

# AppTFG

**Jaime Ramos Romero  
Javier Bastarrica Lacalle**

**Grado en Ingeniería Informática  
Facultad de Informática**

UNIVERSIDAD COMPLUTENSE DE MADRID



**Trabajo de Fin de Grado  
Madrid, Junio 2015**

Directores:  
Samer Hassan Collado  
Pablo Ojanguren





# Autorización de Difusión y Uso

Autorizo a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor, tanto la propia memoria, como el código, la documentación y/o el software desarrollado.

Jaime Ramos Romero  
Javier Bastarrica Lacalle

Madrid, Junio 2015

Copyleft by Jaime Ramos Romero and Javier Bastarrica Lacalle, released under the license Creative Commons Attribution Share-Alike International 4.0 available at:

<https://creativecommons.org/licenses/by-sa/4.0/>



# Índice general

|                                                      | Page       |
|------------------------------------------------------|------------|
| <b>Autorización de Difusión y Uso</b>                | <b>III</b> |
| <b>Abstract</b>                                      | <b>IX</b>  |
| <b>Resumen</b>                                       | <b>XI</b>  |
| <b>1. Introducción</b>                               | <b>1</b>   |
| 1.1. Objetivos del Proyecto . . . . .                | 1          |
| 1.2. Estructura del Documento . . . . .              | 1          |
| <b>2. Migración de Wave a Android</b>                | <b>3</b>   |
| 2.1. Estado del Arte . . . . .                       | 3          |
| 2.2. Wave . . . . .                                  | 3          |
| 2.2.1. Google Wave . . . . .                         | 3          |
| 2.2.2. Apache Wave . . . . .                         | 3          |
| 2.3. Tecnologías y Características de Wave . . . . . | 4          |
| 2.3.1. Características de Wave . . . . .             | 4          |
| Federación . . . . .                                 | 4          |
| Consistencia en tiempo real . . . . .                | 4          |
| Escalabilidad . . . . .                              | 5          |
| Modelo Wave . . . . .                                | 5          |
| 2.3.2. Servidores Wave . . . . .                     | 6          |
| Wave in a Box . . . . .                              | 6          |
| SwellRT . . . . .                                    | 7          |
| 2.4. Metodología de Migración . . . . .              | 8          |

|           |                                                                        |           |
|-----------|------------------------------------------------------------------------|-----------|
| 2.4.1.    | Objetivo . . . . .                                                     | 8         |
| 2.4.2.    | Plataforma: Entorno de Desarrollo, Construcción y Depuración . . . . . | 8         |
|           | Eclipse . . . . .                                                      | 8         |
|           | Proceso de Construcción por Consola . . . . .                          | 11        |
|           | Proceso de Depuración . . . . .                                        | 13        |
| 2.4.3.    | Migración: Identificación y Solución de Problemas . . . . .            | 14        |
|           | Conexion HTTP . . . . .                                                | 15        |
|           | Conexión WebSocket . . . . .                                           | 18        |
|           | Login y Logging . . . . .                                              | 22        |
| 2.4.4.    | Organización y Resultados . . . . .                                    | 23        |
|           | Servicio Android . . . . .                                             | 23        |
|           | Resultado de la Migración . . . . .                                    | 24        |
|           | Diagramas y Dependencias . . . . .                                     | 25        |
| <b>3.</b> | <b>Construyendo la idea: DemoCritics</b>                               | <b>27</b> |
| 3.1.      | Introducción . . . . .                                                 | 27        |
| 3.1.1.    | Política en el mundo de la Informática . . . . .                       | 29        |
| 3.1.2.    | Democracia . . . . .                                                   | 30        |
|           | Democracia representativa . . . . .                                    | 30        |
|           | Democracia participativa . . . . .                                     | 30        |
|           | Democracia directa . . . . .                                           | 30        |
|           | Democracia deliberativa . . . . .                                      | 30        |
| 3.1.3.    | Adentrándonos en la idea . . . . .                                     | 30        |
| 3.2.      | Estado del Arte . . . . .                                              | 31        |
| 3.2.1.    | Programas Políticos . . . . .                                          | 31        |
|           | UPyD Parla . . . . .                                                   | 31        |
|           | #RecuperaCórdoba . . . . .                                             | 32        |
|           | PP Canarias . . . . .                                                  | 33        |
| 3.2.2.    | Participación Ciudadana . . . . .                                      | 34        |

|                                             |           |
|---------------------------------------------|-----------|
| reddit . . . . .                            | 34        |
| Change.org . . . . .                        | 35        |
| Programa Colaborativo AhoraMadrid . . . . . | 37        |
| <b>4. Resultados y Conclusiones</b>         | <b>39</b> |
| 4.1. Discusion de Resultados . . . . .      | 39        |
| 4.1.1. Evaluación con usuarios . . . . .    | 39        |
| 4.2. Conclusiones . . . . .                 | 39        |
| <b>5. Trabajo a Futuro</b>                  | <b>41</b> |
| 5.1. Mejoras . . . . .                      | 41        |
| <b>Bibliografía</b>                         | <b>41</b> |

# Índice de figuras

|      |                                                               |    |
|------|---------------------------------------------------------------|----|
| 2.1. | Modelo Conversacional de Wave . . . . .                       | 6  |
| 2.2. | Cliente Wave In A Box . . . . .                               | 7  |
| 2.3. | Distribución Actual de Versiones Android (Fuente: Google) . . | 10 |
| 2.4. | Emulador Android API 19 . . . . .                             | 12 |
| 2.5. | Ejemplo de Traza de Error en Logcat . . . . .                 | 16 |
| 2.6. | Pantalla de Login de WaveAndroid . . . . .                    | 22 |
| 3.1. | Brainstorming sobre la idea a desarrollar . . . . .           | 28 |
| 3.2. | UPyD Parla . . . . .                                          | 32 |
| 3.3. | #RecuperaCórdoba . . . . .                                    | 33 |
| 3.4. | PP Canarias . . . . .                                         | 34 |
| 3.5. | Plaza Podemos utilizando reddit . . . . .                     | 35 |
| 3.6. | Change.org · La mayor plataforma de peticiones del mundo . .  | 36 |
| 3.7. | Creación colaborativa del programa de Ahora Madrid. . . . .   | 37 |

# Índice de cuadros

|      |                                |    |
|------|--------------------------------|----|
| 2.1. | Technologies Summary . . . . . | 25 |
|------|--------------------------------|----|

# Abstract

Abstract en inglés.

## **Keywords:**

Apache Wave, Collaboration, Android, Apps, Real Time, Politics



# Resumen

Abstract en español.

**Palabras Clave:**

Apache Wave, Colaboración, Android, Apps, Tiempo Real, Política



# **Capítulo 1**

## **Introducción**

### **1.1. Objetivos del Proyecto**

Los objetivos son los siguientes:

- 1<sup>a</sup> parte: Migración de Wave a Android
  - Estudiar la implementacion actual de Wave en Java y GWT.
  - Adaptar la implementación para hacer uso de las caracteristicas nativas de Android.
- 2<sup>a</sup> parte: Creación de una aplicación Android
  - Evaluar posibles ideas de aplicación y estudiar su viabilidad.
  - Evaluar con usuarios su usabilidad.
  - Implementar la aplicación.
  - Testear y evaluar con usuarios el resultado.

### **1.2. Estructura del Documento**

- Capítulo 2 - Migración de Wave a Android: Descripcion de la tecnologia Wave y de la Metodología utilizada para la migración a Android.
- Capítulo 3 - AppTFG:
- Capítulo 4 - Resultados y Conclusiones:
- Capítulo 5 - Trabajo Futuro:



# **Capítulo 2**

## **Migración de Wave a Android**

### **2.1. Estado del Arte**

— PENDIENTE —

### **2.2. Wave**

#### **2.2.1. Google Wave**

Ideado y presentado en 2009 por ingenieros de Google [1], Wave es a la vez un protocolo de comunicaciones [2] y una plataforma web de código libre, que permiten a sus usuarios comunicarse y colaborar entre sí en tiempo real (Ver sección 2.3.1) y de forma federada (Ver sección 2.3.1) a través de Internet. Inicialmente fue desarrollado con el objetivo de integrar en una sola plataforma servicios ampliamente utilizados como son el correo electrónico, las redes sociales y la mensajería instantánea. Pese al gran entusiasmo generado entre la comunidad de desarrolladores tras su anuncio, en el año 2010 Google anuncia el abandono del proyecto [3] debido a su poca acogida entre los desarrolladores y a que decide reorientar el uso de la tecnología hacia sus plataformas de edición de documentos Google Docs [4] y a su red social Google + [5]. Es en este momento cuando el desarrollo libre del proyecto pasa a manos de la Apache Software Foundation bajo el nombre de Apache Wave.

#### **2.2.2. Apache Wave**

Al cambiar de manos su desarrollo en 2010, la tecnología pasa a formar parte de la incubadora de la fundación Apache [6] como software de código libre bajo licencia Apache [7]. Así, se produce el desarrollo de Wave In a Box

(WIAB) (Ver sección ??), plataforma que integra un cliente web sencillo y una implementación de un servidor Wave que cualquiera puede descargar y desplegar en su ordenador.

## 2.3. Tecnologías y Características de Wave

### 2.3.1. Características de Wave

Como plataforma de código libre desarrollada para ser utilizada en red, Wave hace uso de distintas tecnologías y protocolos bien conocidos. Entre sus características más destacadas están las siguientes:

#### Federación

El Protocolo Wave [2] fue desarrollado para utilizar un modelo federado [8] [9] de comunicación basado en la tecnología XMPP [10] [2]. Se trata por tanto de un modelo descentralizado en el que cualquiera de los participantes en la conversación es libre de actuar tanto como servidor como cliente sin que ello afecte a su participación en la conversación. Además, a diferencia de otras tecnologías (como el correo electrónico) en las que cada participante almacena su propia copia de la conversación y cada vez que hay cambios se debe transmitir la conversación entera a todos los participantes, Wave tiene la ventaja de que actúa de forma que es el servidor de la conversación el único que almacena la copia entera y se encarga de calcular los cambios que se han producido para transmitir solamente dichos cambios por la red a los participantes, con las consiguientes ventajas en términos de latencia que ello conlleva.

#### Consistencia en tiempo real

El Protocolo Wave [2] utiliza la tecnología de Transformaciones Operacionales (OT) [11] para garantizar la consistencia en la comunicación en tiempo real entre los participantes. Es decir, cualquier cambio producido por cualquiera de los participantes en la conversación se transmite automáticamente y en tiempo real al resto de los participantes sin pérdida de información y garantizando que los cambios se muestran en el estricto orden en el que se produjeron sin errores [12].

### Escalabilidad

Wave fue desarrollado como un protocolo de alta escalabilidad que permite gestionar la existencia de una gran cantidad de conversaciones y participantes sin que por ello se resienta la productividad del sistema.

### Modelo Wave

Además de definir el protocolo del que hace uso Wave, Google definió un Modelo de Datos Conversacional [13] que refleja la arquitectura de los datos que componen las conversaciones en Wave. Así, a grandes rasgos, podemos ver dichas conversaciones como documentos XML sobre los que los usuarios participantes (cualquiera es libre de unirse a una conversación en cualquier momento) actúan creando nuevos elementos o modificando los ya existentes. Este modelo de datos define una nomenclatura propia para los elementos que componen esta tecnología [14] [9]:

- **Wave:** Conjunto de wavelets (conversaciones).
- **Wavelet:** conjunto de documentos de una conversación y sus participantes.
- **Blip:** documento con el contenido de un mensaje en la conversación. Un blip puede tener otros blips dentro de él y los blips pueden ser publicados o no en función de si su visibilidad se extiende o no al resto de participantes de la conversación respectivamente.
- **Manifiesto conversacional:** documento con metadatos que definen la estructura de una conversación.
- **Hilo conversacional:** conjunto de Blips consecutivos que forman parte de una conversación.
- **Extensiones** [15]: pequeñas aplicaciones que se ejecutan dentro de una Wave y aportan nuevas funcionalidades que no forman parte del modelo conversacional básico. Pueden ser de dos tipos:
  - **Gadget:** aplicación que se ejecuta en el contexto de una Wave y en la que todos sus usuarios participan.
  - **Robot:** aplicación que participa en una Wave a modo de usuario automatizado e interactúa con el contenido pudiendo modificarlo y responder a eventos por acciones de otros usuarios reales.

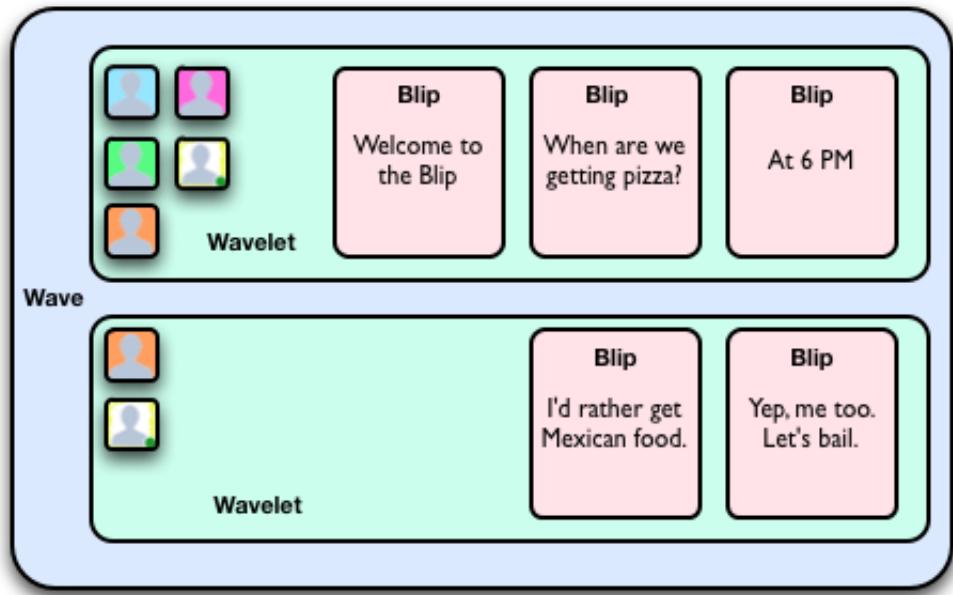


Figura 2.1: Modelo Conversacional de Wave

### 2.3.2. Servidores Wave

#### Wave in a Box

Wave In a Box (WIAB) [16] es el nombre de la implementación de un servidor Wave desarrollado por la Apache Software Foundation tras pasar el proyecto a sus manos en el año 2012. Al igual que el resto del código de la tecnología que heredó de Google, está implementado en Java usando OpenJDK [17]. La instalación trae consigo un cliente web desarrollado en Javascript usando el framework Google Web Toolkit [18]. Este cliente web sirve como prueba de concepto de las funcionalidades básicas del Modelo Conversacional de Wave, pudiendo gestionar waves, usuarios y extesiones. Actualmente cualquiera puede descargar y desplegar WIAB en su ordenador siguiendo los pasos que nos proporcionan en su wiki [19]. La aplicación se distribuye en forma de código fuente, accesible entre otras formas desde su repositorio de GitHub [20]. Existen asimismo servidores de prueba ya desplegados en Internet sobre los que se puede observar el funcionamiento de WIAB [21].

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

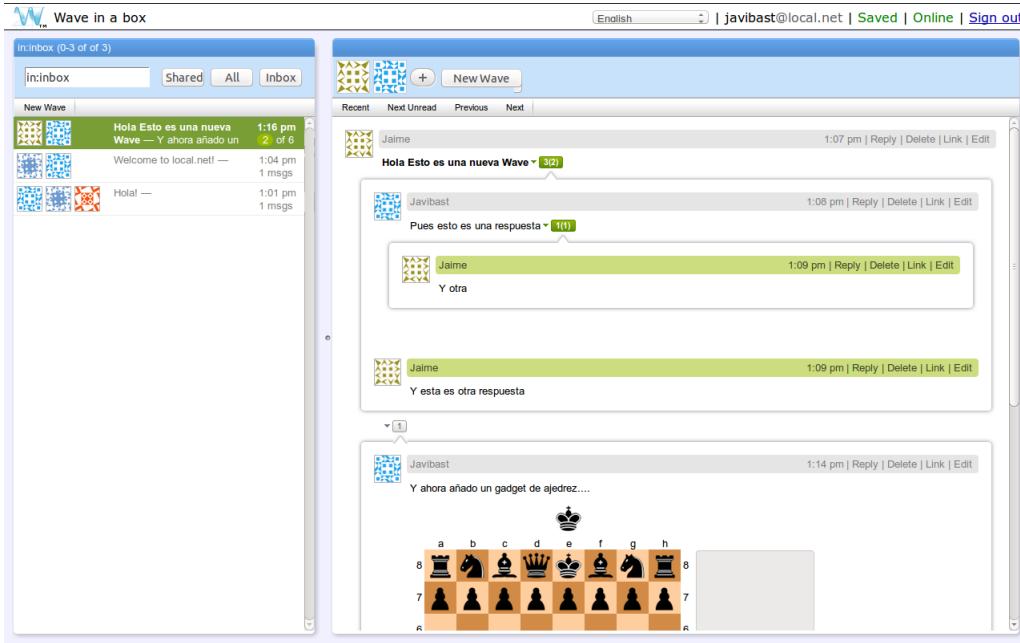


Figura 2.2: Cliente Wave In A Box

### SwellRT

Como parte del proyecto europeo P2PValue [22] existe SwellRT, un fork de WIAB que amplía las características de éste último añadiendo un nuevo modelo de datos (Modelo de Datos Colaborativo) más allá del Modelo de Datos Conversacional de Wave original. Proporciona también un API escrito en Java que permite trabajar sobre los datos de ese nuevo modelo en forma de tres tipos básicos: mapas, listas y strings. Es por tanto un framework de colaboración en tiempo real que basa su funcionamiento en Apache Wave y cuyo principal propósito es permitir la integración de la tecnología Wave en otras aplicaciones, que podrán compartir objetos (de los tipos antes mencionados) de forma federada y en tiempo real. Su código fuente está disponible en GitHub [23], así como sus instrucciones de instalación (Ver el Readme en GitHub).

Para este proyecto se ha usado el framework SwellRT como base para la migración de la tecnología de Apache Wave a la plataforma Android [24]. Se pretende con esto que SwellRT haga uso de las funcionalidades nativas de Android.

## 2.4. Metodología de Migración

### 2.4.1. Objetivo

El framework de SwellRT utiliza un servidor WIAB y el protocolo Wave, ambos desarrollados en Java. El **SDK de Android** [25] es compatible con Java, así que a priori la implantación del servidor no supone problemas en los dispositivos móviles. Sin embargo, existe un problema con el API de SwellRT, ya que el lado del cliente fue desarrollado en Javascript usando el framework GWT. Android no soporta de forma nativa estas tecnologías, así que es necesario estudiar el código de SwellRT para sustituir todo el código que haga uso de Javascript/GWT por código compatible con Android. El objetivo de esta parte del proyecto es conseguir que un cliente desplegado en Android sea capaz de conectarse e interactuar con un servidor Wave sin problemas.

### 2.4.2. Plataforma: Entorno de Desarrollo, Construcción y Depuración

Existen dos entornos de desarrollo (IDE) recomendados por Google para desarrollar en Android: Eclipse [26] y Android Studio.[27] Eclipse es un entorno de desarrollo genérico que, mediante plugins, permite extender sus funcionalidades para desarrollar en diversas plataformas y lenguajes. Android Studio es un IDE basado en el entorno de desarrollo Java IntelliJ IDEA [28] adaptado para trabajar con todas las funcionalidades de Android. En el momento de empezar con la migración Android Studio se encuentra en fase beta de desarrollo, pues Google pretende convertirla en el IDE de desarrollo oficial para Android. Mientras no se lanza la versión final de Android Studio, Google recomienda utilizar Eclipse para desarrollar en Android, y las guías para desarrolladores Android están escritas para Eclipse. En consecuencia tomamos la decisión de utilizar el entorno de desarrollo Eclipse para la migración de SwellRT a Android.

#### Eclipse

El IDE de Eclipse [26] soporta el desarrollo con Android a través del plugin **ADT (Android Development Tools)** [29], que integra en un solo paquete todas las herramientas necesarias para desarrollar, construir y depurar el

---

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

código de la aplicación fácilmente.

**Android SDK** [25]: paquete que integra el conjunto de herramientas necesarias para desarrollar en Android. Entre estas herramientas destacan las siguientes:

- **Librerías con el API** de Android y **Documentación** asociada [30]
- **Android Virtual Device Manager (AVDM)** [31] herramienta para gestionar la creación, modificación, ejecución y eliminación de emuladores en Android. Un **emulador** [32] es una máquina virtual que ejecuta una determinada versión de Android. Permite desplegar un dispositivo móvil en el ordenador que imita las características software y hardware de uno real para poder hacer pruebas de desarrollo sin necesidad de poseer un dispositivo con Android.
- **Android SDK Manager** [33] herramienta para gestionar las versiones de SDK y herramientas asociadas instaladas. Android se encuentra actualmente en la versión 5.1 (API 22), pero un desarrollador puede elegir desarrollar para una versión anterior si lo estima necesario, por lo que puede descargarse por separado dicha versión y mantener varias API si lo necesita.
- **Dalvik Debug Monitor Server (DDMS)** [34] herramienta que provee las características de entorno de depuración para las aplicaciones en desarrollo.

Teniendo en cuenta la distribución actual de versiones instaladas en dispositivos Android [35] (Ver figura 2.3) se ha decidido realizar la migración de SwellRT con el API 19 de Android (Versión 4.4 "KitKat"). El emulador desplegado para las pruebas de desarrollo utilizará por tanto Android 4.4 .

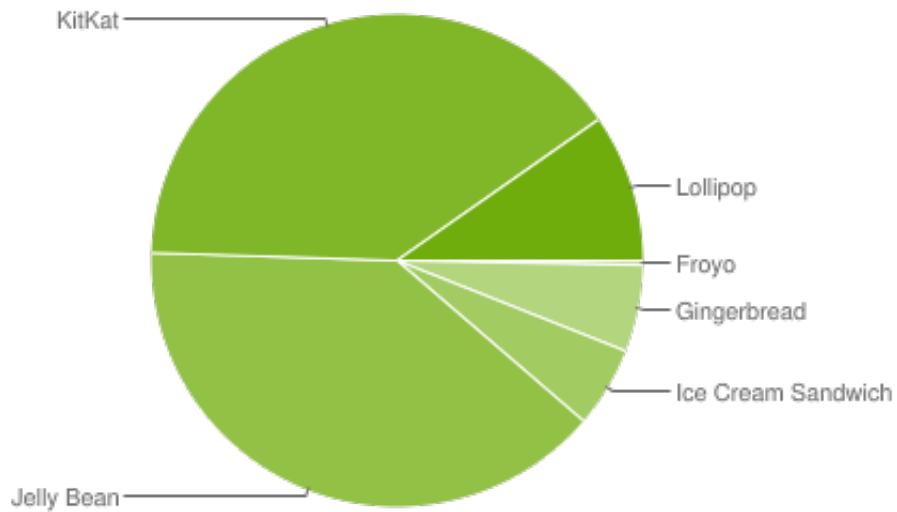


Figura 2.3: Distribución Actual de Versiones Android (Fuente: Google)

Sin embargo existe un problema con la construcción y depuración del código de SwellRT en Eclipse. Android **limita el número de métodos máximos de una aplicación a 65K** [36] por cuestiones de eficiencia. Para evitar esta limitación, durante el proceso de construcción el SDK de Android utiliza, entre otras, una herramienta llamada **ProGuard** [37]. Esta herramienta se encarga de optimizar el código de la aplicación buscando remover clases que no se utilizan y ofuscando el código para prevenir la ingeniería inversa. En el caso de SwellRT, el código posee un gran número de clases java necesarias para desplegar el servidor y el cliente de la herramienta, por lo que es necesaria dicha optimización de código realizada por ProGuard. El sistema de compilación de aplicaciones de Android tiene dos formas: compilación de la aplicación en modo debug (para hacer pruebas cuando todavía se encuentra en fase de desarrollo) y en modo release (la aplicación se encuentra en su versión final y se empaqueta y se firma digitalmente para lanzarla al público). En el caso de Eclipse, ProGuard solo se ejecuta cuando se construye en modo release, por lo que cuando se intenta compilar una aplicación con tantas clases como SwellRT mientras se desarrolla (modo debug) el sistema da error y no se puede compilar el código para probarlo en el emulador.

La solución que encontramos fue desarrollar en Eclipse (por las facilidades que el entorno proporciona para escribir código) pero realizar el proceso de construcción del código por consola de comandos, ya que en este caso sí que

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

---

se puede compilar la aplicación en modo debug utilizando ProGuard.

### Proceso de Construcción por Consola

Para construir la aplicación por consola de comandos, Android utiliza la herramienta Apache Ant [38] para automatizar el proceso de construcción [39]. Es importante asimismo tener definida la variable de entorno JAVA\_HOME con la ruta de acceso al JDK de java instalado en la máquina. Conviene también, por comodidad a la hora de trabajar con la consola, añadir al PATH del sistema las rutas a la carpeta donde esta el SDK de android (/sdk) y dentro de esta ruta añadir asimismo rutas a las carpetas /tools y /platform-tools.

Existen dos formas de realizar la construcción en modo debug de una app:

#### **1 - Sin tener previamente lanzado un emulador o conectado al ordenador un dispositivo android en modo debug [40]:**

En este caso es necesario construir la aplicación y luego lanzar el emulador para después instalar la aplicación en él. Para construir la aplicacion en modo debug nos vamos a la carpeta raíz de nuestro proyecto y ejecutamos el siguiente comando:

```
$ ant clean debug
```

Esto nos generará una aplicacion instalable en el directorio /bin del proyecto bajo el formato que Android usa para sus aplicaciones (.apk). El siguiente paso es ejecutar un emulador o conectar un dispositivo android por USB. Para ejecutar un emulador, abrimos otra consola y utilizamos el siguiente comando:

```
$ android avd
```

Lo que nos despliega la herramienta Android Virtual Device Manager (Ver Sección 2.4.2) para que elijamos/creemos el emulador que queremos ejecutar. Podemos elegir multitud de parámetros [?] para el dispositivo que emula (resolución y tamaño de pantalla, de memoria Ram, elementos hardware emulados, etc.) siendo lo más importante elegir un API (versión de Android) que se corresponda con el API que hemos elegido para nuestra aplicación (en nuestro caso API 19). Es recomendable también elegir una imagen del sistema que use un procesador con arquitectura Intel x86, ya que si elegimos

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

---

la opción por defecto de ARM (los dispositivos móviles actuales usan procesadores ARM) la ejecución del emulador se ralentiza mucho al tener que emular una arquitectura de procesador distinta a la suya (los ordenadores actuales usan arquitectura Intel x86 en su mayoría). Esto únicamente afecta al rendimiento del emulador, la aplicación es independiente de la arquitectura que haya por debajo.

Una vez lanzado el emulador/dispositivo móvil, procedemos a instalar la aplicación en él ejecutando el siguiente comando en la primera consola (en la que construimos la aplicación):

```
$ adb install XXXX.apk
```

Siendo XXXX la ruta a donde se encuentra el .apk de la aplicación que previamente hemos construido (/bin). La herramienta ADB (Android Debug Bridge) [41] es la que permite la comunicación entre el proceso de la consola de comandos y el emulador/dispositivo móvil. Es importante destacar que si se tienen varios emuladores/dispositivos móviles en ejecución/conectados hay que especificar en cual se quiere instalar la aplicación añadiendo al comando lo siguiente: **-s emulator -YYYY** siendo esto último el identificador del emulador que podemos encontrar en el título de la ventana del emulador.



Figura 2.4: Emulador Android API 19

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

---

De esta manera podemos probar la aplicación, que será lanzada en el emulador/dispositivo una vez termine su instalación.

**2 - Teniendo un emulador previamente lanzado (ver sección anterior para ver cómo se lanza) o un dispositivo móvil ya conectado por USB:**

En este caso es todavía más sencillo el proceso de construcción. Nos vamos a la carpeta raíz del proyecto y podemos compilar e instalar la aplicación con un solo comando:

```
$ ant debug install
```

Es importante destacar que este comando solo funciona si tenemos un único emulador o dispositivo conectado, de lo contrario habrá que utilizar el método anterior.

### Proceso de Depuración

Una vez instalada una aplicación, podemos depurar su código en ejecución usando la herramienta DDMS del ADT en conjunto con la vista de Debug de Eclipse. Pero antes hay que especificar qué aplicación queremos depurar de las que puedan estar instaladas en el dispositivo o emulador.

En el caso del emulador debemos lanzar la aplicación llamada “Dev Tools” y abrir el menú “Developer Options”. Dentro de este menú habilitaremos las opciones de “USB debugging” y de “Wait for debugger”. Además pulsaremos sobre “Select Debug app” y seleccionaremos la aplicación que queremos depurar.

En el caso de un dispositivo Android debemos ir a los Ajustes del dispositivo y seleccionar el menú de Opciones de Desarrollador. Aquí habilitamos las opciones de ”Depuración de USB”(si no esta habilitada ya) y de .<sup>E</sup>sperar al depurador”. Además pulsamos donde pone ”Seleccione una aplicación para depurar” elegimos la aplicación que queremos depurar.

Una vez hecho esto, cada vez que ejecutemos la aplicación saldrá un mensaje de advertencia y se quedará esperando a que conectemos un depurador para continuar con su ejecución. Para esto, nos vamos a Eclipse y abrimos la vista de DDMS. Aquí nos aparecerá, entre otras cosas, un espacio con todos los procesos en ejecución en el dispositivo/emulador. Localizamos el proceso de nuestra aplicación y pulsamos sobre el bichillo verde para conectar el depurador a ella. Llegados a este punto la aplicación continua su ejecución en el emulador y aparece un escarabajo verde al lado del proceso de la app en la ventana de DDMS, que indica que se está depurando ese proceso. Es entonces cuando podemos abrir la vista de depuración de Eclipse y proceder a trabajar con breakpoints para depurar y estudiar el código con el fin de solucionar errores.

### 2.4.3. Migración: Identificación y Solución de Problemas

El objetivo de esta parte del proyecto es conseguir que el cliente de SwellRT se pueda desplegar en Android para así conseguir que se conecte al servidor WIAB que también incluye. Para ello lo primero que haremos será desplegar el servidor en nuestro ordenador clonando el repositorio de GitHub de SwellRT y siguiendo los pasos descritos en el Readme del proyecto [23]. Para comprobar que el servidor se ha instalado correctamente, podemos ejecutarlo por consola (ver Readme) y abrir un navegador web con la dirección <http://localhost:9898>. Si nos aparece una ventana de Login de WIAB es que ya tenemos un servidor WIAB corriendo en nuestro ordenador. Creamos entonces un usuario y contraseña de prueba. Este paso es importante ya que la aplicación Android intentará conectarse contra este servidor mientras estemos haciendo pruebas de desarrollo.

A continuación crearemos un proyecto Android en Eclipse e incluiremos en él todas las clases de SwellRT. Uno de los componentes principales de Android a la hora de desarrollar son las **Actividades** [42], que representan las pantallas que se le muestran al usuario y que responden a su interacción programáticamente. Por tanto, crearemos una nueva actividad principal (`waveAndroid.java`) que se ejecutará al lanzar la aplicación y que por el momento intentará conectarse al servidor especificando por código el usuario y contraseña que hemos creado antes en el servidor. Wave realiza este login contra el servidor usando dos tecnologías: HTTP [43] y WebSockets [44].

### Conexion HTTP

Wave fue desarrollado para utilizar el protocolo WebSocket para la conexión al servidor, pero esta tecnología necesita realizar una autenticación HTTP previa. Lo primero que haremos será otorgar **permisos de conexión a internet** a nuestra aplicación. Android utiliza un **sistema de permisos** [45] para controlar los privilegios de cada aplicación. Estos permisos se declaran en el **Manifiesto** de la aplicación [46], archivo que declara sus características. Para ello basta con añadir lo siguiente al manifest.xml de la aplicación:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Tambien hay que tener en cuenta que cuando nos encontramos en el emulador no estamos en la misma red que el ordenador en el que trabajamos, por lo que la conexión a la URL `http://localhost:9898` no es valida. No obstante, esto tiene facil solucion pues el **emulador de Android define unas direcciones IP de red especiales** [47] para este tipo de casos. Basta con sustituir localhost por la dirección `10.0.2.2` para conseguir acceder al servidor WIAB desplegado en el ordenador. La dirección URL sera por tanto: `http://10.0.2.2:9898`.

Lo siguiente que haremos será ejecutar el código de Login del cliente SwellRT para intentar localizar dónde se lleva a cabo la conexión HTTP. Para ello llamamos desde la actividad principal (WaveAndroid.java) al método `startSession()` de la clase WaveClient.java pasándole el usuario y la contraseña antes creados.

Esto provoca un error de ejecución y la aplicación se cierra. Lo siguiente que hacemos es depurar la aplicación (Ver Sección 2.4.2) estudiando el LogCat [48] (Ver Figura 2.5) para ver donde se produce el error. Descubrimos que el problema estaba localizado en el método `login()` de la misma clase, que intentaba realizar una **peticion POST HTTP** al servidor utilizando un **RequestBuilder** de la librería `com.google.gwt.http.client`. He aquí el primer problema: la actual conexión utiliza métodos de GWT/Javascript para hacer la petición post y Android no es compatible con esta tecnología.

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

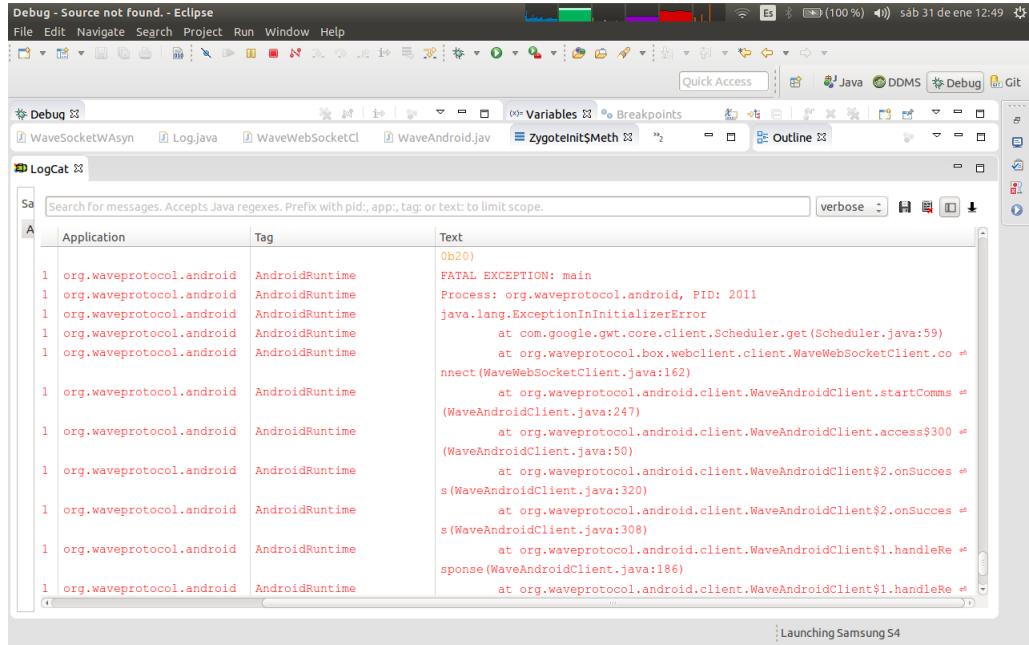


Figura 2.5: Ejemplo de Traza de Error en Logcat

Hay por tanto que encontrar una librería similar compatible con Android que construya una petición **HTTP POST** y la envíe al servidor. La primera opción que valoramos fue utilizar la **librería HTTP Apache** [49], incluida en el SDK de Android desde sus primeras versiones. Sin embargo, Google recomienda [50] a partir del API 10 (Android 2.3 "Gingerbread") utilizar la **librería HttpURLConnection** [51]. Por tanto esta última es la que elegimos para la migración.

De forma simplificada, este sería un esquema de la nueva estructura del login HTTP:

```
import java.net.HttpURLConnection;

private void login(final String user, final String password,
    final Callback<String, String> callback) {

    //Construct the URL String urlString with the
    //server, user and password parameters

    URL url = new URL(urlStr); //String
    HttpURLConnection connection = (HttpURLConnection
        ) url.openConnection(); //Open the connection
    to the given URL
```

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

```
connection.setDoOutput(true); // allow the POST connection
connection.setRequestProperty("Accept-Charset", CHARSET);
connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded; charset=" + CHARSET);

OutputStream out = connection.getOutputStream();
out.write(queryStr.getBytes(CHARSET)); //Set the POST parameters

if (connection.getResponseCode() != 200) {
    //ERROR during the connection
    connection.disconnect(); //Disconnect from the server.
} else {
    //Continue with the login process ( WebSocket)
    connection.disconnect(); //Disconnect from the server.
}
}
```

Sin embargo, aquí no acaba el problema. Por cuestiones de usabilidad y de respuesta a la interacción del usuario, Android establece dos reglas para trabajar con el proceso de la actividad que se le esta mostrando al usuario (llamado **UI Thread**) [52]:

- 1. No bloquear el UI Thread
  - 2. No acceder al UI Thread directamente desde otro Thread

La conexión a un servidor es un proceso susceptible de durar un tiempo variable según las condiciones de la red, lo cual deja la aplicación en espera hasta que se realiza dicha conexión, bloqueando el UI Thread. Por tanto, decidimos usar un hilo (Thread) por separado en forma de **AsyncTask** [?] para llevar a cabo la tarea de Login, tal y como recomienda Google hacer para trabajar con conexiones a la red [53]. La ventaja por tanto de usar otro hilo para esto es que la actividad principal no se bloquea.

El siguiente es un esquema del AsyncTask encargado del Login:

```

private class LoginTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... params) { // method that executes on the new Thread without
        blocking the UI Thread
        login(params[0], params[1], params[2]); //Do the login
    }

    @Override
    protected void onPostExecute(String result) { //method that executes on the UI Thread once doInBackground() finishes its execution.

        if (result != null) {
            callback.onLogin(); //Notify the login success using the proper callback method
        } else { //The doInBackground method has had a problem and the result of its execution was null
            callback.onError("Wave>Login>Error"); //Notify the login error using the proper callback method
        }
    }
}

```

Es importante tambien destacar que la arquitectura de SwellRT y de Wave esta planteada de manera que utiliza llamadas asíncronas (callbacks) para notificar al resto de la aplicacion del resultado de los procesos de conexión al servidor.

**Llegados a este punto, tenemos un proceso de login Http que hace uso de la libreria HttpURLConnection y de un AsyncTask para realizar esa primera conexión al servidor.** Depuramos la aplicación y comprobamos que efectivamente el login Http se realiza correctamente (la respuesta del servidor tiene código 200). **Sin embargo la aplicacion aun no funciona correctamente, pues se cierra al intentar ejecutar el código que se encarga del siguiente paso del login: la conexión por WebSocket.**

### Conexión WebSocket

Para realizar una conexión con el servidor Wave es necesaria una conexión mediante WebSockets[44], tecnología que permite conexiones bidireccionales

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

---

(recordemos que la conexión en el modelo cliente-servidor tradicional está definida como unidireccional de cliente a servidor) y asíncronas entre el servidor y el cliente, de manera que cualquiera de los dos puede iniciar una conexión con el otro en cualquier momento e intercambiar información con éste. En el caso del protocolo Wave este comportamiento es el deseable, ya que al tratarse de un protocolo de comunicaciones federado en el que cualquiera en la red puede ser cliente o servidor, es importante que la conexión sea bidireccional. Además la asíncronía es necesaria ya que para mantener la consistencia en tiempo real 2.3.1 hace falta que el servidor que contiene las waves pueda iniciar una conexión con los clientes para notificar los cambios que se produzcan en dichas waves. Por tanto, nuestro cliente Android debe ahora establecer una conexión WebSocket con el servidor WIAB.

La metodología a utilizar será la misma que para la conexión HTTP, se ejecutará el código de SwellRT para identificar donde falla y por tanto cómo está estructurada la creación y gestión de WebSockets en la versión GWT.

El cliente SwellRT original realiza esta conexión utilizando una librería llamada Atmosphere [54], que proporciona un framework para Java que permite gestionar conexiones WebSocket junto a la conexión HTTP que subyace por debajo. Sin embargo, esta librería se encarga solo de gestionar la conexión, no de crear el WebSocket propiamente dicho. En el caso de SwellRT este WebSocket se crea utilizando la implementación que proporciona GWT llamada también WebSocket (WebSocket.java). Esta clase nos define las funciones básicas que debería tener nuestro WebSocket: **onOpen()** para establecer la conexión, **onMessage()** para recibir mensajes por el WebSocket, **send()** para enviar mensajes y **onClose()** para cerrar la conexión. Así mismo nuestro Websocket deberá también implementar una serie de callbacks (definidos en la interfaz WebSocketCallback.java) para notificar a la aplicación de la llegada de estos eventos del servidor. Los callbacks son: **onConnect()**, **onDisconnect()** y **onMessage(message)** respectivamente.

Este WebSocket se crea utilizando un **patrón de diseño Factory**, que abstrae la creación de un objeto de su implementación, de manera que el desarrollador tenga acceso al objeto sin tener que preocuparse de cómo este implementado el WebSocket por debajo (ver WaveSocketFactory.java en SwellRT). En este caso, como ya se ha dicho con Atmosphere y WebSocket GWT. No obstante, la aplicación no funciona tal y como está hecho en SwellRT ya que android no soporta GWT de forma nativa. Hay que sustituir este código buscando una librería open-source que implemente un WebSocket en Android sobre Atmosphere y que porproporcione las mismas funciones básicas descritas en el párrafo anterior.

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

---

La solución encontrada fue importar al proyecto e utilizar wAsync[55], librería proporcionada por Atmosphere para trabajar con Websockets en Node.js, Java y Android. wAsync trabaja creando un socket que responde a eventos diversos, estando entre ellos eventos similares a los utilizados por la versión GWT de SwellRT: `on(EVENT.name())`, siendo EVENT el nombre del evento al que debe responder. Un ejemplo sencillo de utilización de wAsync sería el siguiente:

```
//Create the atmosphere client
AtmosphereClient client = ClientFactory.getDefault().
    newClient(AtmosphereClient.class);

//Configure client with URL
SphereRequestBuilder requestBuilder = client.
    newRequestBuilder()
    .method(Request.METHOD.GET).trackMessageLength(true).uri(
        WaveSocketWAsync.this.urlBase)
    .transport(Request.TRANSPORT.WEBSOCKET)

//Create and configure socket
SocketWAsync.this.socket = client.create(client.
    newOptionsBuilder().runtime(ahc).build())
.on(Event.OPEN.name(), new Function<String>() { //Equivalent
    to GWT onOpen() method
@Override
public void on(String arg0) {
    // set the actions to do and call the proper
    callback function (callback.onConnect())
}
}).on(Event.CLOSE.name(), new Function<String>() { //Equivalent
    to GWT onClose() method
@Override
public void on(String arg0) {
    // set the actions to do and call the proper callback
    function (callback.onDisconnect())
}
}).on(Event.MESSAGE.name(), new Function<String>() {
@Override
public void on(String arg) { //Equivalent to GWT onMessage
    () method
    // set the actions to do and call the proper callback
    function (callback.onMessage())
}
}).on(new Function<Throwable>() {
@Override
public void on(Throwable t) {
    // catch possible exceptions
}
```

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

```
});  
  
{  
    // connect to the server  
cket.open(requestBuilder.build());  
tch (IOException e) {  
    // catch possible exceptions  
  
nd a given message to the server, equivalent to GWT send(msg)  
method  
et.fire(Data);
```

Como la arquitectura Wave utilizaba el patrón factoria, fue necesario sustituir la clase de WebSocket GWT por una de nueva creación llamada **WaveSocketWAsync.java**[56] que implementa los métodos de creación y configuración de un Websocket y de callback antes descritos. Asimismo se modificó la clase `WaveSocketFactory.java` para hacer uso ahora de esta nueva implementación del WebSocket compatible con Android.

Sin embargo, ejecutamos esta nueva versión de código y nos encontramos con que la librería WAync incluye dependencias a código que no está presente en la propia librería y que es necesario para crear el cliente `AsyncHTTPClient` que gestiona la conexión HTTP que subyace por debajo del WebSocket. Concretamente hace falta utilizar un `HTTPProvider` compatible con Atmosphere, tal y como recomienda hacer wAsync en su wiki???. Para ello basta con añadir al proyecto las siguientes librerías y configurar el cliente segun lo descrito en dicha wiki:

- **AsyncHttpClient 1.8.14**[57]
- **Grizzly 2.3.18**[58]

```
AsyncHttpClientConfig ahcConfig = new AsyncHttpClientConfig.  
    Builder().build();  
cHttpClient ahc = new AsyncHttpClient(new  
    GrizzlyAsyncHttpProvider(ahcConfig));
```

Ahora, al ejecutar la aplicación comprobamos que la conexión al servidor se produce correctamente. Para completar esta parte del proyecto solo falta pedir al usuario su user y su password, ya que hasta ahora las habíamos especificado nosotros en el propio código para realizar pruebas.

### Login y Logging

Para probar que la aplicación migrada es funcional y es capaz de utilizar las características nativas de Android haremos una pequeña y sencilla pantalla de Login que pedirá al usuario la dirección del servidor Wave, un usuario y una contraseña.

En el diseño de aplicaciones Android el componente principal de una app es la Actividad, que se corresponde con la pantalla con la que interactúa el usuario. La interfaz gráfica se usuario (UI) de las pantallas se encuentra separada del código de la aplicación en ficheros xml de Layout[?]. A una Actividad se le especifica cuál es su archivo de layout en su método onCreate(), responsable de la creación de la actividad y sus recursos.

Creamos una Actividad llamada WaveAndroid.java que haga uso del layout Main.xml, en el cual incluimos tres cajas de texto (llamadas EditText en Android) para que el usuario introduzca los datos. Además guardamos una referencia a estas cajas de texto en la Actividad para poder acceder al texto introducido y pasárselo al método login de Wave que hemos migrado anteriormente.

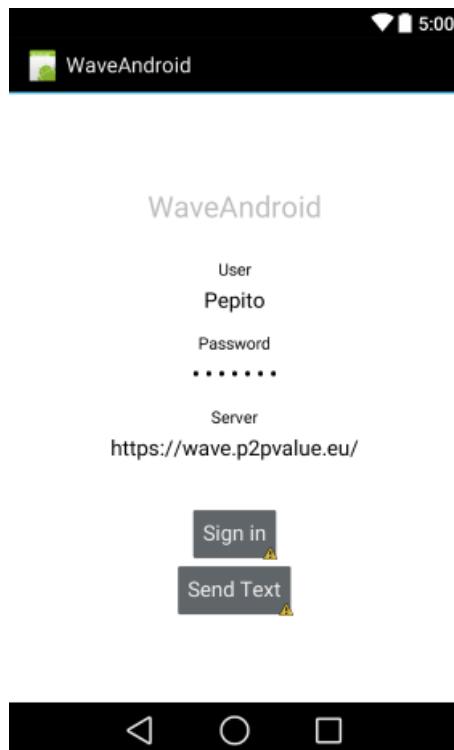


Figura 2.6: Pantalla de Login de WaveAndroid

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

---

Además decidimos mejorar el sistema de mensajes de Log de la aplicación sustituyendo el framework de la librería SLF4J para Java usada por SwellRT por una versión más reciente desarrollada para Android[59].

Por último instalamos la aplicación en el emulador o el dispositivo móvil y probamos que se nos muestra la pantalla de login anterior. Introducimos los datos del servidor WIAB (en este caso utilizaremos el servidor de P2PValue desplegado para pruebas en <https://wave.p2pvalue.eu/>), de usuario y contraseña; y comprobamos que hemos conseguido el objetivo de esta parte del proyecto: **el login del cliente Android contra el servidor WIAB se realiza de forma correcta y sin fallos.**

### 2.4.4. Organización y Resultados

Una vez conseguida la conexión al servidor desde Android, decidimos revisar el código para intentar optimizarlo y organizarlo de manera que aproveche mejor las características de Android. Además el API que permite gestionar el modelo de datos de SwellRT (Ver Sección 2.3.2 está escrito en java, por lo que es plenamente funcional y compatible con el código de nuestra migración a Android. En los siguientes puntos hablaremos de dicha reorganización de código y del resultado final de esta parte del proyecto.

#### Servicio Android

##### NO TERMINADO

Como ya se ha comentado anteriormente, el proceso de login se debe hacer en un hilo de ejecución separado del hilo principal o de UX, ya que Android no recomienda[52] que tareas que tarden mucho tiempo en ejecutarse (como por ejemplo descarga de datos de la red) se ejecuten en el mismo hilo que la interfaz de usuario, pudiendo bloquear dicho hilo y obstaculizando por tanto la interacción del usuario con el dispositivo.

Hasta ahora habíamos utilizado para ello una Actividad que contenía el AsyncTask[?] encargado de ejecutar el código de conexión al servidor en un hilo separado del UI Thread. Sin embargo, tal y como está definida la arquitectura de Wave y de SwellRT, la utilización de callbacks (ver secciones 2.4.3 y 2.4.3) es necesaria para notificar al resto de la aplicación de los eventos relacionados con el intercambio de datos con el servidor. Un AsyncTask ejecuta de una sola vez y de forma asíncrona el código que se le asigne a su método doInBackground(), de manera que en nuestro caso no podríamos utilizar los

callbacks PORQUE...

Además cada vez que quisiéramos realizar algún tipo de interacción con el servidor habría que utilizar un AsyncTask específico para ello, lo cual no hace sino añadir más código a la aplicación. Otra desventaja de los AsyncTask en la implementación actual es que si la Actividad que lo esta ejecutando pasa a segundo plano (por ejemplo si el usuario cambia de aplicación) se detiene el proceso de conexión.

Considerando todo esto decidimos utilizar otro componente de Android pensado para ejecutar tareas en background y con la opción de hacerlo de forma independiente de la aplicación: el Servicio[60].

Un Servicio Android se diferencia de una Actividad en que es un componente que se ejecuta en segundo plano y no proporciona una interfaz de usuario con la que éste pueda interactuar. Un Servicio ejecuta tareas de larga duración (como la descarga de datos de la red) a petición de otros componentes de la aplicación que estén "suscritos" (el término utilizado por android es "bind") a dicho Servicio, a modo de cliente-servidor. De esta manera la Actividad (cliente) que se suscriba al Servicio (servidor) puede interactuar con éste último haciendo peticiones y recibiendo notificaciones cuando el Servicio obtenga resultados. Pero un Servicio se ejecuta dentro del mismo proceso que el UI Thread, por lo que aun así tendremos que utilizar

WebSocket usa AsyncTask? HANDLER -¿MENSAJES CIERRE DE APLICACIÓN

### Resultado de la Migración

Después de todos estos pasos disponíamos de una versión funcional de SwellRT capaz de conectarse al servidor WIAB de forma nativa desde Android. **El resultado de esto se puede ver en el GitHub de esta parte del proyecto[56].**

Solo restaba poner el API de SwellRT encima de lo nuestro para poder acceder y trabajar a nivel de wave con el modelo de datos de SwellRT. De esto se encargó uno de los desarrolladores del proyecto inicial, Pablo Ojanguren, con el cual habíamos trabajado para realizar lo anteriormente descrito.

El API con el cliente de SwellRT adaptado a Android, con el cual trabajaremos en la siguiente parte del proyecto para crear una aplicación Android que haga uso de ella, se encuentra en el GitHub de SwellRT[61].

---

## CAPÍTULO 2. MIGRACIÓN DE WAVE A ANDROID

---

### Diagramas y Dependencias

El cliente Android de SwellRT migrado y desarrollado en esta parte del proyecto hace uso de las siguientes dependencias con librerías externas a Android:

TABLA

| Nombre               | Versión | Descripción                                                                                                          |
|----------------------|---------|----------------------------------------------------------------------------------------------------------------------|
| WAsync[55]           | 1.4.3   | WebSockets/HTTP Client Library for Asynchronous Communication                                                        |
| AsyncHttpClient [57] | 1.8.14  | Library that allows Java applications to easily execute HTTP requests and asynchronously process the HTTP responses. |
| Grizzly [58]         | 2.3.18  |                                                                                                                      |

Cuadro 2.1: Technologies Summary



# Capítulo 3

## Construyendo la idea: DemoCritics

### 3.1. Introducción

Una vez que habíamos elegido la tecnología que soportaría el núcleo de nuestra aplicación, teníamos que decir que implementación le íbamos a dar a nuestra aplicación móvil. Para ello teníamos que tener en cuenta las características que nos ofrecía Wave:

Edición colaborativa.

Tiempo real.

Consistencia.

Después de darle unas cuantas vueltas de las posibles implementaciones que podríamos realizar sobre estas características potenciales, decidimos realizar una sesión de brain storming. En esta sesión aparecieron temas tan dispersos como wikis colaborativas, aplicaciones con inteligencia artificial, aportaciones colaborativas en política, edición de vídeos y música, cursos de formación colaborativos, etcétera.

## CAPÍTULO 3. CONSTRUYENDO LA IDEA: DEMOCRITICS

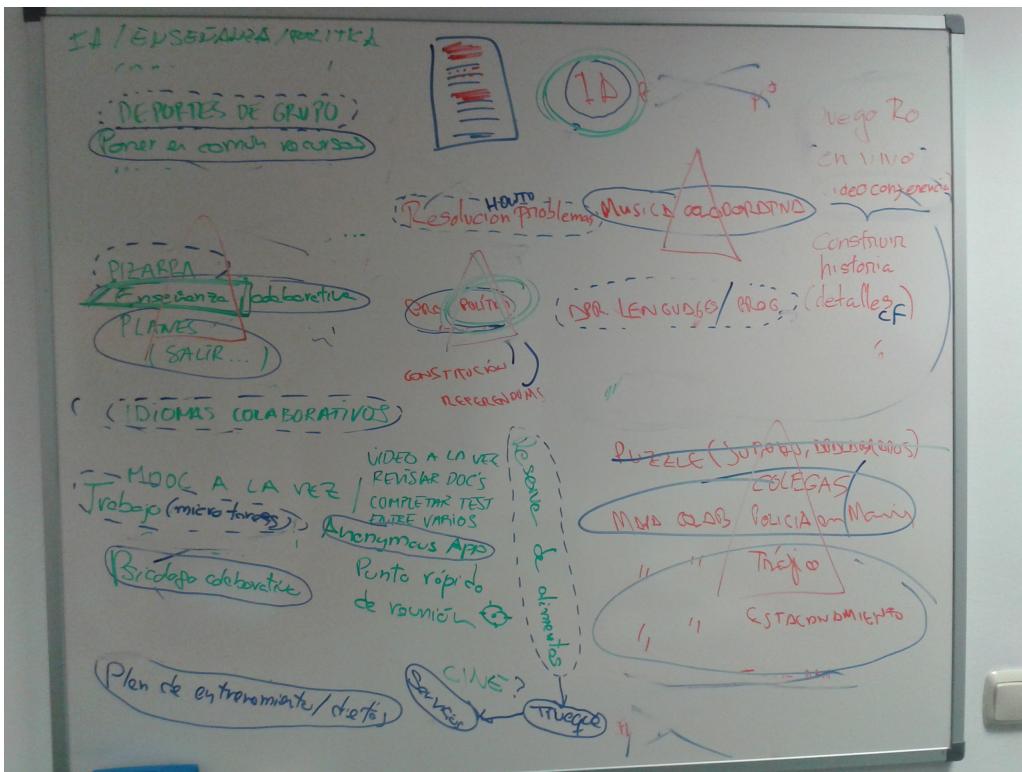


Figura 3.1: Brainstorming sobre la idea a desarrollar

Con un gran repertorio de ideas expuestas sobre la sesión, descartamos aquellas que no nos motivaban llevarlas a cabo. Por lo que nos quedamos con tres ideas fundamentales a desarrollar en nuestra aplicación: Política, Música, Inteligencia Artificial y Mapas. Surgieron varias ideas colaborativas como desarrollar documentos políticos, programas electorales, comunicación entre colectivos en tiempo real, aprendizaje de música, edición de partituras y obras, aplicaciones colaborativas con inteligencia artificial, edición de mapas en tiempo real, lexicalización, etcétera.

Finalmente debido a intereses comunes, decimos realizar una aplicación colaborativa relacionado con el mundo de la política. Con el objetivo de que pudiera tener cierta repercusión y utilidad en las próximas citas electorales durante el año 2015. En esta aplicación podríamos recurrir a la edición de contenidos en tiempo real, ya fueran propuestas políticas, programas electorales y otro tipo de documentos. Como también hacer uso de alguna herramienta de Inteligencia Artificial para automatizar algunas tareas o realizar recomendaciones sociales.

### **3.1.1. Política en el mundo de la Informática**

A primera vista podemos pensar que la informática no parece entusiasmar a los informáticos. Pues podemos ubicar la política como una parte de las ciencias sociales, situando la informática en ciencias formales. Pero si factores como la gestión de los privilegios de una aplicación entre los que definiremos de alguna forma una jerarquía, estaremos haciendo política de alguna manera. También encontraremos características políticas en el diseño relacional de una base de datos. Definiendo los campos de una base de datos podemos encontraremos con algunos valores como el sexo, la nacionalidad, la edad o incluso las relaciones o restricciones que existen entre las tablas. Estaremos definiendo unas reglas básicas funcionamiento de la base de datos establecidas por unos principios políticos.

Además sumergiéndonos en el mundo de las licencias de uso en el desarrollo de software encontraremos más política aún. Licencias que determinan el uso de un tipo de software, ya sea para compartir, vender, distribuir copias, entre otros. Multitud de reglas políticas definidas en una licencia de uso. Así como las restricciones que establecemos en la metodología orientada a objetos, estableciendo las relaciones de herencia, restricción de métodos, variables, etcétera.

Regresando a la actualidad y basándonos en no muy lejanos acontecimientos pasados, habremos oído cómo algunos gobiernos recopilan datos de la actividad de los usuarios en las redes sociales, analizando todo el contenido que generan. Incluso vemos cómo algunas aplicaciones móviles piden aprobar permisos con los que operar libremente en tu dispositivo.

Como podemos observar la política está más integrada en la informática de lo que parece, dejando a un lado la informática más científica, más formal, pasando a la informática social, la de los gobiernos, la de los negocios o la de las relaciones sociales.

## CAPÍTULO 3. CONSTRUYENDO LA IDEA: DEMOCRITICS

---

### 3.1.2. Democracia

**Democracia representativa**

**Democracia participativa**

**Democracia directa**

**Democracia deliberativa**

### 3.1.3. Adentrándonos en la idea

La idea a desarrollar generada en una época dónde la política parecía haber despertado el interés de una parte considerable de la ciudadanía, podría ser una herramienta útil para participar en temas políticos que forma sencilla y atractiva. Dejando atrás los tópicos yo no entiendo de política, la política es aburrida, no sé a quién votar o no he leído nunca un programa electoral entre otros.

La herramienta ofrecería una nueva forma de participar en la política y de llevar a los ciudadanos los programas electorales ofertados por las diferentes formaciones políticas. De tal forma que los ciudadanos pudieran leer aquellos puntos de los programas más leídos, debatidos, comentados, etcétera. Así cualquier usuario tendría todos programas electorales en su bolsillo, por lo que no tendría que ir a la página web de cada formación política y descargar un documento de 200 páginas. Pensamos que esta forma de presentar un programa político en un mundo donde las posibilidades de comunicarnos se han desarrollado exponencialmente, no era la mejor manera de llegar a la mayor parte de la ciudadanía.

Por otra parte, la aplicación también debería ofrecer alguna herramienta donde realizar propuestas y debatirlas entre todos. De tal forma que tanto la ciudadanía como las formaciones políticas pudieran saber en cualquier momento cuáles son las principales preocupaciones de los ciudadanos y qué medidas o soluciones proponen para resolverlas.

Desde un primer punto de vista subjetivo, la aplicación quedó dividida en dos partes. Por un lado tendríamos los programas políticos que presentaran las formaciones políticas. Y por otro, todas las propuestas que elaboraran los ciudadanos individualmente o en colectivos sociales.

## 3.2. Estado del Arte

En esta sección exploraremos las principales aplicaciones informáticas destinadas a la participación ciudadana en propuestas, lectura de programas electorales o divulgación de candidaturas.

### 3.2.1. Programas Políticos

En la actualidad no existe ningún tipo de aplicación orientada a debatir los programas electorales de los partidos políticos. Concretamente no hay ningún tipo de plataforma que agrupe los programas electorales de las diferentes candidaturas. Lo más parecido que hemos podido encontrar han sido aplicaciones elaboradas por un partido político, orientada a dar a conocer su candidatura. En ella podremos ver la candidatura, vídeos y el programa electoral entre otros. Por ello pasamos a analizar las aplicaciones encontradas:

#### UPyD Parla

La aplicación que presenta el candidato de UpyD Carlos Alt Bustelo para la alcaldía de Parla. Se trata de una aplicación divulgativa donde podemos conocer todo lo esencial de la candidatura de UpyD para las elecciones del municipio de Parla en Mayo de 2015. Los candidatos, el programa, vídeos, etcétera.

## CAPÍTULO 3. CONSTRUYENDO LA IDEA: DEMOCRITICS

---



Figura 3.2: UPyD Parla

### #RecuperaCórdoba

De forma similar a la anterior, la candidatura de Pedro García a la provincia de Córdoba de Izquierda Unida, presenta su propuesta de gobierno de forma compacta. En la aplicación podremos encontrar la lista de los candidatos propuestos a la comunidad cordobesa, el programa electoral de la formación, las propuestas del partido, noticias de última hora y vídeos.

## CAPÍTULO 3. CONSTRUYENDO LA IDEA: DEMOCRITICS

---



Figura 3.3: #RecuperaCórdoba

### PP Canarias

La delegación del Partido Popular en Canarias, presenta su aplicación móvil para promocionar a sus candidatos para las elecciones autonómicas y municipales de Mayo de 2015. La aplicación nos avisará de los eventos electorales, podremos consultar los candidatos, novedades, galería de imágenes y por su puesto el programa electoral.

## CAPÍTULO 3. CONSTRUYENDO LA IDEA: DEMOCRITICS



Figura 3.4: PP Canarias

### 3.2.2. Participación Ciudadana

Centrándonos en la participación ciudadana ya sea generación de propuestas, desarrollo colaborativo de programas o recogida de firmas, existen numerosos portales en internet y aplicaciones móviles destinadas a ello. Realizaremos un breve repaso a las aplicaciones más destacadas.

#### reddit

Reddit [62] es un sitio web donde los usuarios pueden crear temas, propuestas o compartir enlaces web a otros sitios. A primera vista puede parecer un foro, la principal diferencia respecto a este último radica en que los usuarios pueden votar a favor o en contra de los enlaces, haciendo que aparezcan más o menos destacados. De tal forma que los temas de conversación, enlaces, o propuestas aparecerán en el orden que haya escogido la comunidad según la

## CAPÍTULO 3. CONSTRUYENDO LA IDEA: DEMOCRITICS

puntuación positiva o negativa. En principio el uso de reddit está destinado a todos los temas, donde podemos encontrar algunos ejemplos relacionados fuertemente con la participación ciudadana en Plaza Podemos [63]. Un espacio utilizado para que la ciudadanía pueda expresar sus propuestas, compartir noticias relacionadas con la actualidad política o debatir aquellos temas que más preocupan. Así de un simple vistazo podemos saber qué es lo más debatido entre la ciudadanía, las propuestas que quieren llevar a cabo en el gobierno o cuáles son los temas que más preocupan.



Figura 3.5: Plaza Podemos utilizando reddit

## Change.org

Change.org [?] es portal web que permite alojar múltiples peticiones por internet. Podríamos definirlo como la evolución de la recogida de firmas en la calle. Cualquiera puede realizar una petición para solicitar apoyos. Las perso-

## CAPÍTULO 3. CONSTRUYENDO LA IDEA: DEMOCRITICS

nas que decidan apoyar la petición, dejarán sus datos personales y constarán entre un número de personas que han apoyado la petición. Una vez que han alcanzado un número se procede a entregar las firmas digitales al organismo, persona o entidad a la que va destinada la petición.

En mayo de 2011, en relación con las movilizaciones del Movimiento 15-M y Democracia Real Ya, ante el desalojo por los Mozos de Escuadra se llevó a cabo la petición “Exige la dimisión fulminante del Conseller de Interior Felip Puig por la violencia utilizada en Pza. Catalunya”.

Desde su creación en 2007, change.org ha logrado muchas de sus peticiones demandadas entre los que se incluyen la atención de pacientes con enfermedades complejas, protección sobre animales y medio ambiente, derechos públicos, leyes, etcétera.



Figura 3.6: Change.org · La mayor plataforma de peticiones del mundo

### Programa Colaborativo AhoraMadrid

Para las pasadas elecciones municipales del 24 de Mayo, la candidatura de unidad popular Ahora Madrid, desarrolló una plataforma en la web para

## CAPÍTULO 3. CONSTRUYENDO LA IDEA: DEMOCRITICS

elaborar su programa electoral de forma colaborativa. En esta plataforma, cualquier usuario tenía la oportunidad de explorar las propuestas por categoría o por distrito. De tal forma que podría debatirlas, puntuarlas o crear sus propias propuestas. Así las propuestas más valoradas por la comunidad, serían llevadas al programa final para las elecciones municipales del 24 de Mayo.



Figura 3.7: Creación colaborativa del programa de Ahora Madrid.



# **Capítulo 4**

## **Resultados y Conclusiones**

### **4.1. Discusion de Resultados**

#### **4.1.1. Evaluación con usuarios**

### **4.2. Conclusiones**



# Capítulo 5

## Trabajo a Futuro

### 5.1. Mejoras



# Bibliografía

- [1] Inc. Google. Meet Google Wave.  
<http://googlecode.blogspot.com.es/2009/05/hello-world-meet-google-wave.html>.
- [2] Inc. Google. Google Wave Federation Protocol Over XMPP.  
<http://wave-protocol.googlecode.com/hg/spec/federation/wavespec.html>.
- [3] Inc. Google. End of Google Wave.  
<https://support.google.com/answer/1083134?hl=en>.
- [4] Google Docs.  
<https://drive.google.com/>.
- [5] Google+.  
<https://plus.google.com/>.
- [6] Apache. Apache Wave (Incubating).  
<http://incubator.apache.org/wave/about.html>.
- [7] Apache. Apache License 2.0.  
<http://www.apache.org/licenses/LICENSE-2.0>.
- [8] Inc. Google. Wave Federation.  
<http://www.waveprotocol.org/federation>.
- [9] Google. Google Wave Federation Architecture White Paper.  
<http://wave-protocol.googlecode.com/hg/whitepapers/google-wave-architecture/google-wave-architecture.html>.
- [10] Peter Saint-Andre. Extensible messaging and presence protocol (xmpp): Core. 2011.  
RFC 6120 Available at <http://tools.ietf.org/html/rfc6120>.
- [11] Understanding Operational Transformation.  
<http://www.codecommit.com/blog/java/understanding-and-applying-operational-transformation>.

- [12] Inc. Google. Google Wave Operational Transformation.  
<http://www.waveprotocol.org/whitepapers/operational-transform>.
- [13] Inc. Google. Google Wave Conversation Model.  
<https://wave-protocol.googlecode.com/hg/spec/conversation/convspec.html>.
- [14] Inc. Google. Google Wave API Overview.  
<http://www.waveprotocol.org/wave-apis>.
- [15] Inc. Google. Google Wave Extensions.  
<http://www.waveprotocol.org/wave-apis/extensions>.
- [16] Apache. Wave In A Box.  
<http://www.waveprotocol.org/wave-in-a-box/>.
- [17] Oracle. OpenJDK.  
<http://openjdk.java.net/>.
- [18] Inc. Google. Google Web Toolkit.  
<http://www.gwtproject.org/>.
- [19] Apache. Install WIAB.  
<https://cwiki.apache.org/confluence/display/WAVE/Install+WIAB>.
- [20] Apache. WIAB Repository.  
<https://github.com/apache/incubator-wave>.
- [21] WIAB Server Example.  
<http://waveinabox.net/>.
- [22] P2PValue. P2P Value European Project.  
<http://www.p2pvalue.eu/>.
- [23] P2PValue. SwellRT, a real-time federated collaboration framework.  
<https://github.com/P2Pvalue/swellrt>.
- [24] Inc. Google. Android Developers Site.  
<http://developer.android.com/index.html>.
- [25] Inc. Google. Android SDK.  
<https://developer.android.com/sdk/index.html>.

- [26] Eclipse Foundation. Eclipse IDE.  
<https://eclipse.org/ide/>.
- [27] Inc. Google. Android Studio IDE.  
<https://developer.android.com/tools/studio/index.html>.
- [28] JetBrains. IntelliJ IDEA IDE.  
<https://www.jetbrains.com/idea/>.
- [29] Google Inc. Eclipse Android Development Tools (ADT).  
<https://developer.android.com/tools/help/adt.html>.
- [30] Google Inc. Android API Reference.  
<http://developer.android.com/reference/packages.html>.
- [31] Google Inc. Android Virtual Device Manager.  
<http://developer.android.com/tools/devices/managing-avds.html>.
- [32] Google Inc. Using the Android Emulator.  
<http://developer.android.com/tools/devices/emulator.html>.
- [33] Google Inc. Android SDK Manager.  
<https://developer.android.com/tools/help/sdk-manager.html>.
- [34] Google Inc. Android Dalvik Debug Monitor Server.  
<http://developer.android.com/tools/debugging/ddms.html>.
- [35] Google Inc. Android Versions Distribution.  
<https://developer.android.com/about/dashboards/index.html>.
- [36] Google Inc. Building Apps with Over 65K Methods.  
<https://developer.android.com/tools/building/multidex.html>.
- [37] Google Inc. ProGuard.  
<http://developer.android.com/tools/help/proguard.html>.
- [38] Apache Foundation. Apache Ant.  
<http://ant.apache.org/>.
- [39] Google Inc. Building and Running from the Command Line.  
<https://developer.android.com/tools/building/building-cmdline-ant.html>.
- [40] Google Inc. Setting up a Device for Development.  
<https://developer.android.com/tools/device.html#setting-up>.

- [41] Google Inc. Android Debug Bridge.  
<http://developer.android.com/tools/help/adb.html>.
- [42] Google Inc. Activities.  
<https://developer.android.com/guide/components/activities.html>.
- [43] Roy Fielding. Hypertext transfer protocol (http/1.1): Authentication. 2014.  
RFC 7235 Available at <https://tools.ietf.org/html/rfc7235>.
- [44] Ian Fette and Alexey Melnikov. Hypertext transfer protocol (http/1.1): Authentication. 2011.  
RFC 6455 Available at <https://tools.ietf.org/html/rfc6455>.
- [45] Google Inc. System Permissions.  
<https://developer.android.com/guide/topics/security/permissions.html>.
- [46] Google Inc. App Manifest.  
<https://developer.android.com/guide/topics/manifest/manifest-intro.html>.
- [47] Google Inc. Emulator Networking: Network Address Space.  
<https://developer.android.com/tools/devices/emulator.html#emulatornetworking>.
- [48] Google Inc. Android LogCat.  
<https://developer.android.com/tools/help/logcat.html>.
- [49] Google Inc. Apache HTTP Library.  
<https://developer.android.com/reference/org/apache/http/package-summary.html>.
- [50] Google Inc. Android's HTTP Clients.  
<http://android-developers.blogspot.com.es/2011/09/androids-http-clients.html>.
- [51] Google Inc. HttpURLConnection Library.  
<https://developer.android.com/reference/java/net/HttpURLConnection.html>.
- [52] Google Inc. Processes and Threads.  
<https://developer.android.com/guide/components/processes-and-threads.html>.

- [53] Google Inc. Connecting to the Network.  
<https://developer.android.com/training/basics/network-ops/connecting.html>.
- [54] Async-IO.org. Atmosphere: The Asynchronous WebSocket/Comet Framework.  
<https://github.com/Atmosphere/atmosphere>.
- [55] Async-IO.org. wAsync: A WebSockets/HTTP Client Library for Asynchronous Communication.  
<https://github.com/Atmosphere/wasync>.
- [56] Wave Client for Android.  
<https://github.com/Zorbel/swell-android>.
- [57] Async-IO.org. Async Http Client.  
<https://github.com/AsyncHttpClient/async-http-client>.
- [58] Project Grizzly. Project Grizzly: NIO Event Development Simplified.  
<https://grizzly.java.net/index.html>.
- [59] QOS.ch. SLF4J for Android.  
<http://www.slf4j.org/android/>.
- [60] Google Inc. Android Services.  
<https://developer.android.com/guide/components/services.html>.
- [61] P2PValue. SwellRT Client for Android.  
<https://github.com/P2Pvalue/swellrt/tree/master/android>.
- [62] reddit: the front page of the internet.  
<https://www.reddit.com/>.
- [63] Plaza Podemos: ¡Sí se puede!  
<https://www.reddit.com/r/podemos/>.