, sin esperar a la próxima actualización por parte del servidor.

5.2.2 Desarrollo

Antes de comenzar el desarrollo de la aplicación se realizó un estudio sobre que plataforma sería la más adecuada para ofrecerle al usuario una aplicación con la que poder gestionar y consumir las noticias.

En un primer momento, se pensó en ofrecer la aplicación en un sitio web en el servidor para tener así integrada todo el servicio en un mismo lugar. Esta aplicación estaría basada también en PHP y presentaría la información mediante HTML.

Sin embargo, es evidente la gran popularidad de la que disponen las aplicaciones móviles, ya que van allá dónde va el usuario mediante su Smartphone. Esta movilidad hace de las aplicaciones la mejor plataforma para el consumo de noticias ya que el usuario puede verse

tentado a consumir los contenidos en momentos de espera como pueden ser los viajes en transporte público por la ciudad. Además, la integración con otras apps del dispositivo permite una gran difusión de la noticia en las redes sociales. Por estas y otras razones se decidió desarrollar la aplicación para una plataforma móvil como Android, iOS o Windows Phone.

A la hora de decidir la plataforma móvil sobre la que trabajar se tuvieron en un primer momento factores de disponibilidad, quedando en este caso Windows Phone descartado al no conocer la plataforma ni sus limitaciones y por otro lado al no disponer de ningún dispositivo con la que realizar las pruebas de campo.

Android supone una alternativa muy flexible a la hora de desarrollar una aplicación para la plataforma, permitiendo llevar a cabo el desarrollo no sólo de aplicaciones sino también de widgets, servicios... En este caso, se podría aprovechar incluso la oportunidad de elaborar un servicio que recopile las noticias del servidor periódicamente para que el usuario disponga de las mismas desde el primer momento que accede a la aplicación.

Sin embargo, Android cuenta con la limitación de la fragmentación, se requerirían librerías de compatibilidad en el caso de que se quisiera dar soporte a versiones antiguas que aún siguen con una base de usuarios activa y sería necesario un desarrollo extra de la interfaz de usuario para cada rango de pantallas disponible en Android.

Por otro lado, iOS ofrece una alternativa cerrada, propietaria, no económica (se necesita abonar 99$ al año para ser desarrollador), con mejores herramientas de desarrollo en algunos aspectos y la posibilidad de tener un producto funcional en relativamente poco tiempo, al no tener que tratar con decenas de tamaños de pantalla y tener a disposición un gran SDK que reduce las dificultades que puedan presentarse a la hora de procesar los datos.

Además, cabe destacar que se tiene una amplia experiencia en el campo del desarrollo de aplicaciones en iOS, por lo que, aprovechando que se conoce la plataforma y se dispone de licencia de desarrollo activa, se decide finalmente llevar a cabo el desarrollo de la aplicación para iOS. En concreto, la app se ha desarrollado mediante el IDE Xcode de Apple para cualquier iPhone o iPod Touch con iOS 7 o superior. Cabe destacar, que en esta plataforma, la base de usuarios que utiliza versiones del sistema diferentes a la última es mínima al poderse disfrutar una actualización automáticamente a través de casi todos los dispositivos [9].

El esquema general de los componentes de la aplicación es el siguiente:

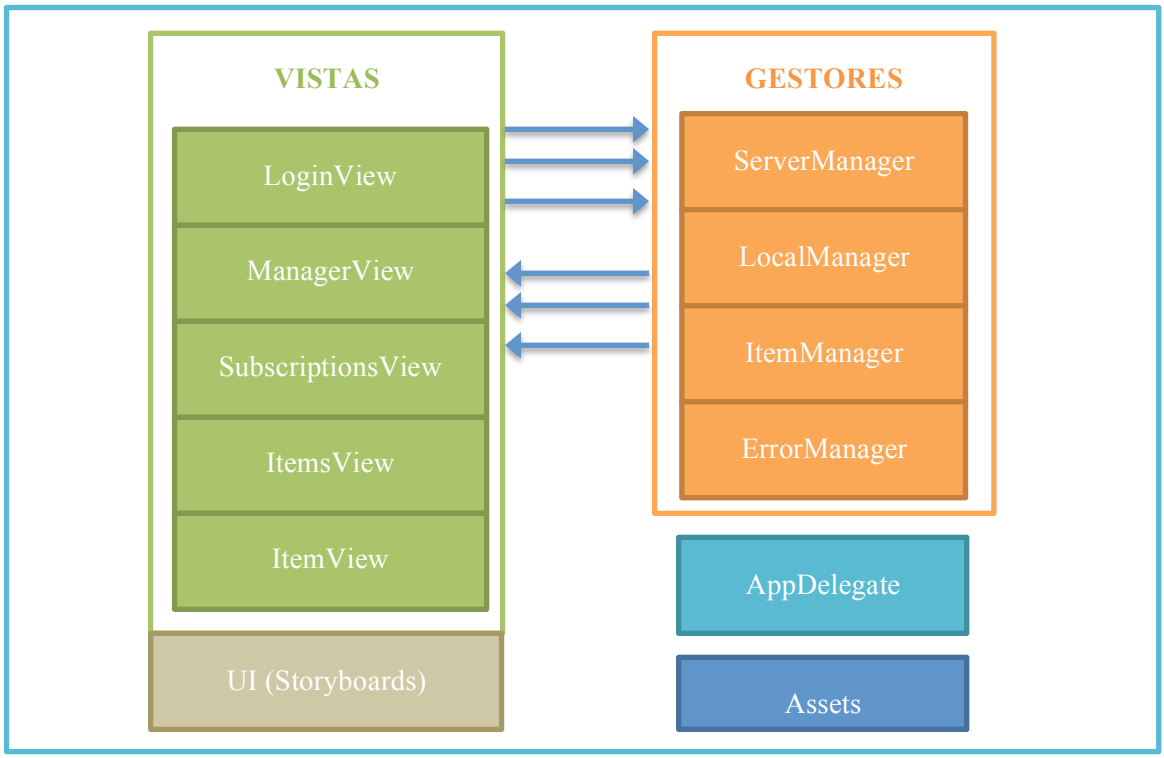


Figura 3. Componentes de la aplicación.

Por un lado, se encuentran los controladores de cada una de las vistas especificadas en el apartado anterior.

Vistas	
LoginView	Vista de inicio de sesión
ManagerView	Vista de gestión de suscripciones
SubscriptionsView	Vista de suscripciones
ItemsView	Vista de elementos de suscripción
ItemView	Vista de contenido

Y por otro lado, se encuentran los gestores, clases singleton[10] encargadas de realizar las conexiones pertinentes para la obtención y envío de datos desde la aplicación, gestión de datos, procesamiento de los mismos, gestión de errores…

Las funciones de cada uno de estos gestores vienen definidas en la siguiente tabla:

Gestores	
ServerManager	Peticiones a la API
LocalManager	Actualización de datos
	Lectura y clasificación de elementos
	Eliminación e inserción de elementos
	Escritura en disco
ItemManager	Obtención de fecha a partir de timestamp
	Obtención de imagen destacada a partir de la lectura del contenido HTML
ErrorManager	Presentación y tratamiento de errores

Cabe destacar que para la gestión y procesamiento de datos se ha usado una de las principales librerías de iOS, la librería Foundation [11] , con la que los datos pueden muy fácilmente tratados desde objetos JSON y posteriormente convertidos a diccionarios (NSDictionary) de objetos clave-valor y vectores (NSArray), con lo que la lectura y clasificación de datos se realiza de una manera sencilla y rápida. Además, al ser clases propias del sistema permiten la persistencia de datos mediante el guardado de los mismos en disco y su posterior carga en el mismo tipo de objetos.

Adicionalmente, se han utilizado las siguientes librerías de terceros:

Librerías externas	
MBProgressHUD [12]	Indicador de carga en caja negra
SDWebImage [13]	Carga eficiente de imágenes web

Capítulo 6. Conclusiones y propuesta de trabajo futuro

6.1 Conclusiones

Como se ha podido observar durante el desarrollo del trabajo, actualmente se cuenta con tecnología suficiente y al alcance de todos, con la que imitar en gran medida el funcionamiento de plataformas tan grandes y potentes como Google Reader.

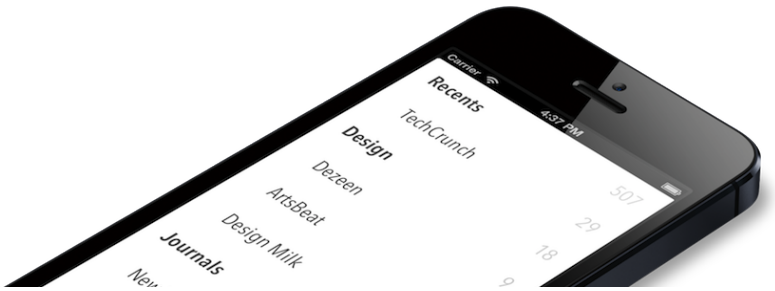
Con este trabajo se ha podido afianzar conceptos sobre el desarrollo de servicios web que permiten gestionar bases de datos a través de diferentes ficheros o scripts alojados en un servidor. Además se han podido desarrollar actitudes propias de administrador de sistemas, al desarrollar un servicio periódico en el servidor mediante trabajos CRON.

En cuánto a la aplicación, se ha conseguido reforzar las aptitudes en el desarrollo de la plataforma iOS en conjunto con otras tecnologías como JSON o HTML, al permitir la lectura y envío de información al servidor y su consiguiente presentación al usuario.

Por su parte, RSS sigue liderando el mercado en cuánto a lo que difusión de noticias en tiempo real se refiere, sin embargo, es inevitable pensar que Google abandonó uno de sus plataformas estrella, como lo indicó el enfado por parte de los millones de usuarios con los que contaba. Una plataforma abandonada en favor a una red social, Google+ en este caso, que poco a poco va contando con más integrantes y en la que son los propios usuarios los encargados de difundir las noticias más interesantes.

Quizás sea pronto para Google+, pero otras redes sociales más instantáneas y maduras como Twitter han conseguido que muchos usuarios abandonen los tradicionales clientes RSS debido a la incorporación de sus publicaciones favoritas a esta red [14] . Los usuarios ahora siguen y no se suscriben, teniendo un trato más directo con las publicaciones y su contenido a la vez que pueden compartirlas con un simple click. Por eso, ahora más que nunca, el futuro de RSS está aún por determinar ya que no cuenta, por sí sola como tecnología, de las características sociales tan demandadas hoy en día.

6.2 Espresso Reader



En el caso del presente trabajo, hablar de una propuesta de trabajo es hablar de un producto comercial existente actualmente. Este trabajo nació en 2013, cuando Google abandona Google Reader, el agregador de noticias más famoso entonces y del que se alimentaban infinidad de aplicaciones web, de escritorio y móviles.

Una de esas aplicaciones móviles era Espresso Reader [15] , la cuál estaba disponible desde hacía meses en la App Store de Apple. Se trataba de un cliente RSS basado en Google Reader y de funcionamiento idéntico al de este trabajo. El abandono de la plataforma supuso que la aplicación quedase inútil ante este cambio, por ello era necesario buscar alternativas.

Durante los primeros meses tras el anuncio de Google de la futura retirada, no existía ningún servicio maduro y completo que ofreciera una API abierta a los desarrolladores para llevar a cabo aplicaciones que permitieran a los usuarios disfrutar de sus noticia en tiempo real.

Es así como nace este trabajo, como alternativa a su predecesor y para poder ofrecer funcionalidad a Espresso Reader, que ya contaba con miles de usuarios.

Su funcionamiento y desarrollo fue bastante aceptable con miles de suscripciones y una primer grupo de usuarios seleccionados. Sin embargo, una vez superadas las primeras etapas del desarrollo, aparecía en el mercado la que actualmente se ha convertido en el sustituto definitivo de Google Reader, Feedly. Su primera API ofrecía el mismo conjunto de funciones y datos que el de Google, para ahorrar recursos y hacer la migración lo más transparente posible a los desarrolladores, que para entonces era unos pocos que habían sido seleccionados para probar una futura API pública. Entre ellos el desarrollador del presente trabajo y Espresso Reader. Fue así como este cliente RSS para iPhone sobrevivió y pudo alimentarse de Feedly para continuar en el mercado.

Por tanto, Google Reader y Feedly han servido de base e inspiración para este trabajo, que ha posibilitado el desarrollo de las primeras versiones de un producto comercial que ahora mismo es una realidad y cuenta con más de 10.000 usuarios y en la que se sigue trabajando y añadiendo mejoras y características con regularidad.

Capítulo 7. Anexos

7.1 SimplePie

7.1.1 Descripción

Una parte importante del núcleo de este trabajo es realizada por la librería PHP SimplePie.

La API ofrecida por esta librería gestiona todo lo relacionado con la obtención, caché, lectura y normalización de las estructuras de datos entre formatos, encargándose de la transcodificación de caracteres y saneando los datos resultantes.

SimplePie es un software open-source gratuito desarrollado y mejorado por personas que han querido ofrecer un buen software para hacer la vida más fácil a la gente.

Se trata de una librería muy bien documentada mediante una completa referencia a la API, tutoriales y screencasts pedidas por parte de los usuarios.

Así mismo, es una solución dónde se ha trabajado duro para que las personas que quieran utilizarla lo hagan de la manera más fácil, siempre que tengan conocimiento de PHP.

Sus principios son: rapidez, facilidad de uso, compatibilidad y estandarización.

Alternativas sobre las que destaca: MagpieRSS [16] , SimpleXML [17] , Universal Feed Parser [18] y Google AJAX Feed API (utilizada en este trabajo como complemento)

7.1.2 Compatibilidad

Versiones RSS y Atom	Elementos de espacios de nombre
RSS 0.90	*Dublin Core 1.0*
RSS 0.91 (Netscape)	*Dublin Core 1.1*
RSS 0.91 (UserLand)	*GeoRSS*
RSS 0.92	*iTunes RSS 1.0*
RSS 1.0	*Media RSS 1.1.1*
RSS 2.0	*RSS 1.0 Content Module*
Atom 0.3	*W3C WGS84 Basic Geo*
Atom 1.0	*XML 1.0*
	XHTML 1.0

Capítulo 8. Bibliografía

[1] Wikipedia, “RSS (Rich Site Summary)” http://en.wikipedia.org/wiki/RSS [Online].

[2] Wikipedia, “News aggregator” http://en.wikipedia.org/wiki/News_aggregator [Online].

[3] Google Reader Team, “A final farewell” http://googlereader.blogspot.com.es/2013/07/a-final-farewell.html [Online].

[4] Wikipedia, “RSS-DEV Working Group” http://en.wikipedia.org/wiki/RSS-DEV_Working_Group [Online].

[5] Dublin Core, “Dublin Core” http://dublincore.org/ [Online].

[6] Ryan Parman, “SimplePie Documentation. Learn how to use this thing. It’s way better than going to school.” http://simplepie.org/wiki/ [Online].

[7] Sourceforge, “PHPMyAdmin. Bringing MySQL to the web” http://sourceforge.net/projects/phpmyadmin/ [Online].

[8] Wikipedia, “CRON” http://en.wikipedia.org/wiki/Cron [Online].

[9] AppleInsider, “Ahead of iOS 8 announcement iOS 7 adoption nears 90% for iPhone, 85% for iPad” http://appleinsider.com/articles/14/05/30/ahead-of-ios-8-announcement-ios-7-adoption-nears-90-for-iphone-85-for-ipad- [Online]

[10] Matt Galloway, “Singletons in Objective-C“ http://www.galloway.me.uk/tutorials/singleton-classes/ [Online]

[11] Apple, “Foundation Framework Reference” https://developer.apple.com/library/ios/documentation/cocoa/reference/foundation/ObjC_classic/_index.html [Online]

[12] Jonathan George, “MBProgressHUD” https://github.com/jdg/MBProgressHUD [Online]

[13] Bogdan Poplauschi, “SDWebImage” https://github.com/rs/SDWebImage [Online]

[14] Jesse Wojdylo, “Why Google+ is the New Google Reader” http://www.googleplusdaily.com/2013/03/why-google-is-new-google-reader.html [Online]

[15] Javier Lacort, “App del día: Espresso Reader (iOS)” http://www.celularis.com/imagen-del-dia/espresso-reader/ [Online]

[16] Kellan, “MagpieRSS” https://github.com/kellan/magpierss [Online]

[17] PHP, “SimpleXML” http://php.net/manual/es/book.simplexml.php [Online]

[18] f8dy, “Universal Feed Parser” http://sourceforge.net/projects/feedparser/ [Online]