

DemoCritics

Aplicación Android de participación política con edición colaborativa en tiempo real

**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
Autorización de Difusión y Uso	III
Abstract	IX
Resumen	XI
1. Introducción	1
1.1. Objetivos del Proyecto	1
1.2. Estructura del Documento	1
2. Construyendo la idea de la aplicación: DemoCritics	3
2.1. Introducción	3
Brainstorming de ideas	3
2.2. Marco teórico de la idea	5
2.2.1. Política en el mundo de la Informática	5
2.2.2. Democracia	6
Democracia representativa	6
Democracia participativa	6
Democracia directa	6
Democracia deliberativa	6
2.3. Adentrándonos en la idea	6
3. Estado del Arte	9
3.1. Wave	9
3.1.1. Apache Wave: Wave In A Box (WIAB)	9
3.2. Edición Colaborativa en Tiempo Real	10

3.2.1.	APIs Centralizadas	10
	Google Realtime API	11
	Microsoft RTC Client API	11
	WebRTC	11
	Mozilla TogetherJS	11
	ShareJS	12
	Goodow	12
3.2.2.	Plataformas Web y Android	12
	Google Docs	12
	Etherpad	13
	Colorillo	14
	ShareLaTeX	15
	Samepage	16
	Quip	17
3.3.	Aplicación Android: DemoCritics	18
3.3.1.	Programas Políticos	18
	UPyD Parla	19
	#RecuperaCórdoba	20
	PSOE Andalucía	21
	PP Canarias	22
3.3.2.	Participación Ciudadana	23
	Reddit	23
	Change.org	24
	Programas Colaborativos de Ahora Madrid y Zaragoza en Común	26
	Appgree	28
4.	Tecnologías del Proyecto	31
4.1.	Tecnologías de Wave	33
4.1.1.	Google Wave	33

4.1.2.	Apache Wave	33
4.1.3.	Características de Wave	33
Federación	33	
Consistencia en tiempo real	34	
Escalabilidad	34	
4.1.4.	Servidores Wave	34
Wave in a Box	34	
SwellRT	35	
4.1.5.	Eclipse	35
4.1.6.	Java	36
4.1.7.	GWT	36
4.1.8.	JavaScript	37
4.1.9.	HTTP	37
4.1.10.	WebSocket	37
4.1.11.	Git	38
4.1.12.	GitHub	38
4.2.	Tecnologías de la Aplicación Android	39
4.2.1.	Android Studio	39
4.2.2.	Android	39
4.2.3.	XML	40
4.2.4.	JSON	40
4.2.5.	SQL	40
4.2.6.	MySQL	41
4.2.7.	PhpMyAdmin	41
4.2.8.	PHP	41
4.2.9.	Laravel 5	42
4.2.10.	OpenShift	42
4.2.11.	Sublime Text	42
4.2.12.	POP: Prototyping On Paper	43

5. Metodología del Proyecto	45
5.1. Uso de Software Libre	45
5.2. Metodología de Migración de Wave a Android	46
5.2.1. Objetivo	46
5.2.2. Plataforma: Entorno de Desarrollo, Construcción y Depuración	46
Eclipse	46
Proceso de Construcción por Consola	49
Proceso de Depuración	51
5.2.3. Migración: Identificación y Solución de Problemas	52
Conexión HTTP	53
Conexión WebSocket	57
Conexión final y Logging	59
Organización del código: Servicio Android	61
5.2.4. Dependencias	64
5.2.5. Resultado de la Migración	65
5.3. Diseño de la Aplicación: Diseño Guiado Por Objetivos	67
5.3.1. Investigación	68
Intención Inicial: Prototipo básico	68
Hipótesis de personas	69
Entrevista con Labodemo	70
Conclusión	72
Entrevista con Javier de la Cueva	73
Conclusión	74
5.3.2. Modelado de Personas	74
5.3.3. Definición de personas	76
5.3.4. Definición de Escenarios y Requisitos	76
5.3.5. Framework de diseño	79
Framework de interacción	79
5.3.6. Principios de Diseño	85

5.4.	Implementación de DemoCritics	88
5.4.1.	Metodología	88
Control de versiones	88	
GitHub	88	
Reparto de tareas	88	
Revisiones de código	88	
Evaluaciones de usabilidad	89	
5.5.	Evaluación con Usuarios	89
6.	Arquitectura del Proyecto	91
6.1.	Arquitectura de Wave	91
6.1.1.	Modelo Conversacional Wave	92
6.1.2.	Modelo de Contenidos SwellRT	93
6.2.	Arquitectura de la aplicación	94
6.2.1.	Base de datos	95
6.2.2.	Service REST	96
Arquitectura	99	
Rutas	101	
Controladores	103	
6.2.3.	Servicio Android de Conexión con Wave	105
6.2.4.	Cliente Android	105
Interfaz Gráfica	106	
Estructuración de Programas Políticos	111	
Conexión y Peticiones a Service REST y Base de Datos	114	
Gestión de Usuarios	115	
Peticiones a Wave	115	
7.	Resultados y Conclusiones	117
7.1.	Discusion de Resultados	117
7.1.1.	Evaluación con usuarios	117

7.2. Conclusiones	117
8. Trabajo a Futuro	119
8.1. Mejoras	119
Bibliografía	119

Índice de figuras

2.1.	Brainstorming sobre la idea a desarrollar	4
3.1.	Cliente Wave In A Box	10
3.2.	Capturas de Google Docs	13
3.3.	Captura de TitanPad: Ejemplo de uso de EtherPad	14
3.4.	Captura de Colorillo	15
3.5.	Captura de ShareLaTeX	16
3.6.	Capturas de Samepage	17
3.7.	Capturas de Quip	18
3.8.	Capturas de UPyD Parla	19
3.9.	Capturas de #IURecuperaCórdoba	20
3.10.	Capturas de PSOE Andalucía	21
3.11.	Capturas de PP Canarias	22
3.12.	Plaza Podemos utilizando la plataforma Reddit	24
3.13.	Change.org · La mayor plataforma de peticiones del mundo . .	25
3.14.	Creación colaborativa del programa de Ahora Madrid.	27
3.15.	Creación colaborativa del programa de Zaragoza en Común. .	27
3.16.	Capturas de Appgree	29
4.1.	Nube de Tecnologías utilizadas en el Proyecto	31
5.1.	Distribución Actual de Versiones Android (Fuente: Google) . .	48
5.2.	Emulador Android API 19	50
5.3.	Ejemplo de Traza de Error en Logcat	54
5.4.	Pantalla de Login de WaveAndroid	60
5.5.	Proceso de conexión Http con Servicio	63

5.6.	Proceso de conexión WebSocket con Servicio	64
5.7.	Esquema de Clases de SwellRT-Android con Servicio	66
5.8.	Estudiante universitario	76
5.9.	Activista político	77
5.10.	Modelo de desarrollo de los prototipos.	82
5.11.	Primeros prototipos sobre programas electorales.	83
5.12.	Prototipos para el desarrollo de propuestas.	84
5.13.	Principios de diseño en la aplicación.	87
6.1.	Arquitectura de Wave	92
6.2.	Modelo Conversacional de Wave	93
6.3.	Cambio de Modelo de Wave	94
6.4.	Arquitectura de la aplicación.	95
6.5.	Modelo entidad-relación de la base de datos.	96
6.6.	Frameworks PHP más populares. Fuente: SitePoint	97
6.7.	Arquitectura de Laravel	100
6.8.	Camino desde la ruta al método del controlador.	104
6.9.	Esquema de Implementación de TopItem (Con vista de Section y Proposal)	108
6.10.	Esquema de Implementación de Menú Izquierdo (Con vista del Menú)	110
6.11.	Ejemplos de Codificación de Secciones y Subsecciones	112
6.12.	Esquema de estructura en árbol de Programa Político	114
6.13.	Esquema de Clases de Conexión HTTP mediante AsyncTask .	115

Índice de cuadros

4.1.	Tecnologías usadas en el Proyecto	32
5.1.	Software libre utilizado durante el desarrollo.	45

5.2. Dependencias de SwellRT-Android	65
6.1. Funciones CRUD	98
6.2. Códigos de estado de la respuesta del servidor	99
6.3. Adapters, Views y Clases utilizadas en el proyecto	111

Abstract

Abstract en inglés.

Keywords:

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

Resumen

Actualmente vivimos en una sociedad de transición democrática en la que las personas comienzan a adquirir un papel determinante en el terreno político. Son cada vez más los movimientos sociales que proliferan a raíz del interés generado por participar en política. Esta participación se ha visto canalizada por la aparición de nuevas tecnologías que permiten establecer nuevas formas para que las personas se organicen y expresen su opinión. La gran mayoría de estas nuevas herramientas se desarrollan en un ámbito cercano a plataformas interconectadas en red y basadas en una interfaz web. Encontrándonos en un momento sujeto a varias citas electorales que propicia el aumento del interés de las personas por involucrarse en política, el desarrollo de este tipo de herramientas se hace cada vez mas necesario.

Teniendo en cuenta este contexto, se plantea el desarrollo de una aplicación que proporcione acceso a nuevas formas de participación desde dispositivos móviles. Dicha aplicación tendrá dos objetivos claros: poner en un primer plano los programas electorales (que normalmente pocas personas leen) incentivando su lectura y debate; y abrir un espacio común en el que la ciudadanía exprese sus propuestas alternativas a las soluciones expuestas por los partidos tradicionales que les representan.

Nos basaremos en el uso de tecnologías open-source ya existentes de edición colaborativa en tiempo real, adaptándolas a dispositivos móviles. Concretamente se explorará el uso de la plataforma web Apache Wave, llevando a cabo un proceso de migración que permita explotar su potencial en dispositivos basados en Android. Haciendo uso de esta tecnología los usuarios tendrán la oportunidad de desarrollar textos colaborativos en tiempo real.

Como resultado de este trabajo se desarrollará una primera versión de la aplicación que, a modo de prueba de concepto, haga uso de las funcionalidades antes descritas. De esta manera el objetivo final que nos planteamos es poner en práctica su uso durante futuras citas electorales.

Palabras Clave:

Desarrollo Colaborativo, Política, Democracia, Participación Ciudadana, Tiempo Real, Android, Apache Wave, Elecciones, Programas Electorales, Propuestas.

Capítulo 1

Introducción

1.1. Objetivos del Proyecto

Los objetivos son los siguientes:

- 1^a parte: Migración de Wave a Android
 - Estudiar la implementacion actual de Wave en Java y GWT.
 - Adaptar la implementación para hacer uso de las características nativas de Android.
- 2^a 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: Descripción de la tecnología 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

Construyendo la idea de la aplicación: DemoCritics

2.1. Introducción

Una vez que habíamos migrado la tecnología de Wave que soportaría el núcleo de nuestra aplicación, era hora de decidir qué aplicación le íbamos a dar de cara a desarrollar una app Android que hiciera uso de ello. Era importante tener en cuenta las características que nos ofrecía Wave:

- Edición colaborativa.
- Consistencia en Tiempo Real.

Brainstorming de ideas

EXPLICAR UN POQUILLO BRAINSTORMING

Después de darle nosotros mismo unas cuantas vueltas a posibles ideas de implementaciones que podrían hacer uso de estas características, decidimos realizar una sesión de brainstorming junto a nuestros profesores para identificar ideas potenciales. En esta sesión aparecieron temas tan variados como wikis colaborativas, aplicaciones de inteligencia artificial, aportaciones colaborativas en política, edición de vídeos y música, cursos de formación colaborativos, visualización de mapas, etc.

CAPÍTULO 2. CONSTRUYENDO LA IDEA DE LA APLICACIÓN: DEMOCRITICS

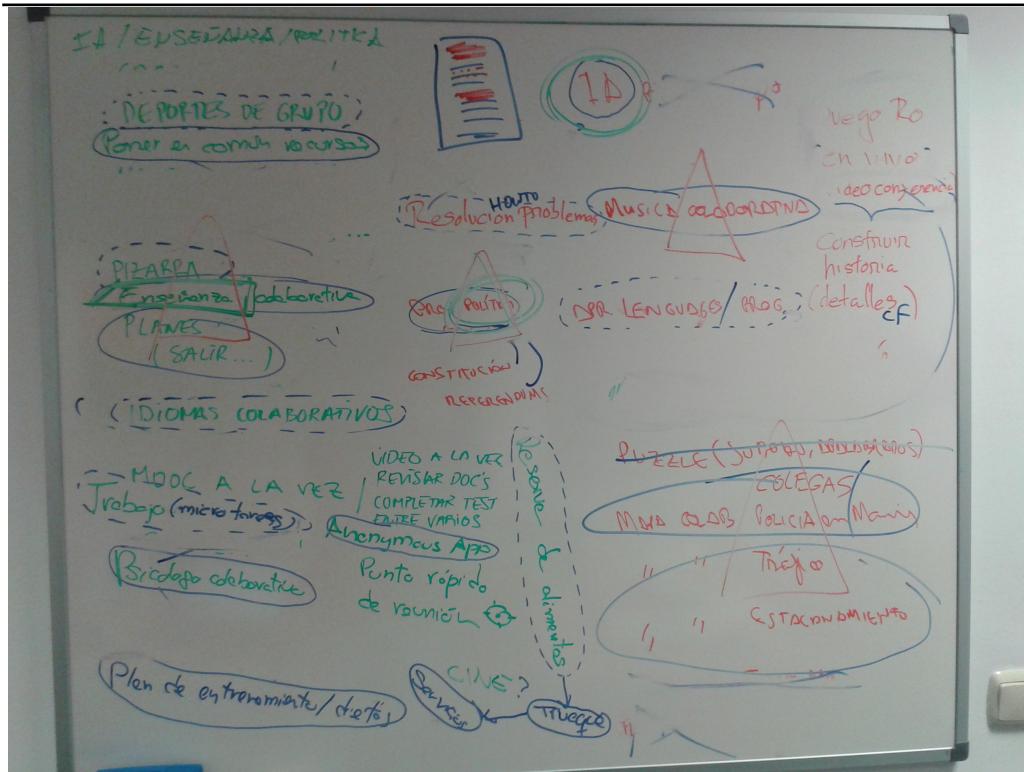


Figura 2.1: Brainstorming sobre la idea a desarrollar

Con un gran repertorio de ideas expuestas en la sesión, decidimos centrarnos primero en descartar aquellas que a nosotros no nos motivaba llevar a cabo. De esta manera nos quedamos con cuatro ideas fundamentales a desarrollar en nuestra aplicación: Política, Música, Inteligencia Artificial y Mapas. Centrándonos ahora solo en estos temas, 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, etc.

Finalmente, ya que ambos teníamos interés por la política, decidimos realizar una aplicación colaborativa relacionada con dicho mundo. En esta aplicación podríamos recurrir a la edición de contenidos en tiempo real, ya fueran propuestas políticas, programas electorales u otro tipo de documentos. Más adelante también podríamos hacer uso incluso de alguna herramienta de Inteligencia Artificial para automatizar algunas tareas o realizar recomendaciones sociales.

Lo que sí que tuvimos claro desde el principio es la idoneidad del momento

CAPÍTULO 2. CONSTRUYENDO LA IDEA DE LA APLICACIÓN: DEMOCRITICS

actual para desarrollar una app de temática política, dado que nos encontramos en año electoral. Nos propusimos el objetivo de desarrollar algo que pudiera tener cierta repercusión y utilidad en las próximas citas electorales de este año 2015. Intentaríamos pensar en la aplicación no solo como un Trabajo de Fin de Grado sino como algo que pudiéramos llevar más allá y que resultara útil a la sociedad.

2.2. Marco teórico de la idea

En los siguientes puntos se discutirá acerca de una serie de consideraciones sobre el estado actual de la política y la democracia. Nos centraremos sobre todo en su relación con las nuevas tecnologías como herramientas capaces de cambiar la manera en la que se conciben ambas disciplinas.

2.2.1. Política en el mundo de la Informática

A primera vista podemos pensar que la informática no parece entusiasmar a los informáticos. Podemos ubicar la política como una parte de las ciencias sociales, situando la informática en ciencias formales. Pero si pensamos en factores como la gestión de los privilegios de una aplicación entre los que definiremos de alguna forma una jerarquía, estaremos en cierta manera haciendo política. También encontraremos características políticas en el diseño relacional de una base de datos. Definiendo los campos de una base de datos podemos encontrarnos 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 de funcionamiento de la base de datos establecidas por unos principios políticos.

Además si nos sumergimos 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 o distribuir copias. Multitud de reglas políticas” definidas en un documento de 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

CAPÍTULO 2. CONSTRUYENDO LA IDEA DE LA APLICACIÓN: DEMOCRITICS

permisos con los que operar libremente en tu dispositivo.

Observamos por tanto que la política está más integrada en la informática de lo que parece, sobre todo si dejamos a un lado la informática más científica y formal y pasamos a la informática social, la de los gobiernos, la de los negocios o la de las relaciones sociales.

2.2.2. Democracia

Democracia representativa

Democracia participativa

Democracia directa

Democracia deliberativa

2.3. Adentrándonos en la idea

La idea a desarrollar está generada en una época en la que la política parece haber despertado el interés de una parte considerable de la ciudadanía. Podría ser por tanto una herramienta útil para participar en temas políticos de forma sencilla y atractiva. Dejando así atrás los tópicos a menudo escuchados de *“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 expuestos por las diferentes formaciones políticas. De forma que, para potenciar el uso social de la aplicación, los ciudadanos podrían leer aquellos puntos de los programas más vistos, debatidos, comentados, etc. Así, cualquier usuario tendría a su disposición todos los programas electorales en su bolsillo, por lo que no tendría que ir a la página web de cada formación política y descargarse un documento de 200 páginas. Pensamos que esta forma tradicional de presentar un programa político en un solo documento en un mundo donde las posibilidades de comunicarnos se han desarrollado exponencialmente mediante las nuevas tecnologías no es la mejor manera de generar interés por su lectura y la implicación en política de las personas.

Por otra parte, y teniendo en cuenta la tendencia actual de los nuevos movimientos ciudadanos de elaborar programas políticos en base a propuestas

CAPÍTULO 2. CONSTRUYENDO LA IDEA DE LA APLICACIÓN: DEMOCRITICS

de los ciudadanos, la aplicación también debía ofrecer alguna manera de realizar Propuestas y debatirlas entre todos. De esta forma tanto la ciudadanía como las formaciones políticas podrían saber en cualquier momento cuáles son las principales preocupaciones de los ciudadanos y qué medidas o soluciones proponen para resolverlas. Además pensamos que podríamos aprovechar las características de Wave para realizar estas Propuestas de forma colaborativa y en tiempo real, aportando un valor diferenciador respecto a las actuales soluciones desarrolladas para web (Ver sección 3.3.2).

Por tanto, desde un primer punto de vista subjetivo, la aplicación quedó dividida en dos partes. Por un lado tendríamos la presentación estructurada de los programas políticos que presentan las formaciones políticas. Y por otro todas las propuestas que elaboran de forma colaborativa los ciudadanos, ya sea individualmente o en colectivos sociales.

Capítulo 3

Estado del Arte

3.1. Wave

Wave es a la vez un protocolo de comunicaciones basado en XMPP[1] y una plataforma web de código libre, que permiten a sus usuarios comunicarse y colaborar entre sí en tiempo real y de forma federada (Ver sección 4.1.3) a través de Internet. Fue anunciado originalmente por Google en 2009 [2] con la intención de aunar en una sola plataforma servicios como el correo electrónico, las redes sociales y la mensajería instantánea. Sin embargo, debido a su poca aceptación, en 2010 Google abandona el proyecto [3] y reorienta la tecnología a su plataforma Google Docs [4]. Desde ese momento el proyecto original pasa a manos de la Apache Software Foundation bajo el nombre de Apache Wave.

3.1.1. Apache Wave: Wave In A Box (WIAB)

Al pasar el proyecto a sus manos, como software de código libre bajo licencia Apache [5], el desarrollo se centra en conseguir implementar una plataforma que integre un servidor Wave y un cliente web sencillo donde crear y manipular el contenido de las waves. A esta plataforma se le denomina Wave In A Box (WIAB) y actualmente sigue en desarrollo por parte de la comunidad y se distribuye en forma de código fuente, accesible entre otras formas desde su repositorio de GitHub [6]. Cualquiera puede descargar y "jugar" con WIAB en su ordenador siguiendo los pasos que nos proporcionan en su wiki [7]. Existen también servidores de prueba ya desplegados en Internet sobre los que se puede observar el funcionamiento de WIAB [8].

CAPÍTULO 3. ESTADO DEL ARTE

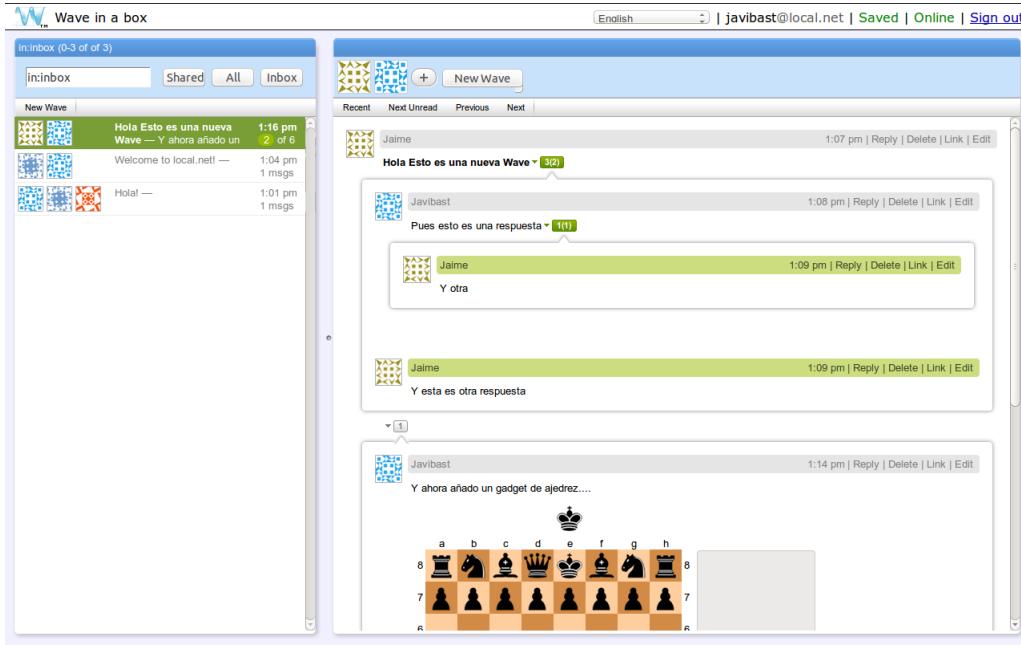


Figura 3.1: Cliente Wave In A Box

3.2. Edición Colaborativa en Tiempo Real

En esta sección veremos algunas de las soluciones disponibles actualmente que, al igual que Wave, permiten editar contenido de forma colaborativa y en tiempo real. Sin embargo, todas estas plataformas utilizan una arquitectura de servidor centralizado para ofrecer estas funcionalidades, mientras que Wave utiliza una arquitectura federada en la que no existe un servidor central (Ver Sección 4.1.3).

De esta manera, primero veremos las principales herramientas de desarrollo (APIs) más utilizadas actualmente, para después pasar a ver algunas de las plataformas que hacen uso de esta tecnología. En su mayoría son para web, aunque existen unas pocas para Android.

3.2.1. APIs Centralizadas

La mayor parte del desarrollo de tecnologías de Colaboración en Tiempo Real (RTC) se realiza mediante APIs que son propiedad de determinadas compañías y que no utilizan una arquitectura federada, sino que toda la

CAPÍTULO 3. ESTADO DEL ARTE

información pasa por un servidor central que controla el RTC. A continuación veremos algunas de las más utilizadas hoy en día.

Google Realtime API

Google Realtime API [9] permite construir aplicaciones de colaboración en tiempo real utilizando la tecnología de Transformaciones Operacionales (OT) presente en Google Docs. Utiliza JavaScript para poder construir en nuestro cliente web un modelo de datos (Realtime Data Model) que se guarda en sus servidores y gestiona automáticamente los cambios realizados por cualquiera de los usuarios que colaboran en tiempo real. Estos cambios en el modelo son notificados al servidor y al resto de usuarios mediante eventos que se lanzan al modificar los datos sobre los que se está colaborando. Para utilizarlo es necesario tener cuenta en la Consola de Desarrolladores de Google y activar el uso de esta API con nuestras credenciales de usuario.

Microsoft RTC Client API

Microsoft RTC Client API [10] permite construir aplicaciones que permitan realizar llamadas de audio/video o sesiones de mensajería instantánea (IM) de texto por Internet y en tiempo real. Las aplicaciones se deben escribir en C++ o Visual Basic y pueden ser utilizadas tanto en PC como en dispositivos móviles siempre que ejecuten un sistema operativo Windows.

WebRTC

WebRTC [11] (Web Real-Time Communication) es un API open-source (bajo licencia BSD) actualmente en desarrollo por Google y la World Wide Web Consortium (W3C) y que pretende dotar a los navegadores web de capacidades de comunicación en tiempo real entre sí sin necesidad de plugins externos. En la actualidad se encuentra en su versión 1.0 y soporta los navegadores Firefox y Chrome.

Mozilla TogetherJS

Mozilla TogetherJS [12] es una librería gratuita y de código libre (bajo licencia pública Mozilla v2) que permite añadir capacidades de colaboración en tiempo real a una página web. Utiliza WebRTC y webSockets para establecer comunicaciones peer-to-peer (P2P) entre dos o más navegadores Web. No

proporciona almacenamiento persistente de los datos y es necesario tener un Servidor que establezca la conexión. Con esta herramienta se puede editar texto en tiempo real (usando Transformaciones Operacionales), establecer chats de audio/video y sincronizar el contenido de los navegadores. Dispone de GitHub para contribuir a su desarrollo.

ShareJS

ShareJS [13] es una plataforma open-source (bajo licencia MIT) que dispone de un pequeño servidor basado en Node.js y una librería de cliente JavaScript que permiten la edición colaborativa de contenido mediante Transformaciones Operacionales. Permite actuar sobre objetos JSON o sobre texto plano. Dispone de GitHub para contribuir a su desarrollo.

Goodow

Goodow [14] es un framework open-source de reciente desarrollo que proporciona un API muy similar a la de Google (Ver Sección 3.2.1) para colaboración en tiempo real mediante el uso de Transformaciones Operacionales. Dispone asimismo de dos clientes básicos para Android e iOS, utilizando una implementación del Servidor propia.

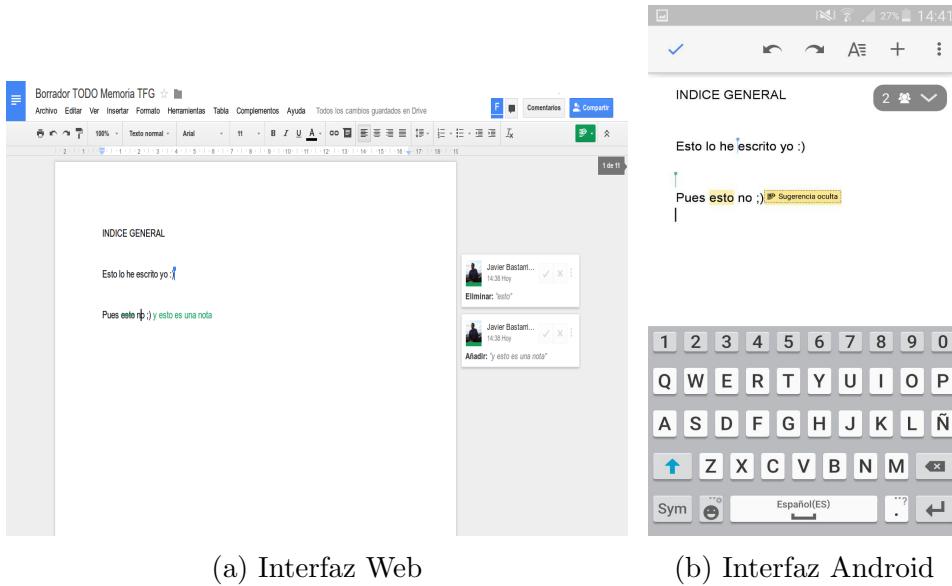
3.2.2. Plataformas Web y Android

En las siguientes secciones veremos algunas de las plataformas Web y Android que hacen uso de tecnologías de Colaboración en Tiempo Real.

Google Docs

Google Docs [4] es la plataforma de Google para edición de documentos de forma colaborativa y en Tiempo Real usando su API Realtime (Ver Sección 3.2.1). Permite que varios usuarios con cuenta de Google creen y editen colaborativamente un documento a la vez. Puedes ver los cursos de cada usuario e interactuar con ellos mediante un chat. También permite escribir sugerencias a modo de notas en el margen sobre lo ya escrito. Dispone de versión web y móvil, pudiendo interactuar entre ellas sin problemas.

CAPÍTULO 3. ESTADO DEL ARTE



(a) Interfaz Web

(b) Interfaz Android

Figura 3.2: Capturas de Google Docs

Etherpad

Etherpad [15] es un editor colaborativo en tiempo real open-source (bajo licencia Apache 2.0). Permite a varios autores editar a la vez un mismo documento de texto, resaltando en distintos colores lo editado por cada persona y con la opción de un chat para comunicarse entre sí. Existen múltiples servicios que hacen uso de este editor, siendo uno de las más conocidas TitanPad [16].

CAPÍTULO 3. ESTADO DEL ARTE

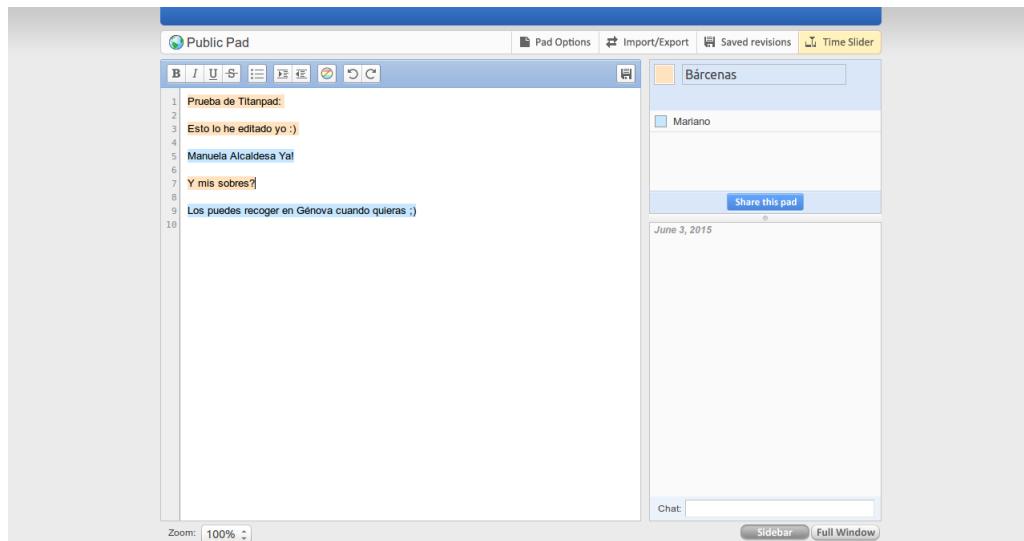


Figura 3.3: Captura de TitanPad: Ejemplo de uso de EtherPad

Colorillo

Colorillo [17] es una aplicación web básica de dibujo colaborativo en tiempo real. Cualquier usuario puede empezar a dibujar sobre un nuevo lienzo en blanco con diversos colores y compartir este lienzo con otros usuarios para dibujar entre todos. De cada usuario podemos ver su procedencia aproximada sobre un mapa y el color que actualmente está utilizando. Existe también opción para chatear con otros usuarios. Los dibujos se pueden descargar y tienen todos licencia Creative Commons BY 3.0.

CAPÍTULO 3. ESTADO DEL ARTE

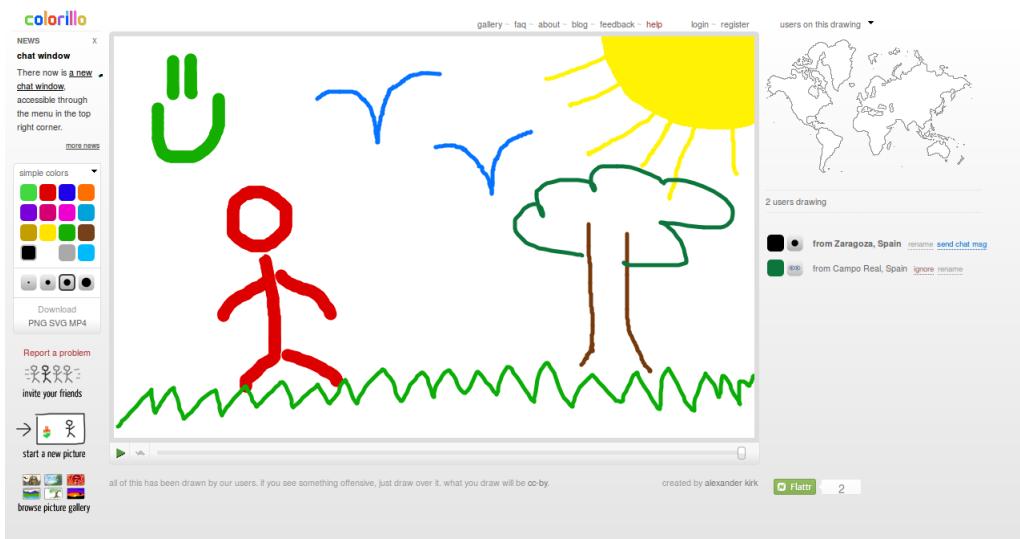


Figura 3.4: Captura de Colorillo

ShareLaTeX

ShareLaTeX [18] es un editor web colaborativo de documentos escritos en LaTeX en tiempo real. Permite elaborar documentos LaTeX entre varias personas, ofreciendo una interfaz que incluye una previsualización del resultado en PDF. Desde 2014 es open-source (bajo licencia AGPL v3) y cualquiera puede descargarselo de GitHub e instalar su propio servidor (escrito en Node.js) de ShareLaTeX. En su web ofrecen también opciones de pago con extras como un control de versiones o sincronización con Dropbox.

CAPÍTULO 3. ESTADO DEL ARTE

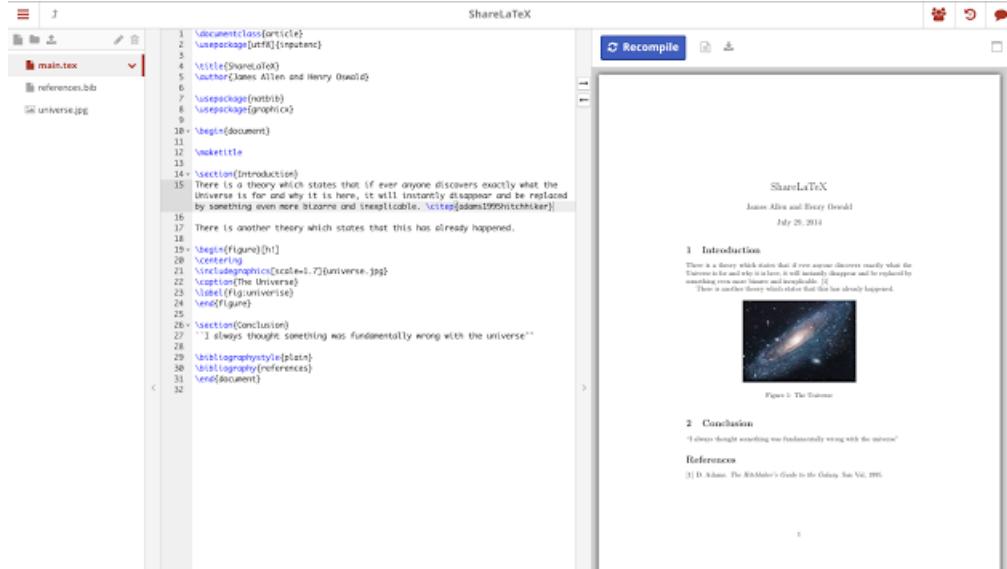


Figura 3.5: Captura de ShareLaTeX

Samepage

Samepage [19] es una aplicación, tanto para web como para plataformas móviles (Android e iOS), que permite crear páginas que pueden ser editadas de forma colaborativa y en tiempo real por múltiples usuarios. Para ello es necesario solo tener una cuenta de samepage. Tanto el cliente web como el móvil permiten interactuar entre ellos para crear nuevas páginas, editar texto y hacer comentarios, aunque la versión web permite también añadir tablas, imágenes y archivos.

CAPÍTULO 3. ESTADO DEL ARTE

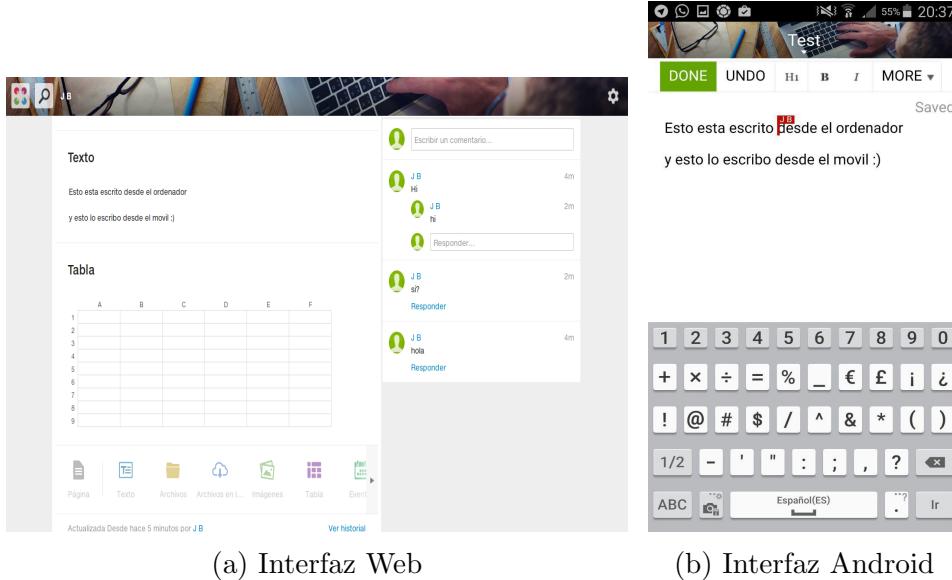


Figura 3.6: Capturas de Samepage

Quip

Quip [20] es una aplicación para dispositivos móviles (Android e iOS) desarrollada con el objetivo de aumentar la productividad en los trabajos en grupo. Permite elaborar documentos, hojas de cálculo y listas de tareas compartidas y editables de forma colaborativa en tiempo real, pudiendo realizar también comentarios sobre ellas. Dispone también de una versión web, pero es necesario tener cuenta para usarla.

CAPÍTULO 3. ESTADO DEL ARTE

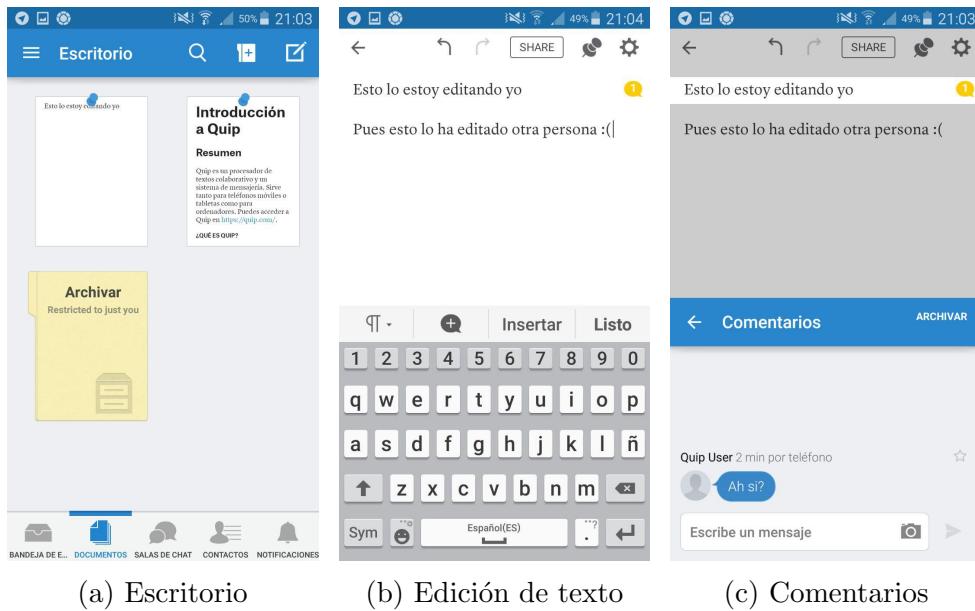


Figura 3.7: Capturas de Quip

3.3. Aplicación Android: DemoCritics

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

3.3.1. Programas Políticos

En la actualidad no existe ningún tipo de aplicación móvil orientada a debatir los programas electorales de los partidos políticos en su conjunto. Concretamente no existe ningún tipo de plataforma que agrupe en un solo sitio los programas electorales de las diferentes candidaturas. Lo más parecido que hemos podido encontrar han sido aplicaciones elaboradas por un partido político, orientadas a dar a conocer su candidatura. En ellas podemos ver normalmente, entre otros, a presentación de candidatura, vídeos propagandísticos y el programa electoral.

Pasamos ahora a analizar algunas de las aplicaciones móviles encontradas, identificando en cada caso aspectos e ideas que nos han resultado positivos y negativos.

CAPÍTULO 3. ESTADO DEL ARTE

UPyD Parla

La aplicación presenta al candidato de UpyD Carlos Alt Bustelo para la alcaldía de Parla. Se trata de una alicación 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, etc.



Figura 3.8: Capturas de UPyD Parla

- Aspectos positivos:

- Presentación de Programa Electoral estructurado con Indice inicial.
- El Programa se lee dentro de la app, no nos lleva a leer el programa en PDF de la web.
- Presentación de una Sección del Programa Electoral de forma resumida, teniendo la opción de leer la sección entera al pulsar un botón.

- Aspectos negativos:

- Posee una sección llamada "Memes" cuyo nombre no se entiende ya que se limita a mostrar carteles propagandísticos de la candidatura.

CAPÍTULO 3. ESTADO DEL ARTE

#RecuperaCórdoba

Esta app presenta la candidatura de Pedro García de Izquierda Unida a la provincia de Córdoba, informando de su propuesta de gobierno de forma resumida. 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.

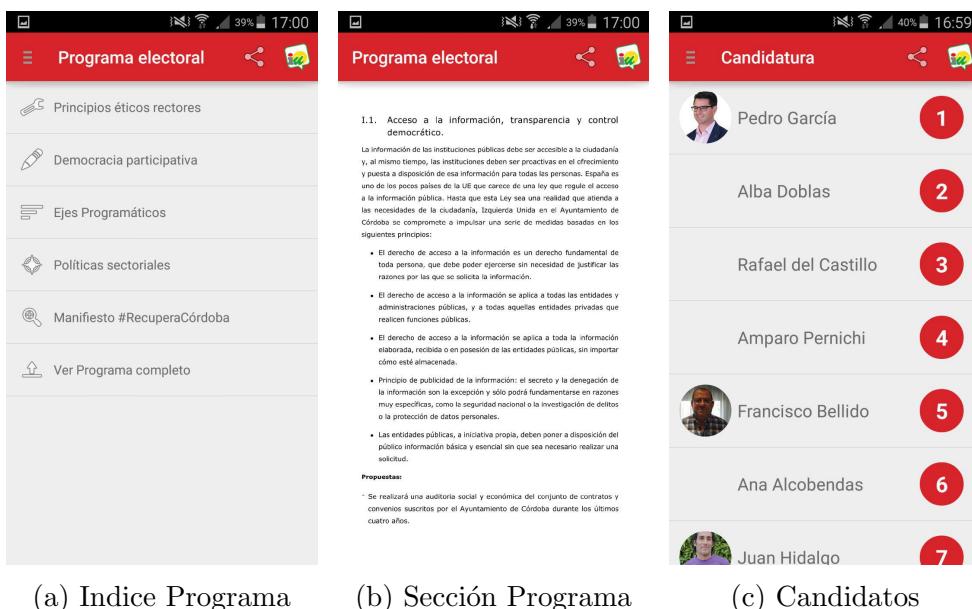


Figura 3.9: Capturas de #IURecuperaCórdoba

- Aspectos positivos:

- Interfaz limpia y sencilla de diseño plano.
- Consistencia en la aplicación: la información se presenta siempre en formato de lista ofreciendo los mínimos datos necesarios sin sobrecargar de información al usuario.
- Menú lateral disponible en cualquier pantalla con las principales acciones de la aplicación: Candidatos, Programa, Videos y Noticias.
- El programa electoral está estructurado en un índice primer nivel y a veces con segundo nivel.
- Aporta la opción de ver el programa completo.

CAPÍTULO 3. ESTADO DEL ARTE

- Aspectos negativos:

- Utiliza a veces iconos cuyo propósito no se entiende: ¿Un avión de papel para noticias? ¿Un "a&z" para la lista de candidatos?
- Las distintas secciones se abren dentro de la aplicación, pero da acceso a una navegación lateral por las páginas del PDF del programa en cuestión.
- Si haces zoom en una sección no permite pasar de página.

PSOE Andalucía

Esta app presenta la candidatura del PSOE a la junta de Andalucía para las elecciones del 22 de Marzo, promocionando básicamente su programa electoral y a la candidata Susana Díaz. Permite también estar al día de noticias y eventos relacionados con dicha candidatura.

La navegación por el programa, aunque estructurada en un primer nivel, se realiza directamente visualizando páginas que parecen extraídas del programa en PDF.

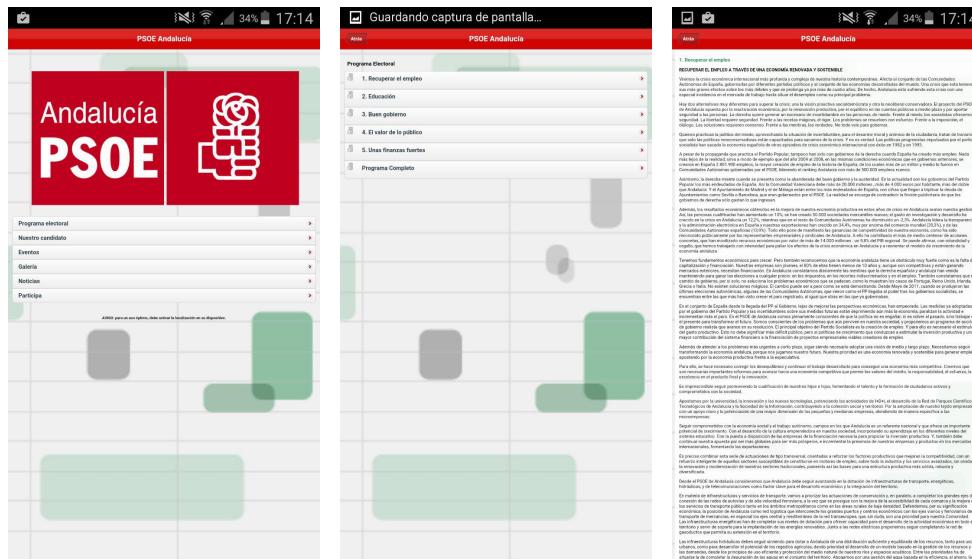


Figura 3.10: Capturas de PSOE Andalucía

- Aspectos positivos:

CAPÍTULO 3. ESTADO DEL ARTE

- Posee un indice de primer nivel para estructurar el programa.

- Aspectos negativos:

- La interfaz y los botones no se adaptan al tamaño de pantalla y permanecen de un tamaño pequeño que dificulta la interacción.
- El programa electoral se visiona en forma de una página que parece descargada directamente de la versión PDF y que permanece en un tamaño pequeño e ilegible, no permitiendo tampoco hacer zoom. En general, la interfaz parece hecha para una web más que para un móvil.

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 supuesto ver el programa electoral.

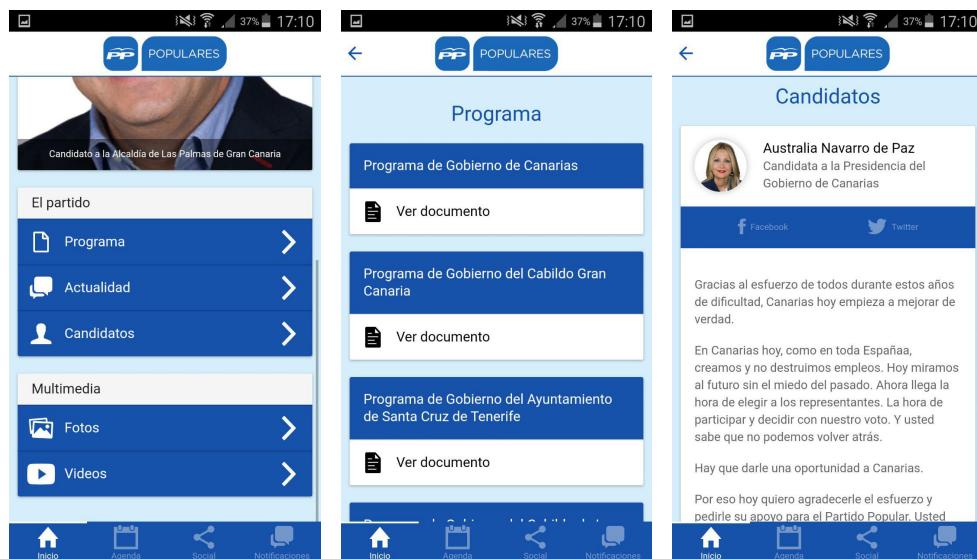


Figura 3.11: Capturas de PP Canarias

- Aspectos positivos:

CAPÍTULO 3. ESTADO DEL ARTE

- Interfaz limpia y atractiva con una organización clara en secciones que aparecen en una barra inferior al estilo de iOS.

- Aspectos negativos:

- Aunque dispone de una sección para programas electorales, no dispone de ellos en local, si no que te obliga a abrir un navegador para ver la versión entera en PDF.
- El problema de la barra inferior de menús es que quita espacio al contenido principal. Se podría haber puesto el menú en alun sitio menos intrusivo.

3.3.2. Participación Ciudadana

Centrándonos en la participación ciudadana ya sea mediante la generación de Propuestas, el desarrollo colaborativo de programas o la 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 [21] es una plataforma web de código libre donde los usuarios pueden crear temas, propuestas o compartir enlaces web a otros sitios. A primera vista puede parecer un foro, aunque la principal diferencia respecto a éste último radica en que otros usuarios pueden votar a favor o en contra de los enlaces, haciendo que el sistema los haga aparecer como más o menos destacados. De esta forma los temas de conversación, enlaces, o propuestas aparecerán en el orden que haya escogido la comunidad según la puntuación positiva o negativa que le hayan dado. En principio el uso de reddit está destinado a todo tipo de temas, entre los que podemos encontrar algunos ejemplos fuertemente relacionados con la participación ciudadana. Es el caso de Plaza Podemos [22]: un espacio utilizado para que la ciudadanía pueda expresar sus propuestas, compartir noticias relacionadas con la actualidad política o debatir aquellos temas que más les preocupan. Así, aunque no seamos participantes de reddit, de un simple vistazo podemos saber qué es lo más debatido por la ciudadanía, las propuestas que quieren llevar a cabo en el gobierno o cuáles son los temas que más les preocupan.

CAPÍTULO 3. ESTADO DEL ARTE



Figura 3.12: Plaza Podemos utilizando la plataforma Reddit

- Aspectos positivos:

- Opción de filtrado por tipos de contenido (propuestas, noticias...) y ordenación de distintas formas (nuevos, populares, activos...)
- Sistema de votación sencillo desde la propia previsualización del contenido mediante flechas (arriba y abajo), pudiendo ver entre ellas el número actual de votos.

- Aspectos negativos:

- Para un usuario que lo usa por primera vez puede resultar confuso que haya contenido redactado con usuarios mezclado con enlaces a noticias externas.

Change.org

Change.org [23] es un portal web que permite lanzar múltiples peticiones de cambio en Internet. Podríamos definirlo como la evolución de la recogida

CAPÍTULO 3. ESTADO DEL ARTE

de firmas en la calle: cualquiera puede realizar una petición para solicitar el apoyo de otros. Las personas que decidan apoyar la petición, dejarán sus datos personales y constarán entre el número de personas que han firmado a favor de la petición. Una vez que han alcanzado un número objetivo de apoyos se procede a entregar las firmas digitales al organismo, persona o entidad a la que va destinada la petición.

Por ejemplo: en mayo de 2011, en relación con las movilizaciones del Movimiento 15-M y Democracia Real Ya, y ante el desalojo por los Mossos de Esquadra se llevó a cabo la petición “Exige la dimisión fulminante del Coneller 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, etc.



Figura 3.13: Change.org · La mayor plataforma de peticiones del mundo

- Aspectos positivos:

- Página principal con peticiones cuyo objetivo de firmas se ha conseguido ("victoria") y las más destacadas aun por conseguir.

CAPÍTULO 3. ESTADO DEL ARTE

- De cada petición se muestra un "preview" con los aspectos más destacados: foto, título, autor, número de firmas y fecha de creación.

- Aspectos negativos:

- Aunque existen filtros para peticiones (Destacadas, Populares y Recientes) ¿de qué depende esta clasificación? No queda claro cómo se elige la opción en dicha lista de peticiones,

Programas Colaborativos de Ahora Madrid y Zaragoza en Común

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

El resultado final fue determinar las cinco propuestas más votadas que fueron incluidas en el programa final como medidas urgentes para realizar en los 100 primeros días de gobierno.

CAPÍTULO 3. ESTADO DEL ARTE



Figura 3.14: Creación colaborativa del programa de Ahora Madrid.

También utilizaron una plataforma similar en la candidatura zaragozana de unidad popular Zaragoza en Común [24].

The screenshot shows the header of the website, which includes the 'GANEMOS Zaragoza' logo and navigation links for 'INICIO' and 'ENTRAR'. The main title is 'Programa colaborativo de Zaragoza en Común' in red. Below it, a message says 'Tienes que estar identificado para votar o comentar las propuestas.' The page is divided into sections:

- Un programa elaborado en Común**: A text block explaining the process of collective program elaboration.
- Ejes temáticos**: A section listing thematic axes with small images:
 - Modelo de ciudad**: An image of a cityscape.
 - Derechos sociales**: An image of children in a group discussion.
 - Economía, trabajo y desigualdad**: An image of people working at a table.

Figura 3.15: Creación colaborativa del programa de Zaragoza en Común.

CAPÍTULO 3. ESTADO DEL ARTE

- Aspectos positivos:

- Las propuestas están organizadas por distintos temas: "Ejes temáticos" en el caso de Zaragoza en Común y "Areas y Objetivos" en el caso de Ahora Madrid.
- En ambos casos se puede filtrar la lista de propuestas de distintas formas ("Más valoradas", "Más consenso" y "Más debatidas")
- En el "preview" de la propuesta se muestra el número de comentarios y el porcentaje de votos positivos en relación al número total de votos recibidos.

- Aspectos negativos:

- En cada propuesta se muestra un número a la izquierda que no explica lo que significa (entendemos que es el número de votos a favor).

Appgree

Appgree[25] es una plataforma desarrollada con el objetivo de poner a grandes grupos de personas de acuerdo en poco tiempo. Está disponible tanto para web como para móviles y permite que sus usuarios lancen propuestas y debatan sobre cualquier tema pudiendo votar y alcanzar un consenso, obteniendo los resultados de dicha votación casi en tiempo real gracias a un algoritmo estadístico desarrollado por ellos llamado DemoRank[26].

En la aplicación podemos acceder a una lista de canales: que aglutinan encuestas, propuestas y preguntas (ya sea de respuesta abierta o de votación entre dos o mas opciones), pudiendo votar y ver los resultados actuales de votación y respuestas. Para fomentar la participación potencian mucho el uso tops de preguntas más candentes y recientes. Además se pueden compartir preguntas por redes sociales para aumentar su difusión.

CAPÍTULO 3. ESTADO DEL ARTE

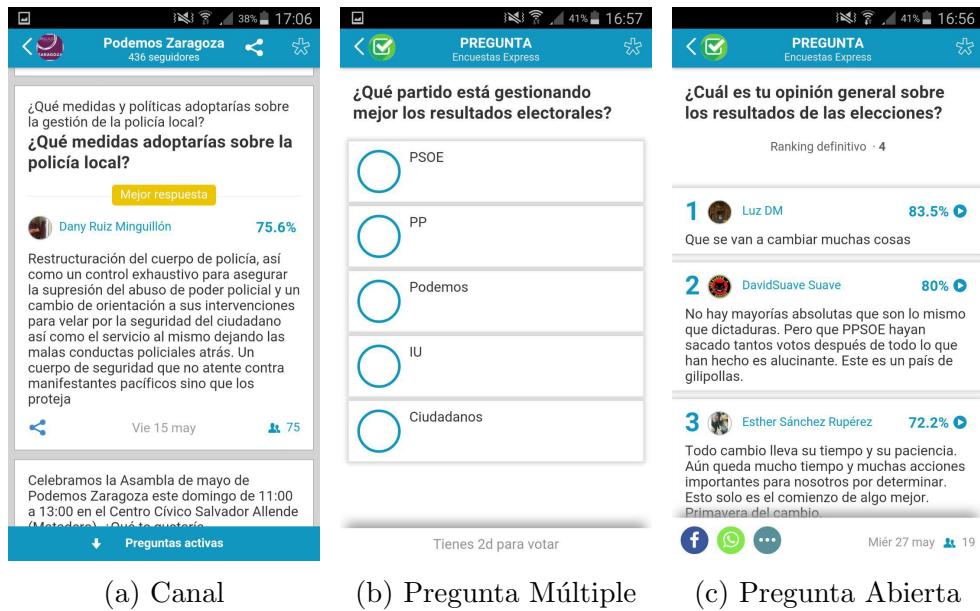


Figura 3.16: Capturas de Appgreet

- Aspectos positivos:

- Interfaz limpia de colores planos (blanco y azul) con pantalla principal de ultimas preguntas destacadas.
- Organización por canales temáticos de las preguntas.
- Opción de marcar canales como favoritos para poder seguirlos y ser notificados de las últimas preguntas añadidas.

- Aspectos negativos:

- Las preguntas se organizan por antigüedad (activas y no activas) pero se navega de abajo hacia arriba, cuando uno esperaría hacerlo al revés.
- Los tipos de preguntas (abiertas, de respuesta múltiple, de sí o no, encuestas...) aparecen todas mezcladas.

Capítulo 4

Tecnologías del Proyecto

En las siguientes secciones explicaremos brevemente las tecnologías y herramientas que hemos utilizado para llevar a cabo este proyecto. Concretamente hablaremos de las tecnologías utilizadas durante la Migración del cliente web de SwellRT (Wave) a Android y durante el desarrollo de la aplicación Android que hará uso del resultado de dicha Migración.

En el caso de Wave, al tratarse de una tecnología bastante reciente, se explicará más detalladamente en qué consiste esta tecnología, pero para el resto de casos se hará una pequeña reseña sobre en qué consiste la tecnología y sus usos. En todos los casos se intentará también explicar de qué manera y para qué se ha utilizado dentro de este proyecto.

Para hacerse una idea general de las tecnologías utilizadas se recomienda ver el resumen que se muestra en la Tabla 4.1.



Figura 4.1: Nube de Tecnologías utilizadas en el Proyecto

CAPÍTULO 4. TECNOLOGÍAS DEL PROYECTO

Tipo	Nombre	Uso	Sección
API	SwellRT	Base de la que se parte para la Migración de Wave a Android	4.1.4
	Android	Plataforma para la Migración y el Desarrollo de la app.	4.2.2
Servidor	Wave In A Box (WIAB)	Servidor que aloja las Waves. Migración y Desarrollo de la App.	4.1.4
	OpenShift	Servidor que aloja el Service REST y la Base de Datos de la App.	4.2.10
Lenguaje	Java	Lenguaje base para trabajar con Android, tanto en la Migración como en la App.	4.1.6
	JavaScript	Lenguaje del antiguo cliente web de Wave presente en SwellRT.	4.1.8
	PHP	Lenguaje para definición del Service REST de la App.	4.2.8
	SQL	Lenguaje para definir e interactuar con la Base de Datos de la App.	4.2.5
	XML	Lenguaje para definición de interfaces en Android.	4.2.3
Formato de Datos	JSON	Formato para intercambio de información con Servidor Wave y REST.	4.2.4
Framework	GWT	Framework JavaScript utilizado por el antiguo Cliente web de Wave presente en SwellRT.	4.1.7
	Laravel 5	Framework de desarrollo del Service REST de la App en PHP.	4.2.9
Protocolo	Google Wave	Protocolo de intercambio de Waves usado para la migración y la App.	4.1.1
	HTTP	Protocolo de transferencia de datos por Internet usado por la Migración y por la app.	4.1.9
	WebSocket	Protocolo de transferencia de datos bidireccionales por Internet, usado por la Migración y por la App.	4.1.10
Base de Datos	MySQL	Sistema Gestor de la Base de Datos de la App	4.2.6
	PhpMyAdmin	Herramienta de Administración de la Base de Datos de la App.	4.2.7
Control de Versiones	Git	Software de control de versiones usado para la Migración y la App.	4.1.11
	GitHub	Plataforma web de control de versiones que aloja los desarrollos de la Migración y de la App.	4.1.12
Prototipado	POP	Aplicación para elaborar los Prototipos Interactivos basados en los Prototipos en Papel de la App.	4.2.12
IDE	Eclipse Luna	Entorno de Desarrollo utilizado para la Migración de Wave.	4.1.5
	Android Studio	Entorno de Desarrollo utilizado para el desarrollo de la App.	4.2.1
	Sublime Text	Editor de texto y de código fuente, utilizado para desarrollar el Service REST.	4.2.11

Cuadro 4.1: Tecnologías usadas en el Proyecto

4.1. Tecnologías de Wave

4.1.1. Google Wave

Ideado y presentado en 2009 por ingenieros de Google [2], Wave es a la vez un protocolo de comunicaciones [1] y una plataforma web de código libre, que permiten a sus usuarios comunicarse y colaborar entre sí en tiempo real (Ver sección 4.1.3) y de forma federada (Ver sección 4.1.3) 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 + [27]. Es en este momento cuando el desarrollo libre del proyecto pasa a manos de la Apache Software Foundation bajo el nombre de Apache Wave.

4.1.2. Apache Wave

Al cambiar de manos su desarrollo en 2010, la tecnología pasa a formar parte de la incubadora de la fundación Apache [28] como software de código libre bajo licencia Apache [5]. Así, se produce el desarrollo de Wave In a Box (WIAB) (Ver sección 4.1.4), plataforma que integra un cliente web sencillo y una implementación de un servidor Wave que cualquiera puede descargar y desplegar en su ordenador.

4.1.3. 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 [1] fue desarrollado para utilizar un modelo federado [29] [30] de comunicación basado en la tecnología XMPP [31] [1]. 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 [1] utiliza la tecnología de Transformaciones Operacionales (OT) [32] 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 [33].

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.

4.1.4. Servidores Wave

Wave in a Box

Wave In a Box (WIAB) [34] 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 [35]. La instalación trae consigo un cliente web desarrollado en Javascript usando el framework Google Web Toolkit (GWT) [36]. Este cliente web sirve como

CAPÍTULO 4. TECNOLOGÍAS DEL PROYECTO

prueba de concepto de las funcionalidades básicas del Modelo Conversacional de Wave, pudiendo gestionar waves, usuarios y extensiones. Actualmente cualquiera puede descargar y desplegar WIAB en su ordenador siguiendo los pasos que nos proporcionan en su wiki [7]. La aplicación se distribuye en forma de código fuente, accesible entre otras formas desde su repositorio de GitHub [6]. Existen asimismo servidores de prueba ya desplegados en Internet sobre los que se puede observar el funcionamiento de WIAB [8].

SwellRT

Como parte del proyecto europeo P2PValue [37] existe SwellRT (Swell Real Time), 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 popósito es permitir la integracion 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 [38], 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 [39]. Se pretende con esto que SwellRT haga uso de las funcionalidades nativas de Android.

4.1.5. Eclipse

Eclipse [40] es un Entorno de Desarrollo Integrado (IDE) open-source (bajo licencia Eclipse Public License) desarrollado por la Eclipse Foundation. Lanzado en 2007 alcanza actualmente su versión 4.4, también denominada Eclipse "Luna". Se trata de uno de los IDEs más utilizados que proporciona herramientas que permiten el desarrollo de código para múltiples lenguajes y tecnologías. Posee además la capacidad de extender sus capacidades gracias al uso de plug-ins que se pueden descargar desde su marketplace. Es el caso por ejemplo del plugin ADT (Android Developer Tools), que permite utilizar el SDK de Android para desarrollar para esta plataforma utilizando

Eclipse como IDE. Hasta finales del año 2014 Eclipse + ADT era el IDE recomendado por Google para el desarrollo en su plataforma móvil, aunque recientemente ha pasado a utilizar su propio IDE llamado Android Studio (Ver Sección 4.2.1)

En este proyecto usaremos Eclipse con el plugin ADT como IDE para estudiar el código del cliente web de SwellRT y llevar a cabo una migración a Android que permita hacer uso de las características nativas de esta plataforma.

4.1.6. Java

Java [41] es un lenguaje de programación de propósito general y orientado objetos. Fue desarrollado en 1995 por Sun Microsystems, actualmente parte de Oracle, y se distribuye en forma de JDK (Java Development Kit), cuya versión oficial privativa alcanza hoy en día la 8. Existe no obstante una versión de código libre (bajo licencia GPLv2) llamada Open JDK [35] disponible actualmente en su versión 7. Java tiene como ventaja su independencia del hardware, ya que cualquier código compilado en java se traduce a un formato bytecode que se ejecuta en una Máquina Virtual Java (JVM) que es independiente del hardware que haya por debajo.

En este proyecto usaremos Java para la Migración del cliente SwellRT a Android, pues dicho cliente (y el que desarrolló Google inicialmente) está escrito en su mayor parte en este lenguaje de programación. Lo utilizaremos también para el desarrollo de la app, ya que el SDK de Android se basa en Java para desarrollar código para dicha plataforma.

4.1.7. GWT

GWT [36] (Google Web Toolkit) es un framework de código libre (bajo licencia Apache 2.0) que facilita el desarrollo de aplicaciones web basadas en AJAX y JavaScript. Concretamente el API de GWT permite desarrollar código en Java que posteriormente será traducido a JavaScript. Fue creado por Google en 2006 y actualmente es un proyecto independiente open-source que alcanza ya su versión 2.7

En este proyecto utilizaremos GWT durante la Migración de Wave a Android, ya que una parte significativa del cliente web de SwellRT está desarrollada con esta tecnología y es necesario estudiar el código para sustituirla por código nativo de Android, ya que la plataforma móvil de Google no es compatible con GWT.

4.1.8. JavaScript

JavaScript [42] (JS) es un lenguaje de programación dinámico y débilmente tipado utilizado a menudo para la programación de aplicaciones web del lado del cliente (navegador web), aunque también existen implementaciones para el lado del servidor (como Node.js). Originalmente desarrollado en 1995 por la extinta Netscape Communications, su versión actual es la 1.8.5.

En este proyecto se utilizará JavaScript para estudiar la implementación del cliente web de SwellRT hecha con GWT y JavaScript que deberemos migrar a código nativo de Android, pues la plataforma móvil de Google no es compatible de forma nativa con JavaScript.

4.1.9. HTTP

HTTP [43] (HyperText Transfer Protocol) es el protocolo de capa de aplicación más utilizado para establecer comunicaciones en la World Wide Web. Fue desarrollado en 1996 conjuntamente por la World Wide Web Consortium (W3C) y el Internet Engineering Taskforce (IETF), aunque la definición de su versión más actual (1.1) se encuentra especificada en la serie de RFCs (Request For Comments) 7230 [43]. Se trata de un protocolo orientado a transacciones que siguen un esquema de petición-respuesta entre cliente y servidor.

En este proyecto utilizaremos el protocolo HTTP para establecer las conexiones con ambos servidores: el servidor SwellRT que aloja las Waves y el servidor Openshift que aloja el Service REST y la Base de Datos de la aplicación. Concretamente con este protocolo realizaremos el proceso de login en Wave y las peticiones al Service REST en la aplicación Android.

4.1.10. WebSocket

WebSocket [44] es un protocolo que permite establecer comunicaciones bidireccionales (de cliente a servidor y viceversa) y full-dúplex (de forma simultánea en ambos sentidos) en la red. Permite utilizar la tecnología de los sockets TCP en la capa de aplicación. Se trata de un protocolo estandarizado por la IETF en 2011 en el RFC 6455 [44].

En este proyecto utilizaremos esta tecnología para establecer un canal de comunicación basado en sockets con el servidor Wave de SwellRT, pues la característica de bidireccionalidad full-dúplex es necesaria para aprovechar

la potencialidad de consistencia en tiempo real de Wave. Concretamente en la Migración deberemos sustituir la implementación actual de WebSocket en GWT por otra que sea compatible de forma nativa con Android.

4.1.11. Git

Git [45] es un Sistema de Control de Versiones (VCS) distribuido y open-source (bajo licencia pública GNU) diseñado para gestionar las distintas versiones de una aplicación independientemente del tamaño de la aplicación y del número de personas que trabajan en ella. Fue creado por Linus Torvalds en 2005 para trabajar en el desarrollo del kernel de Linux, aunque su versión actual (2.4.2) es una de las herramientas de control de versiones mas utilizada para gestionar proyectos software. Su interfaz es por consola de comandos.

En este proyecto usaremos Git por consola de comandos como Sistema de Control de Versiones de todo el software generado, tanto de la Migración de Wave/SwellRT a Android como del desarrollo de la app y el Service REST del que hace uso. Incluso las distintas versiones del código látex de esta Memoria estará gestionado con Git.

4.1.12. GitHub

GitHub [46] es una plataforma web lanzada en 2008 que permite alojar de forma gratuita y en sitios llamados repositorios, proyectos software que utilicen Git como Sistema de Control de Versiones. Los repositorios gratuitos son públicos y accesibles por todo el mundo, de manera que cualquiera puede participar en la elaboración de código, aunque existe la opción de pagar por tener acceso a repositorios privados. GitHub proporciona una interfaz web que además de gestionar los repositorios permite alojar wikis, páginas web, gestión de tareas pendientes (issues) y control de acceso entre otras funcionalidades.

Para este proyecto utilizaremos GitHub como plataforma para alojar todo el código open-source desarrollado durante el proyecto: la Migración de SwellRT a Android (cuyo cliente web antiguo también está disponible en GitHub [38]), el desarrollo de la aplicación Android, el Service REST y la Memoria. Todo esto esta disponible en forma de repositorios bajo la organización llamada "Zorbel" en la siguiente URL:

<https://github.com/Zorbel>

4.2. Tecnologías de la Aplicación Android

4.2.1. Android Studio

Android Studio [47] es el Entorno de Desarrollo Integrado (IDE), basado en IntelliJ IDEA [48], oficial que Google proporciona para desarrollar aplicaciones para Android. Fue anunciado en 2013 y en diciembre de 2014 dejó su fase de beta y pasó a ser el IDE de referencia para el desarrollo en Android, dejando atrás el antiguo IDE de Eclipse junto al plug-in ADT. Android Studio integra en un solo lugar todas las herramientas necesarias para desarrollar en esta plataforma como un gestor de versiones de Android (SDK Manager), un gestor de emuladores virtuales (AVD Manager) o una herramienta de Debug (DDMS) entre otras. Permite asimismo trabajar tanto con código de aplicación como con la interfaz gráfica (XML), la cual podremos previsualizar en el propio IDE.

En este proyecto, y ya que desde principio de año es el IDE de referencia, utilizaremos Android Studio para el desarrollo de la aplicación Android que hará uso de las funcionalidades de Wave.

4.2.2. Android

Android [39] es el Sistema Operativo de código libre (bajo licencia Apache 2.0) para dispositivos móviles de Google basado en el kernel de Linux. Lanzado en 2007, actualmente su API alcanza ya la versión 21 [49], también denominada Android 5.0 "Lollipop". Para desarrollar en esta plataforma basta con descargarse su SDK [50], accesible entonces desde la consola de comandos, aunque siempre resulta más cómodo utilizar un Entorno de Desarrollo como Eclipse o Android Studio (Ver Secciones 4.1.5 y 4.2.1).

En este proyecto utilizaremos Android como plataforma móvil sobre la que llevar a cabo el desarrollo de la migración del cliente de Wave/SwellRT y para el desarrollo de una aplicación que haga uso de las funcionalidades y características de Wave. Concretamente utilizaremos el API 21 (Android 5.0 Lollipop) para tener acceso a la interfaz gráfica de diseño plano "Material Design" que incluye.

4.2.3. XML

XML [51] (eXtensible Markup Language) es un formato de definición, almacenamiento e intercambio de datos de forma estructurada. Fue definido en 1996 por el W3C y actualmente existe una versión 1.1 (2008). Se trata de un lenguaje de marcado que define dicho formato estructurado mediante marcas o "etiquetas" que aportan información acerca del texto que rodean. En el caso particular de Android, esta plataforma define la interfaz gráfica de sus pantallas mediante documentos XML llamados Layouts [52].

En este proyecto se utilizará XML para definir la interfaz gráfica de las pantallas que conforman la aplicación Android.

4.2.4. JSON

JSON [53] (JavaScript Object Notation) es un estándar para intercambio de datos de forma simple y ligera mediante pares clave-valor. Originalmente derivaba de un subconjunto de datos de JavaScript pero actualmente se encuentra definido en el RFC 7159 y el ECMA-404 y es independiente de lenguaje que se utilice para interpretar ("parsear") los datos que contiene. En JSON se puede estructurar estos datos de dos formas: en objetos (que contienen pares clave-valor) y en arrays (que contienen objetos).

En este proyecto utilizaremos JSON para intercambiar datos con el servidor de Wave/SwellRT y con el Service REST de la aplicación. En el caso de SwellRT no trabajaremos directamente a nivel de JSON ya que su API abstracta el formato de intercambio de datos. Sin embargo, en la aplicación sí que trabajaremos directamente con esta tecnología ya que el Service REST responde a las peticiones de la aplicación en forma de mensajes JSON que la propia aplicación debe tambien "parsear" para tratarlos.

4.2.5. SQL

SQL [54] (Structured Query Language) es un lenguaje diseñado específicamente para interactuar con Bases de Datos Relacionales pudiendo definir la estructura de los datos y manipularlos. Desarrollado en 1986, actualmente está estandarizado por la International Standards Organization (ISO) en el estándar ISO/IEC 9075 SQL [54]. Este lenguaje permite mediante la construcción de "consultas" crear tablas, acceder a sus datos y modificarlos entre otras cosas.

CAPÍTULO 4. TECNOLOGÍAS DEL PROYECTO

En este proyecto utilizaremos SQL en el Service REST para construir las consultas a Base de Datos necesarias para devolver al cliente de la aplicación Android los datos que solicite o pida cambiar.

4.2.6. MySQL

MySQL [55] es un Sistema Gestor de Base de Datos (SGBD) Relacionales que permite el almacenamiento, creación y modificación de dicho tipo de Bases de Datos. Desarrollado por Oracle, su versión actual (5.7.4) se distribuye tanto en forma open-source (bajo licencia GPL) o de uso comercial.

En este proyecto usaremos MySQL como SGBD para almacenar la Base de Datos de nuestra aplicación Android.

4.2.7. PhpMyAdmin

PhpMyAdmin [56] es una herramienta de código libre (bajo licencia pública GNU) escrita en PHP y que proporciona un sistema de administración de SGBD MySQL a través de una interfaz web. Fue desarrollado en 1998 y actualmente la versión 4.4.8 es desarrollada y mantenida por The PhpMyAdmin Project. Posee una interfaz sencilla que permite administrar la Base de Datos mediante las operaciones básicas de creación, modificación, eliminación de tablas, creación de consultas SQL y gestión de permisos y usuarios entre otros.

En este proyecto usaremos PhpMyAdmin como sistema de administración de la Base de Datos MySQL que contiene los datos de la aplicación Android.

4.2.8. PHP

PHP [57] es un lenguaje de propósito general del lado del servidor diseñado originalmente para crear aplicaciones web que generaran contenido dinámico. Diseñado en 1996 por Rasmus Lerdorf, su versión actual 5.6.7 es de código libre y se distribuye bajo licencia PHP. Tiene la ventaja de poder ser fácilmente incorporado dentro de los documentos HTML.

Para este proyecto utilizaremos el lenguaje PHP para programar el comportamiento del Service REST que hace de intermediario entre las peticiones de la aplicación Android y la Base de Datos MySQL.

4.2.9. Laravel 5

Laravel [58] es un framework que permite desarrollar aplicaciones y servicios web con PHP 5. Fue desarrollado por Taylor Otwell en 2011 y su versión actual 5.0 se distribuye en forma de código abierto (bajo licencia MIT) en su propio repositorio público en GitHub [59]. Su funcionalidad es extensible mediante módulos.

En este proyecto utilizaremos Laravel 5 para construir en PHP el Service REST que hará de intermediario entre las peticiones HTTP del cliente Android y la Base de Datos de la aplicación.

4.2.10. OpenShift

OpenShift [60] es una plataforma de computación en la nube que ofrece alojar servicios de forma gratuita en sus servidores mediante un modelo de arquitectura Software as a Service (SaaS). Disponible desde 2011, se distribuye bajo licencia Apache 2.0. Dispone también de planes de pago que aumentan las prestaciones del servidor.

En este proyecto usaremos OpenShift como servidor web para alojar los servicios de los que hace uso nuestra aplicación Android: el Service REST y la Base de Datos. Concretamente nuestro servidor esta accesible desde la siguiente URL:

<https://apptfg-servicerest.rhcloud.com/>

4.2.11. Sublime Text

Sublime Text [61] es un editor de texto gratuito multiplataforma muy versátil que proporciona características de atajos de teclado y resaltado de código para diferentes lenguajes de programación muy útiles cuando se desarrolla sin un IDE. Fue desarrollado en 2008 por Jon Skinner y actualmente se encuentra en su versión 2.0.2.

En este proyecto utilizaremos Sublime como editor de texto sobre todo para configurar y programar el Service REST en PHP, pues no vemos necesario utilizar un IDE específico para ello.

4.2.12. POP: Prototyping On Paper

POP [62] (Prototyping On Paper) es una aplicación con versiones web y para dispositivos móviles que permite elaborar en pocos pasos prototipos (mockups) interactivos basados en fotos de prototipos realizados en papel. De esta forma se puede elaborar un prototipo de interfaz gráfica de bajo coste con la que el usuario pueda interactuar.

En este proyecto se utilizará POP durante la fase de diseño de la aplicación Android para elaborar mockups sencillos e interactivos que podamos enseñar a la gente y que nos ayuden a refinar el diseño de la interfaz gráfica de la aplicación. Tiene además la ventaja de que se puede enseñar en la aplicación móvil de POP para imitar^{el} aspecto final que tendría la aplicación y que el usuario se haga una mejor idea de lo que pretendemos diseñar.

Capítulo 5

Metodología del Proyecto

A continuación se exponen las diferentes metodologías que hemos utilizado durante el desarrollo del *Trabajo de Fin de Grado*.

5.1. Uso de Software Libre

Para el desarrollo de todo el software que compone el proyecto, siempre hemos utilizado software libre. Tanto con las herramientas necesarias para el desarrollo como el resultado final de nuestra aplicación están desarrolladas con software libre. Dando libertad para que otros usuarios puedan visualizar el código desarrollado o utilizar las herramientas libremente. Y realizando aportaciones a toda la comunidad subiendo el código del proyecto a **GitHub** bajo una licencia **GNU GPLv3** [63].

A continuación se expone un breve resumen del software libre utilizado y las licencias que poseen:

Software	Licencia
Eclipse	Eclipse Public License
Android Studio	Apache License 2.0
Laravel	MIT License
Apache Wave	Apache License
phpMyAdmin	GNU GPLv2
MySQL	GNU GPL
PHP	PHP License
Android	Apache License 2.0 y GNU GPLv2
Java	GNU GPL
OpenShift	Apache License 2.0

Cuadro 5.1: Software libre utilizado durante el desarrollo.

5.2. Metodología de Migración de Wave a Android

5.2.1. Objetivo

El framework de SwellRT utiliza un servidor WIAB y el protocolo Wave, ambos desarrollados en Java. El **SDK de Android** [50] 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.

5.2.2. Plataforma: Entorno de Desarrollo, Construcción y Depuración

Existen dos entornos de desarrollo (IDE) recomendados por Google para desarrollar en Android: Eclipse [40] y Android Studio.[47] 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 [48] 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 [40] soporta el desarrollo con Android a través del plugin **ADT (Android Development Tools)** [64], que integra en un solo paquete

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

te todas las herramientas necesarias para desarrollar, construir y depurar el código de la aplicación fácilmente.

Android SDK [50]: 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 [65]
- **Android Virtual Device Manager (AVDM)** [66] herramienta para gestionar la creación, modificación, ejecución y eliminación de emuladores en Android. Un **emulador** [67] 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** [68] 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)** [69] 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 [70] (Ver figura 5.1) 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 .

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

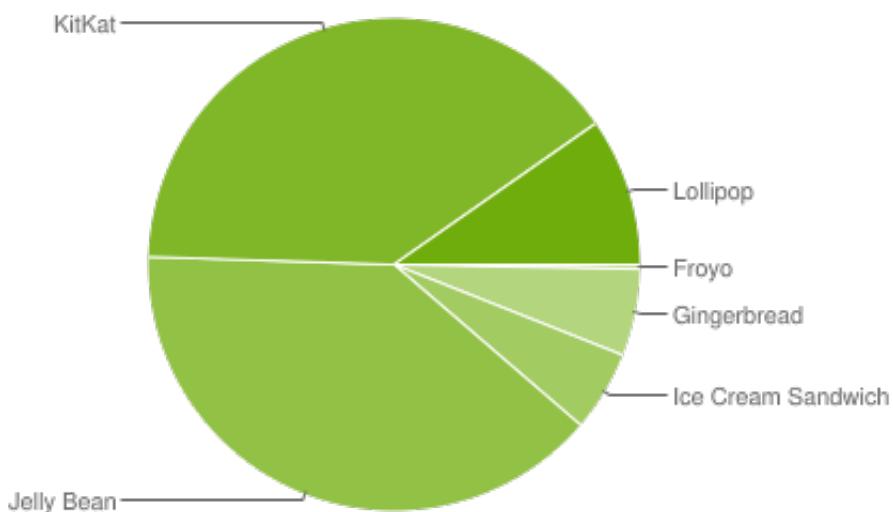


Figura 5.1: Distribución Actual de Versiones Android (Fuente: Google)

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** [71] por cuestiones de eficiencia. Para evitar esta limitación, durante el proceso de construcción el SDK de Android utiliza, entre otras, una herramienta llamada **ProGuard** [72]. 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 se puede compilar la aplicación en modo debug utilizando ProGuard.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Proceso de Construcción por Consola

Para construir la aplicación por consola de comandos, Android utiliza la herramienta Apache Ant [73] para automatizar el proceso de construcción [74]. 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 está 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 [75]:

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

```
$ ant clean debug
```

Esto nos generará una aplicación 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 5.2.2) 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, etcétera.) 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 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

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

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) [76] 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 cuál 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 5.2: Emulador Android API 19

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

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

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 .^Esperar 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.

5.2.3. 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 [38]. 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** [77], 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 [78] y WebSockets [44].

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Conexión 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** [79] para controlar los privilegios de cada aplicación. Estos permisos se declaran en el **Manifiesto** de la aplicación [80], 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"/>
```

También 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 válida. No obstante, esto tiene facil solución pues el **emulador de Android define unas direcciones IP de red especiales** [81] 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 5.2.2) estudiando el LogCat [82] (Ver Figura 5.3) para ver dónde se produce el error. Descubrimos que el problema estaba localizado en el método `login()` de la misma clase, que intentaba realizar una **petición 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 5. METODOLOGÍA DEL PROYECTO

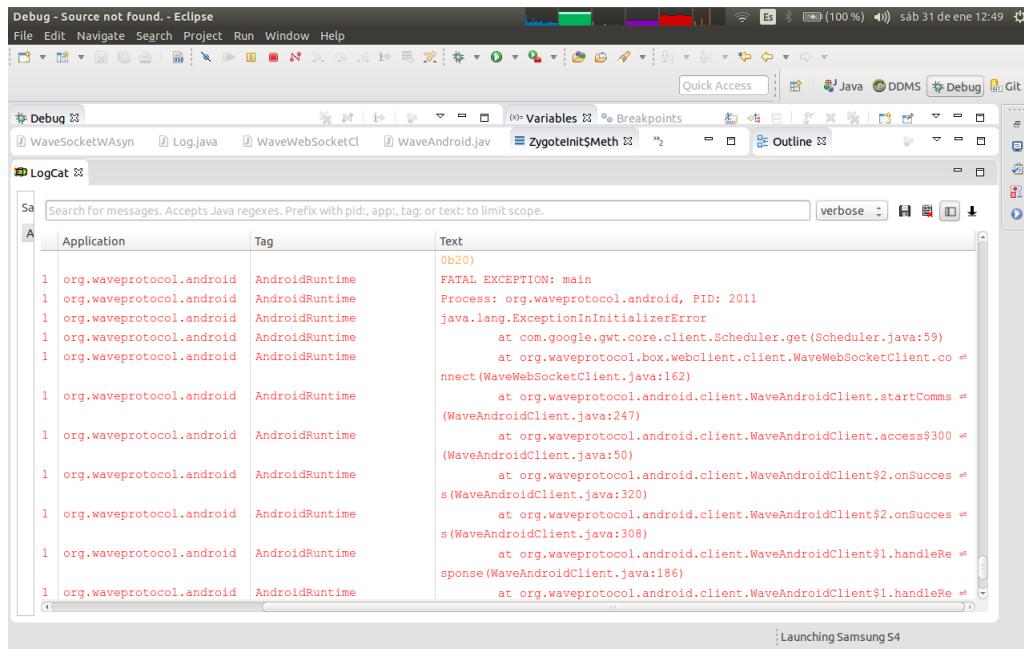


Figura 5.3: Ejemplo de Traza de Error en Logcat

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** [83], incluida en el SDK de Android desde sus primeras versiones. Sin embargo, Google recomienda [84] a partir del API 10 (Android 2.3 "Gingerbread") utilizar la **librería HttpURLConnection**[85], también incluida en el API. Por tanto esta última es la que elegimos para la migración.

De forma simplificada, éste 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 5. METODOLOGÍA DEL PROYECTO

```
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**) [86]:

- **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 [87]. La ventaja por tanto de usar otro hilo para esto es que la actividad principal no se bloquea.

Es importante también destacar que la arquitectura de SwellRT y de Wave está planteada de manera que utiliza llamadas asíncronas (callbacks) para

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

notificar al resto de la aplicación del resultado de los procesos de conexión al servidor, por lo que nuestro AsyncTask tendrá que usar el callback apropiado para notificar del éxito o fracaso de la conexión Http.

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
        return sessionId //String needed for the WebSocket
            connection and based on the cookie received from the
            server.
    }
    @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
        }
    }
}
```

Este proceso de conexión Http nos deberá devolver una Cookie que trataremos con el objetivo de generar un SessionId que será necesario para seguir con la conexión al servidor. **Llegados a este punto, tenemos un proceso de login Http que hace uso de la librería 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 aplicación aún no funciona correctamente, pues se cierra al intentar ejecutar el código que se encarga del siguiente paso de la conexión: conectarse por WebSocket.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Conexión WebSocket

Para realizar una conexión con el servidor Wave es necesaria una conexión mediante WebSockets[44], tecnología que permite conexiones bidireccionales y asíncronas entre el servidor y el cliente (recordemos que la conexión en el modelo cliente-servidor tradicional está definida como unidireccional de cliente a servidor), 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 asincronía es necesaria ya que para mantener la consistencia en tiempo real 4.1.3 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 dónde 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 [88], 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. Asimismo 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 WebSocketFactory.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

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

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.

La solución encontrada fue utilizar wAsync[89], 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) {
```

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

```
// catch possible exceptions
}
});
{
    // 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 **WaveSocketWAAsync.java**[90] 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 WAAsync 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 librerías oportunas (Ver Tabla de dependencias 5.2) y configurar el cliente segun lo descrito en dicha wiki:

```
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 password, ya que hasta ahora las habíamos especificado a mano en el propio código para realizar pruebas.

Conexión final 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

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

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.

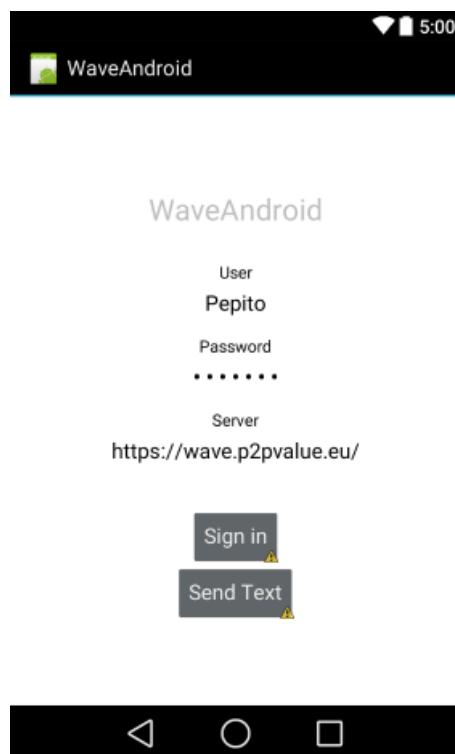


Figura 5.4: Pantalla de Login de WaveAndroid

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[91].

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

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, contraseña y comprobamos que hemos conseguido el objetivo de esta parte del proyecto: **nuestro cliente Android realiza el login contra el servidor WIAB correctamente.**

Organización del código: Servicio Android

Una vez conseguida la conexión al servidor desde Android, decidimos revisar el código para intentar optimizarlo y organizarlo de manera que aprovechara mejor las características de Android y la arquitectura de Wave. Además, el API que permite gestionar el modelo de datos de SwellRT (Ver Sección 4.1.4) está escrito en java, por lo que es plenamente funcional y compatible con el código de nuestra migración a Android. A continuación hablaremos de los motivos de dicha reorganización.

Como ya se ha comentado anteriormente, el proceso de login se debe hacer en un hilo de ejecución separado del hilo principal o UI Thread, ya que Android no recomienda[86] 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 [92] 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 5.2.3 y 5.2.3) 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 una vez que termina su ejecución puede notificar el resultado de la conexión, pero no queda a la espera de otros posibles eventos en el socket (como la recepción de mensajes o la desconexión). Es decir: **aunque se definan callbacks para esperar los eventos del servidor, el socket no podrá notificarlo porque no existe un hilo que quede a la espera de estos eventos.**

Por otro lado, 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

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

en la implementación inicial 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[93].

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. **Por tanto, podremos acceder a la conexión al servidor desde cualquier punto de la aplicación, solo es necesario que la Actividad en ejecución se suscriba al Servicio para hacerlo.**

Pero un Servicio se ejecuta dentro del mismo proceso que el UI Thread, por lo que aun así tendremos que utilizar métodos para ejecutar el código que nos interese en otros hilos de ejecución dentro del propio Servicio. De esta manera **se dividió la reorganización en dos: la conexión Http y la conexión WebSocket.**

Para la conexión HTTP, se decidió utilizar un AsyncTask llamado LoginTask muy similar al ya explicado anteriormente (ver Sección 5.2.3) pero esta vez definido dentro del propio Servicio y con un callback que notificaba a la aplicación si la conexión se realizaba correctamente. Además, se puso el código de la conexión en una clase aparte llamada WaveHttpLogin.java para tenerlo más organizado. El siguiente es un esquema en forma de Diagrama de Secuencia de la conexión Http:

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

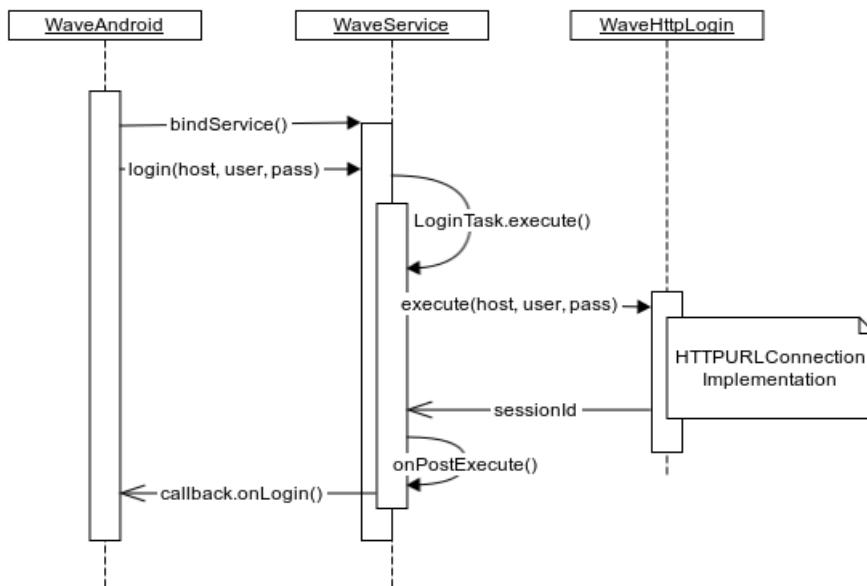


Figura 5.5: Proceso de conexión Http con Servicio

Como vemos, cuando se realiza esta conexión Http la Actividad inicial (WaveAndroid.java) es informada de ello mediante su callback onLogin() para poder notificar al usuario e iniciar la conexión por WebSocket.

Para dicha conexión se debía buscar una solución que permitiera al WebSocket responder a eventos del servidor (recordemos el funcionamiento de WAsync descrito en la Sección 5.2.3) para, mediante callbacks, informar a la aplicación de dichos eventos. **La solución final encontrada fue combinar el uso de un Thread y un Handler**. El Thread se encarga de crear el socket y configurar su respuesta a eventos en forma de mensajes, que enviará entonces al Handler, encargado de quedar a la espera de recibir estos mensajes de forma asíncrona y llamar al callback adecuado. Esta implementación se puede ver concretamente dentro de la clase WaveSocketWAsync.java, donde se define el Thread llamado WebSocketRunnable y el Handler UIHandler. El siguiente es un esquema en forma de Diagrama de Secuencia de la conexión inicial con WebSocket al servidor Wave:

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

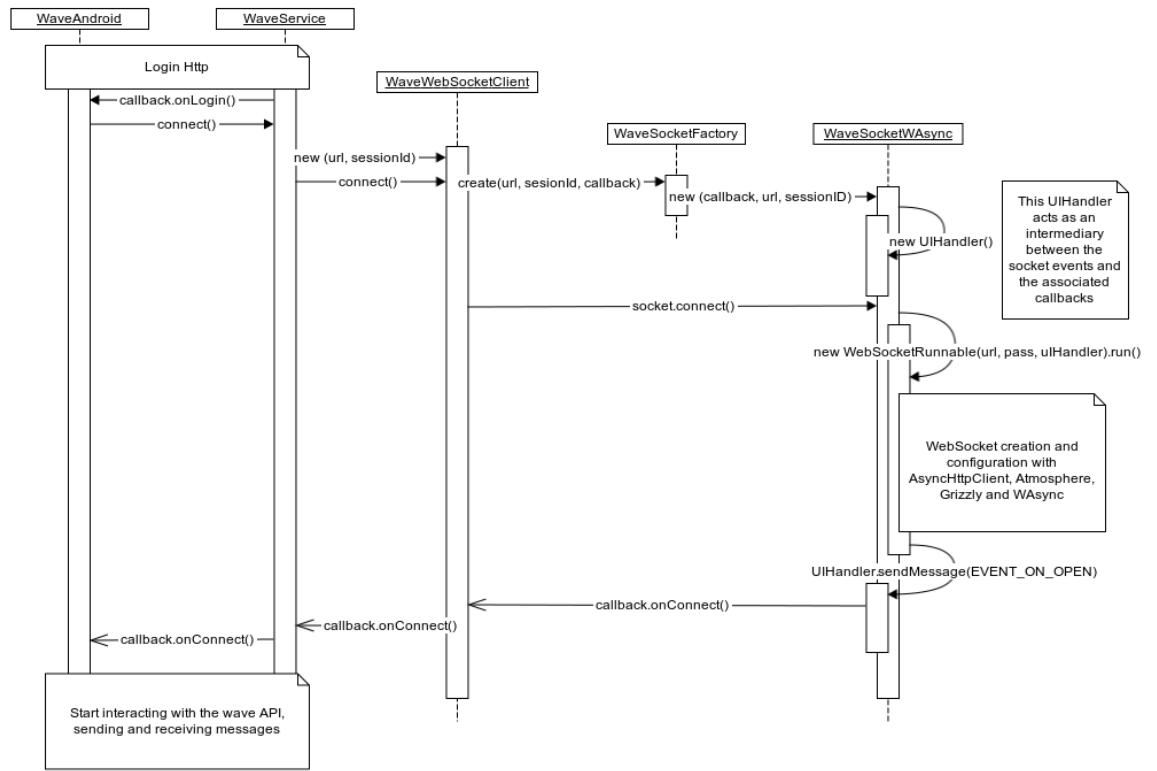


Figura 5.6: Proceso de conexión WebSocket con Servicio

Como vemos, cuando se realiza esta conexión al Servidor Wave la Actividad es notificada de ello mediante la propagación de callbacks `onConnect()` que informan del éxito de la conexión. A partir de este momento el socket informará de la misma forma (mediante el envío de mensajes al UIHandler) de otros eventos que se produzcan, ya sea la recepción de datos o la desconexión por ejemplo. De la misma forma la aplicación podrá enviar datos al servidor, ya que el socket se mantiene en el Servicio.

5.2.4. 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:

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Nombre	Versión	Descripción
WAsync [89]	1.4.3	WebSockets/HTTP Client Library for Asynchronous Communication
AsyncHttpClient [94]	1.8.14	Library that allows Java applications to easily execute HTTP requests and asynchronously process the HTTP responses.
Grizzly-Framework [95]	2.3.18	Core framework for Grizzly applications that provides TCP/UDP transports, memory management services/buffers, NIO event loop/filter chains/filters.
Grizzly-Http [95]		HTTP framework that contains the base logic for dealing with HTTP messages on both the server and client sides.
Grizzly-WebSockets [95]		WebSockets custom API for building Websocket applications on both the server and client sides.
slf4j-Android [91]	1.6.1	Simple Logging Facade for Java: logging framework for Android

Cuadro 5.2: Dependencias de SwellRT-Android

5.2.5. 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[90].** El siguiente es un esquema en forma de Diagrama de Clases del resultado final:

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

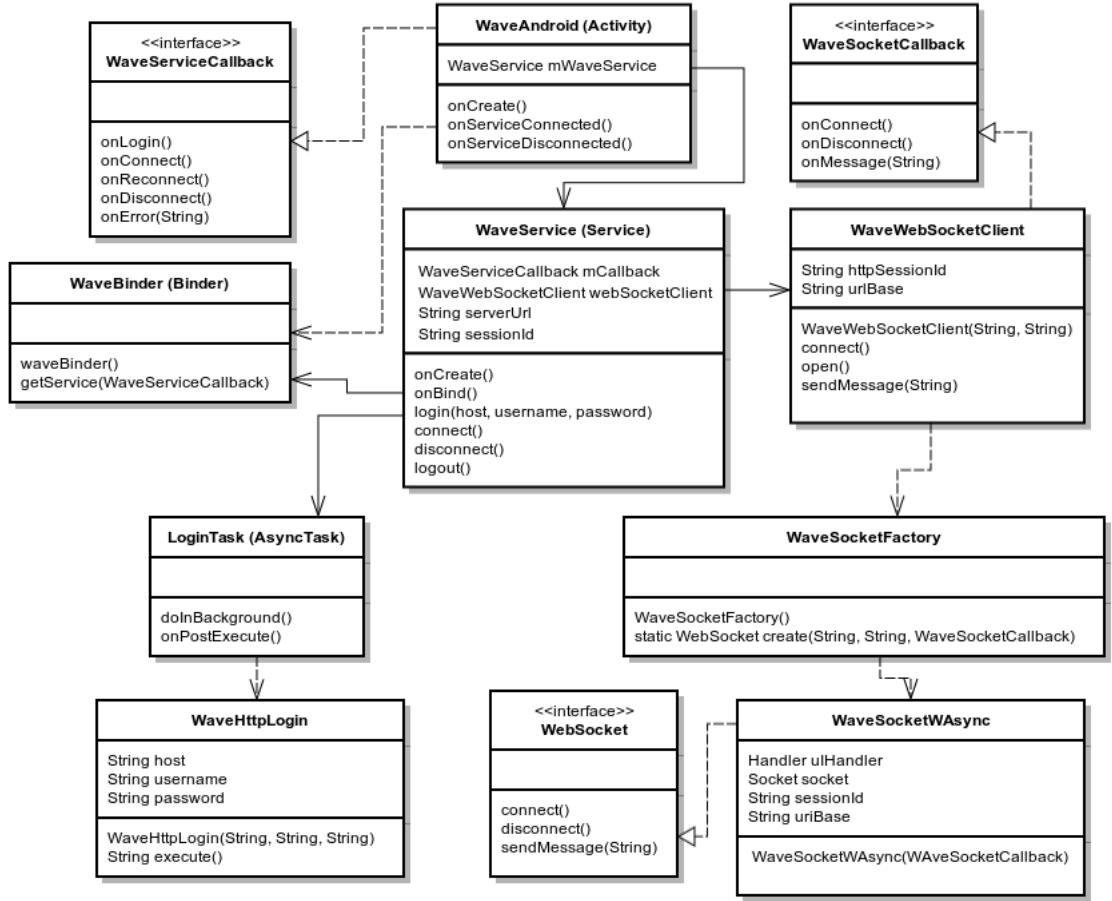


Figura 5.7: Esquema de Clases de SwellRT-Android con Servicio

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 ello, se encuentra en el GitHub de SwellRT[96].

5.3. Diseño de la Aplicación: Diseño Guiado Por Objetivos

Para elaborar el diseño de la aplicación, nos hemos basado en la metodología del Diseño Guiado por Objetivos (**DGO** o *Goal-Directed Design*), que implementa el proceso de la Ingeniería de la Usabilidad propuesto por Alan Cooper [97]. Este proceso constará de las siguientes fases:

1. Investigación

Esta fase consistirá en la realización de estudios para obtener datos cualitativos sobre los usuarios y/o reales de la aplicación y cuáles son sus necesidades. Se realizarán tareas para comprender a los usuarios, saber sus inquietudes y lograr empatía. A lo largo de esta fase se irán identificando patrones de comportamiento que sugerirán los objetivos y motivaciones del usuario. Por último se realizarán estudios de mercado, revisiones y auditorías que ayudarán al diseñador a comprender el dominio, el modelo y las restricciones técnicas que el sistema debe cumplir.

2. Modelado

A lo largo de esta fase se utilizarán los datos provenientes de la fase previa para crear los modelos del dominio y los usuarios. En esta parte se crearán las *personas*, aquetipos de usuarios que contienen información sobre objetivos, motivaciones y comportamientos de los usuarios con el sistema. El resultado final de esta fase serán los tipos de *persona* que representarán a los usuarios del sistema, y que más adelante serán utilizados para proporcionar algún tipo de *feedback*.

3. Definición de Requisitos

Durante esta fase se utilizarán las personas y los datos de la fase anterior, para crear los *escenarios* de contexto e identificar los requisitos o necesidades del usuario. Estos requisitos serán definidos en tres componentes: objeto, acciones y contexto. También se definirán requisitos relacionados con el negocio, la aplicación, requisitos técnicos, etcétera.

4. Definición del framework de Diseño

En esta fase se creará el concepto general del sistema, definiendo los comportamientos y diseño visual. Identificaremos el **framework de**

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

interacción, como un concepto de diseño estable que define la estructura del sistema a partir de patrones y principios de diseño. Por último, se definirá el *framework visual*, como el aspecto visual de la aplicación (diseño, tipografía, colores, iconos, etcétera).

Queremos insistir eso sí en que únicamente nos hemos basado en esta metodología para seguir el proceso del diseño de la aplicación. No seguiremos todas las fases de esta metodología al pie de la letra ya que el alcance en tiempo de este proyecto escapa a una metodología tan formal y laboriosa (que implica gran cantidad de pruebas y procesos) como esta.

5.3.1. Investigación

Intención Inicial: Prototipo básico

Vivimos en una época donde la política parece estar de moda. Esto puede ser debido al cabreo general que muestra la ciudadanía frente a gobiernos conservadores, situaciones de austeridad provocada por la crisis económica, el nacimiento de nuevas fuerzas políticas, . . . , pero sobre todo las numerosas citas electorales a las que seremos citados en 2015. Por tanto, el proyecto podría ponerse a prueba en un escenario real durante la campaña de las elecciones generales previsiblemente convocadas durante el último trimestre del 2015.

Programas electorales

La intención fundamental de la aplicación es llevar los programas electorales a los bolsillos de los ciudadanos y generar interés en participar más activamente en la política, ya sea emitiendo opiniones sobre dichos programas o elaborando nuevas propuestas. Vivimos en una sociedad digital, donde cada vez son más las personas que utilizan sus smartphones para realizar todo tipo de tareas en su vida cotidiana.

En los últimos años las diferentes formaciones políticas han subido sus programas electorales a un documento en formato PDF que suele estar disponible para su descarga en su página web. Este documento tiene generalmente una gran extensión (los hay de 200 páginas), lo cual no hace sino dificultar que las personas se animen a leerlo. Por ello pensamos que una aplicación que pudiera visualizar las principales secciones de los programas políticos podría ser especialmente útil para acercar los programas a los electores.

Además también intentaríamos darle una estructura a estos programas, de

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

manera que el usuario pudiera navegar por ellos a nivel de Sección, a diferencia del método actual de leer un "macro-documento.^{en} PDF. Así, la gente podría opinar sobre los programas políticos a nivel de sección mediante acciones familiares para ellos: "Me gusta", "No me gusta" realizar Comentarios". Añadimos también una acción de "No lo entiendo" que pensamos que sería útil para indicar cuándo la redacción de la sección era de significado difuso.

En definitiva, queríamos crear un espacio donde poder informarse sobre las distintas ofertas electorales y poder debatir sobre las propuestas que propone cada formación política, todo ello en forma una aplicación que podremos consultar en cualquier momento.

Propuestas y Wave

Por otro lado, y en línea con los últimos movimientos políticos ciudadanos, pretendíamos crear también un portal de propuestas ciudadanas en el móvil. Los usuarios podrían visualizar las propuestas de otros usuarios y tener la posibilidad de crear nuevas propuestas. Además, como queríamos aprovechar las características de la migración de Wave previamente desarrollada, pensamos en la posibilidad de elaborar estas propuestas de forma colaborativa y en tiempo real entre muchos usuarios.

Actualmente existen multitud de portales (en su gran mayoría web) donde la ciudadanía puede expresar su opinión, pero creemos que la integración de una aplicación donde puedan situarse las opiniones de los partidos políticos (en forma de sus programas) y la actividad ciudadana (en forma de propuestas), genera una nueva manera de tratar la política en los medios sociales.

También pensamos en la posibilidad de categorizar el contenido de la aplicación (Programas y Propuestas) por temas, para proporcionar filtros a la hora de navegar por dicho contenido. Sin embargo no teníamos muy clara la elección de temas, así como si debíamos darle al usuario la posibilidad de crear nuevos temas o dar nosotros unos temas preestablecidos.

Una vez pensada la intención y los principales objetivos de la aplicación, procedimos a realizar unos primeros prototipos en papel de nuestra idea y a implementar un prototipo básico en el móvil (Ver Sección ??) que mostraba programas políticos estructurados y permitía navegar a nivel de Sección por ellos para leer y emitir opiniones (likes, dislikes, comentarios, etcétera.).

Hipótesis de personas

Para identificar a los usuarios objetivo, tenemos que encontrar a miembros representativos de esos usuarios y animarlos a que participen en nuestra

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

investigación. Para ello reuniremos una serie de características básicas que el usuario deberá cumplir para poder aportarnos los objetivos funcionales de la aplicación. Estas son algunas de las características deseadas:

- **Edad:** Entre 16 y 65 años.
- **Sexo:** Indiferente.
- **Profesión:** Indiferente.
- **Aptitudes deseables:** Activismo social, conciencia política, trabajo colaborativo,
- **Habilidades técnicas:** Usuario con experiencia media en uso de aplicaciones móviles.

Simplificando el tipo de persona que queremos encontrar, lo dividiremos en dos tipos de persona. Por un lado buscaremos al activista social, activo en movimientos sociales, participación en portales con carácter social, etcétera. Mientras que por otro lado buscaremos una persona que muestre cierto interés en el mundo de la política, pero que no participe en movimientos sociales. Finalmente pudimos contactar con dos miembros de Labodemo [98], una organización que se dedica al desarrollo de nuevas formas de participación ciudadana. Este perfil cubría en caso del activista social, mientras que para el tipo de persona entendido de la política pero no tan involucrado, escogimos a Javier de la Cueva [99].

El siguiente paso sería estudiar la viabilidad de esta aplicación entrevistando a las personas seleccionadas para realizar una investigación sobre sus necesidades prácticas. Por otra parte también sería útil enseñarles los prototipos básicos que manejábamos para verificarlos. Entendimos que el tipo de persona con el que nos entrevistáramos debía de estar relacionado de alguna manera con el mundo de la política, pues nada mejor que hablar con gente ya interesada en dichos temas para orientarnos por el buen camino.

A continuación detallamos las entrevistas que realizamos en la investigación:

Entrevista con Labodemo

Tuvimos la oportunidad de mantener una conversación con dos miembros de Labodemo [98], en la que aprovechamos para mostrarles un prototipo de la aplicación que estábamos desarrollando. Ambos tenían experiencia en el

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

desarrollo de plataformas de participación ciudadana en Internet: fueron los responsables del desarrollo de los portales de participación del partido político Podemos y la candidatura ciudadana de unidad popular Ahora Madrid.

En ese momento nuestro prototipo móvil se limitaba únicamente a mostrar las diferentes secciones de cada programa, lo cual les pareció útil, aunque no lo suficiente como para atraer a una cantidad considerable de usuarios. Conforme a su línea de trabajo habitual, eran más partidarios de dar a los usuarios la posibilidad de realizar Propuestas además de ver Programas políticos. Les comentamos entonces que antes de hablar con ellos ya habíamos planteado desarrollar Propuestas colaborativas en tiempo real aprovechando la tecnología de Wave. Pero ellos no eran partidarios de esta opción, ya que según ellos acabaría siendo caótico tener tanta gente editando la misma propuesta de cara a generar contenido útil.

Dándole una vuelta a la categorización del contenido, nos sugirieron que para atraer a usuarios, debíamos considerar la posibilidad de integrar en la aplicación a colectivos sociales que generaran y categorizaran dicho contenido. No eran partidarios de que dieramos nosotros ciertas temáticas preestablecidas, pues debido a la variedad de redacción en los distintos programas sería difícil identificar temáticas que incluyeran a todos los programas y probablemente acabariamos excluyendo temas. Así, serían los colectivos los que se encargaran de "tematizar"^{el} contenido de la aplicación. Por ejemplo: un grupo de animalistas podría tener un espacio en la aplicación donde poder crear sus propias propuestas, e incluso hacer comparativas de lo que proponen los diferentes programas sobre los animales.

De esta manera, y en relación a la aproximación a los foros tradicionales, se planteó la idea de crear "Hilos": elementos "temáticos" que agruparan en un solo sitio Secciones y Propuestas que hablaran sobre un determinado tema. Estos hilos serían también generados por dichos colectivos.

Esto sería útil también para usuarios que buscaran información sobre un determinado tema. Por ejemplo: un usuario poco activo, que resulta ser profesor, podría buscar un colectivo de profesores y ver las Propuestas que se llevan a cabo o visualizar una comparativa respecto las medidas de educación de los diferentes programas políticos.

Nos insistieron mucho en el tema de las comparativas. Sería de gran utilidad que la aplicación tuviera una parte de comparativas en la que los usuarios pudieran comparar los programas políticos en vez de leerlos sección por sección. Resultaría de gran interés a un autónomo visualizar las medidas que proponen los diferentes partidos políticos para los autónomos. Pero estas comparativas no podría realizarlas cualquiera, por lo que deberían realizar-

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

las periodistas o expertos que hubieran realizado algún tipo de comparativa similar anteriormente. Nos sugirieron contactar con periodistas o colectivos que hubieran publicado algún tipo de comparativa en cuanto a programas o medidas, para obtener algún tipo de ayuda o consejo a seguir.

Conclusión

Puntos positivos:

- Nueva forma de participación ciudadana de cara a las elecciones.
- Métodos alternativos para discutir propuestas, partidos, programas, etcétera.
- Ligar propuestas a programas concretos.

Puntos negativos:

- Visualizar un programa electoral en el móvil puede no resultar demasiado interés para los usuarios.
- Desarrollar propuestas colaborativas en tiempo real no maniene una estabilidad en la aplicación.
- La aplicación debería desarrollarse en otras plataformas móviles y de escritorio.

Puntos a tener en cuenta:

- Dejar libertad a usuarios y colectivos creando sus propias categorías o hilos. Espacios donde puedan elaborar sus intereses y generar contenido para la aplicación.
- Desarrollar comparativas por secciones, temas o partidos. De tal forma que un usuario pueda visualizar las diferencias de aquellos temas que le preocupan.
- Contactar con colectivos, asociaciones y/o periodistas que anteriormente hayan elaborado comparativas entre programas políticos en puntos concretos

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Entrevista con Javier de la Cueva

La entrevista con Javier de la Cueva resultó bastante productiva. Ya le conocíamos de algunas conferencias que impartió en la facultad. Javier es abogado y doctorado en Filosofía, estando especializado en temas relacionados con tecnología, Internet y propiedad intelectual. Además, en sus últimas conferencias Javier habla sobre acciones micropolíticas [100]. Estas acciones definen la capacidad que tienen los ciudadanos para realizar aportaciones a la sociedad, el estado o el gobierno que favorezcan la participación ciudadana en una democracia participativa.

Representar los programas electorales en una aplicación móvil le pareció algo interesante y necesario para la sociedad actual. Si bien casi nadie hace el esfuerzo de visualizar un programa electoral en PDF, utilizar una herramienta que facilita el acceso al programa por secciones podría ser una nueva forma de incentivar su lectura y ayudar a fomentar la participación ciudadana en política.

Además nos sugirió la posibilidad de desarrollar una Hemeroteca de programas electorales. De esta forma cualquiera podría consultar los programas de los anteriores gobiernos y comprobar si se cumplieron los objetivos del programa, así como comparar programas de distintos años entre sí,

Pero si en algo nos insistió Javier, fue en la importancia de categorizar el contenido de la aplicación. Un usuario que no tenga conocimientos sobre diversos temas, se encontraría más cómodo si pudiera visualizar las diferentes partes de un programa o las propuestas ciudadanas por categorías o temas generales. Ya que dejar libertad a los usuarios para crear categorías personalizadas podría ser algo negativo para usuarios inexpertos o con pocos conocimientos sobre temas específicos.

Por último, centrándonos en las Propuestas ciudadanas, surgió la idea de elaborar propuestas que tuvieran una especificación concreta. Es decir, a parte de tener una idea de propuesta y redactarla, esta propuesta debería ir acompañada de los recursos que serían necesarios y sobre todo cómo se llevaría a cabo de una forma aproximada. También resultaría interesante definir un pequeño presupuesto de lo que conllevaría realizar la propuesta o cómo se podría financiar. Así evitaríamos una elaboración de propuestas más real, evitando un listado de propuestas infinito sin planterase cómo se llevarían a cabo o cómo se financiarían.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Conclusión

Puntos positivos:

- Llevar los programas políticos de una forma más atractiva a la ciudadanía es algo esencial en la actualidad.
- Categorizar las secciones de los programas políticos para que se puedan explorar por temas.
- Incluir los recursos necesarios que hacen falta para llevar una propuesta a cabo.

Puntos negativos:

- Tratar de convertir la aplicación en un *foro* inconscientemente.
- Dejar libertad a la hora de crear categorías específicas o hilos puede generar confusión entre los usuarios.

Puntos a tener en cuenta:

- Desarrollar categorías semanales en función de las novedades o actualidad política.
- Añadir la posibilidad de solicitar la ayuda de expertos sobre un tema para elaborar una propuesta.
- Crear una emeroteca de programas políticos para realizar análisis sobre el cumplimiento de los programas en legislaturas pasadas.
- Informar sobre la legislación actual cuando visualicemos una propuesta o sección de un programa que quiera mejorar o cambiar la legislación actual.

5.3.2. Modelado de Personas

Las *personas* son una herramienta de diseño y ayuda al diseño de la aplicación. El objetivo es centrar el diseño en este tipo de persona, para lograr los objetivos a la hora de utilizar un sistema. La *persona* será nuestro modelo, una descripción detallada de un individuo imaginario que representa a un

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

grupo de usuarios a los que va destinada la aplicación. Es una representación ficticia pero desarrollada con gran detalle, que será fruto de los datos recogidos en la investigación previa de la fase anterior.

En esta fase definiremos el tipo de persona que interactuará con nuestra aplicación. Para ello hemos identificado dos tipos de personas primarias; un **activista social** y un **ciudadano** de a pie que forme parte del electorado.

- Activista social, 16 años en adelante

Actividad:

- Estudia, trabaja o realiza otras actividades de voluntariado.
- Frecuenta asambleas, participa en diferentes movimientos sociales y está al día de la actualidad política.
- Utiliza redes sociales para comunicarse con otros colectivos, acudir a asambleas, promover ideas u otras actividades relacionadas con la política y el activismo social.

Otros:

- Desconoce las ideas que proponen algunos partidos
- Le gusta aportar nuevas soluciones a la sociedad.

- Ciudadano de a pie, 18 años en adelante

Actividad:

- Estudia, trabaja o realiza otras actividades de voluntariado.
- Es distante al mundo de la política, concibe ciertos temas pero no los conoce en profundidad.
- Visita diferentes medios de comunicación para enterarse de la actualidad.
- Utiliza redes sociales para compartir contenidos con sus amigos o establecer nuevas amistades.

Otros:

- Desconoce por completo los programas electorales.
- Tiene cierta indecisión a la hora de acudir a las urnas, no sabe qué propone cada partido.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

<p>Nombre: Lucía Edad: 21 años.</p> <p>Ocupación y actividades:</p> <ul style="list-style-type: none">• Estudiante de <i>Administración y Dirección de Empresas</i> en la Universidad Rey Juan Carlos.• Le preocupa la situación actual del estado del bienestar y la gestión de los recursos públicos. <p>Cita: "Los programas políticos deberían de tener mayor peso en la sociedad".</p> <p>Actividad política:</p> <p>Lucía es un estudiante universitario que será citado a votar en las elecciones generales que serán celebradas en dos semanas. Para estar al día de la actualidad política sigue las redes sociales, debates televisivos y otros medios.</p> <p>Hay diversas opciones políticas para Lucía, pero ella aún no ha decidido su voto. Quiere explorar los proyectos de futuro que tienen las formaciones políticas para el país.</p>	
---	--

Figura 5.8: Estudiante universitario

5.3.3. Definición de personas

En las figuras 5.8 y 5.9 se exponen una serie de personas ficticias que podrían representar en la vida real los tipos de *persona* a la que va dirigida la aplicación.

5.3.4. Definición de Escenarios y Requisitos

En esta sección se definirán los posibles escenarios que puedan surgir en la aplicación. La idea es situarnos en un escenario real que pudiera ocurrir en cualquier momento a lo largo del día, para detallar la solución al problema. En cada uno se especificarán los requisitos necesarios para solventar el problema o los pasos a seguir para lograr el objetivo. Diferenciaremos los términos de **acción**, como la actividad inmediata que requiere la solución. El **objeto**, como el sujeto principal del escenario. Y por último definimos **contexto** reflejando el objetivo final del requisito.

Escenario I

Se acercan las elecciones municipales y Juan aún no ha decidido a qué partido va dar su voto. No conoce las propuestas que ofertan los partidos a la ciudadanía y tampoco se fía mucho de lo que dicen los medios de comunicación.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Nombre: Carlos
Edad: 34 años.

Ocupación y actividades:

- Trabaja como profesor de educación primaria en un colegio público de Barcelona.
- Miembro de la *Plataforma de Afectados por la Hipoteca*.
- Participa en diversos movimientos sociales.

Cita: "Debemos de incentivar la participación de la gente común en la política".

Actividad política:

Carlos es un activista social que nunca se pierde la oportunidad de participar en todos los movimientos sociales que lo rodea. A menudo portales de participación ciudadana para organizarse con los miembros de su colectivo.

Para cambiar la sociedad Carlos piensa que las personas deberían involucrarse más en la política que desarrollen nuevas ideas y propuestas para el estado. Algunas personas le preguntan de qué forma pueden ayudar a generar nuevas propuestas e ideas.



Figura 5.9: Activista político

Juan coge su móvil y visualiza los diferentes programas electorales por categoría, seleccionando la categoría de educación que es la que más le afecta a él personalmente. La aplicación le muestra un listado de las secciones donde los diferentes partidos hablan de las medidas que van a tomar en torno a la educación.

Requisitos:

1. Visualizar (acción) las diferentes categorías (objeto), para ver las secciones de los programas de una determinada categoría (contexto).
2. Mostrar (acción) un listado de todas las secciones de los programas de los partidos políticos (objeto) en función de la categoría seleccionada por el usuario (contexto).

Escenario II

Pablo es un empleado sanitario de Hospital Clínico de Madrid preocupado por la gestión de los hospitales públicos. Parece que la situación no está muy controlada, por lo que quisiera saber qué propuestas o alternativas propone la ciudadanía para mejorar la situación actual.

A través de su móvil puede explorar las diferentes propuestas por categorías.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Eligiendo la categoría de sanidad, le aparece un listado de las últimas propuestas desarrolladas por la ciudadanía.

Requisitos:

1. Visualizar (acción) el listado de propuestas (objeto), clasificados por la categoría seleccionada por el usuario (contexto).
2. Mostrar (acción) la propuesta (objeto), seleccionada por el usuario para que pueda puntuarla y/o comentarla (contexto).

Escenario III

Lara es una profesora de un colegio de la Comunidad de Madrid. Se acercan las vacaciones de verano y muchos niños se quedarán sin acceso a comedor. Lara está planteándose cómo elaborar una propuesta ciudadana para abrir los colegios en horario no lectivo y que todos los niños tengan derecho a comedor en los meses de verano. Lara no es una experta en gestión pública ni sabe cómo llevar a cabo la propuesta.

Para ello, Laura comienza a desarrollar una propuesta ciudadana en la aplicación, explicando la base de la propuesta. Al no saber cómo financiarlo, deja la propuesta abierta para que la comunidad la pueda ayudar a desarrollarla.

Requisitos:

1. Agregar (acción) una nueva propuesta (objeto) rellenando los principales campos del formulario (contexto).
2. Publicar (acción) la propuesta (objeto) como una propuesta colaborativa para editar (contexto)
3. Mostrar (acción) la propuesta (contexto) en la lista de propuestas colaborativas para que puedan colaborar otros usuarios (contexto).

Escenario IV

Fran es un periodista de un periódico digital. Se acercan las elecciones y está redactando un pequeño artículo acerca de los programas de algunos partidos políticos. Necesita acceder a los programas completos de los partidos y visualizar las secciones que le resulten de interés para comentarlas en su artículo.

Accediendo a la aplicación, Fran puede seleccionar los programas de todos los partidos que se presentan a las elecciones. Listando el índice del programa y accediendo a sus secciones.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Requisitos:

1. Visualizar (acción) todos los partidos (objeto) que se presentan a las elecciones (contexto).
2. Seleccionar (acción) un partido político (objeto) para visualizar el programa electoral (contexto).
3. Listar (acción) el índice (objeto) de secciones del programa seleccionado (contexto).
4. Visualizar (acción) la sección (objeto) del programa seleccionado por el usuario (contexto).

5.3.5. Framework de diseño

En esta sección detallaremos todos los aspectos relacionados con el desarrollo del aspecto visual y la interacción con la aplicación. Se utilizarán los escenarios y requisitos definidos en la fase anterior para crear los bocetos y prototipos interactivos. Detallaremos los prototipos desarrollados en papel, algunos prototipos intermedios y el prototipo final de la aplicación presentado.

Framework de interacción

Para definir el Framework de interacción realizaremos un proceso de seis etapas:

Factor de forma, postura y métodos de entrada

La aplicación será visualizada en un smartphone bajo el sistema operativo Android (desde la API 15), con tamaños de pantalla entre 3,5 y 6 pulgadas. Que pueda visualizarse en interiores y exteriores.

La **postura** será **temporal**. El usuario puede utilizarlo en periodos de tiempo muy breves como puede ser la consulta de alguna sección, propuesta o dar su valoración. Elaborando una propuesta o participando en el desarrollo de una, el usuario necesitará algo más de tiempo, pero seguirá utilizando funciones básicas. Propias de una postura temporal.

Elementos y datos funcionales

Elementos de datos:

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

- Programa político
 - Atributos: partido, programa, secciones, índice, temas.
 - Relaciones: partido-programa, índice-sección.
- Temas
 - Atributos: categoría, programa, sección, partido, propuesta.
 - Relaciones: categoría-programa, categoría-propuesta, partido-sección.
- Propuestas Ciudadanas
 - Atributos: propuesta, categoría, usuario.
 - Relaciones: propuesta-usuario, propuesta-categoría.
- Propuestas Colaborativas
 - Atributos: propuesta, categoría, usuario, edición colaborativa.
 - Relaciones: usuario-propuesta, propuesta-categoría.

Elementos funcionales:

- Visualizar los partidos políticos que se presentan a las elecciones.
- Mostrar el índice de cada programa político.
- Mostrar, valorar y debatir las secciones de los programas políticos.
- Visualizar categorías por temas, distinguiendo entre propuestas y secciones de programas.
- Mostrar, valorar y debatir las propuestas de los usuarios.
- Crear una nueva propuesta en una categoría.
- Mostrar las propuestas colaborativas y participar en su desarrollo.
- Crear una nueva propuesta incompleta para desarrollarla de forma colaborativa.

Grupos funcionales y jerarquías

Pantalla principal de la aplicación. Grupos de elementos funcionales que contiene:

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

- Lectura y valoración de Programas políticos.
 - Visualizar programas políticos.
 - Ver secciones de programas.
 - Visualización, valoración y debate de secciones de programas políticos.
- Clasificación de secciones de programas y propuestas ciudadanas por categorías.
 - Ver secciones de programas políticos por tema.
 - Ver propuestas ciudadanas por tema.
- Lectura, valoración y creación de Propuestas Ciudadanas.
 - Ver propuestas ciudadanas.
 - Valorar y debatir propuestas ciudadanas.
 - Creación de propuesta ciudadana.
- Lectura, colaboración y creación de Propuestas Colaborativas.
 - Visualizar las propuestas ciudadanas en desarrollo.
 - Crear una propuesta colaborativa.
 - Colaborar en el desarrollo de una propuesta.

Boceto del framework de interacción

El proceso de desarrollo de bocetos se ha realizado en espiral, de tal forma que una vez que identificábamos una determinada funcionalidad, la representábamos en un mockup a papel para debatirlo y discutirlo en las entrevistas. A continuación trasladábamos el mockup a un prototipo de alto nivel desarrollado en Android. Para las siguientes implementaciones que añadían nueva funcionalidad el proceso era el mismo:

1. Desarrollo de un boceto a papel identificando los elementos funcionales.
2. Definir la interacción a través de la herramienta POP [62].
3. Implementar un prototipo de mayor fidelidad en Android.

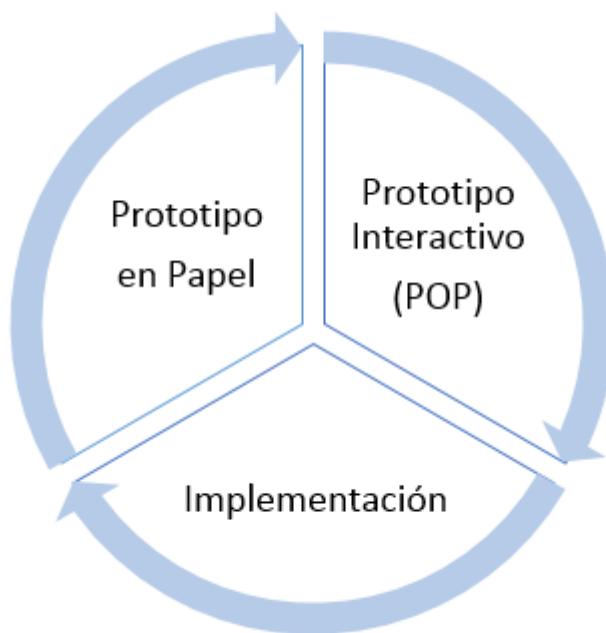


Figura 5.10: Modelo de desarrollo de los prototipos.

De esta forma tuvimos que dar un total de tres vueltas a nuestro modelo en espiral para completar el desarrollo.

1. Programas Políticos

Estos fueron los primeros bocetos que definían la interacción de la visualización de los programas políticos:

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

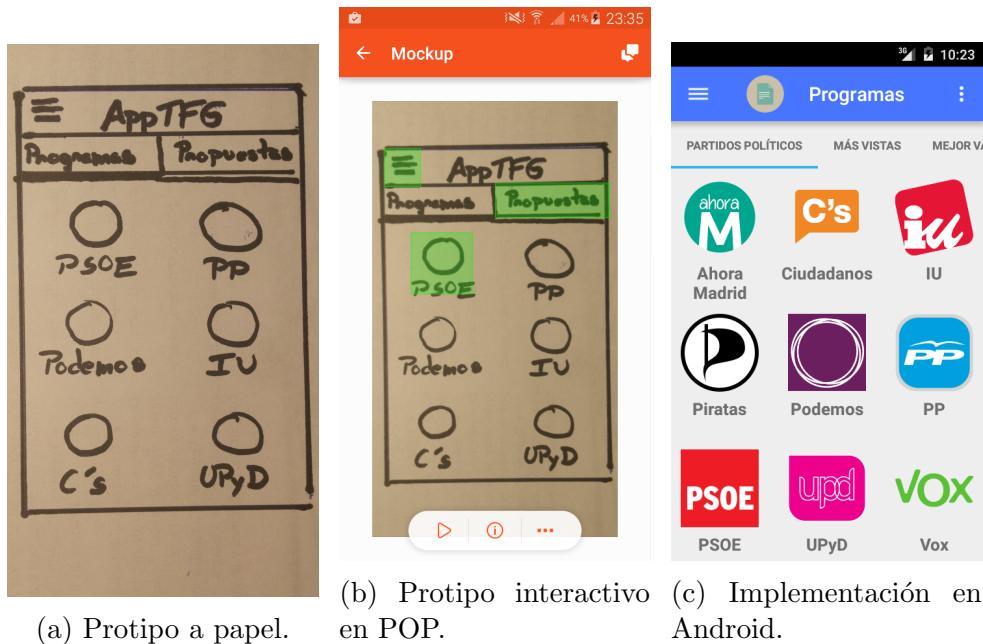


Figura 5.11: Primeros prototipos sobre programas electorales.

Elementos del prototipo a papel:

- a) Partidos políticos
- b) Programas políticos
- c) Índice de programas políticos
- d) Secciones de prorammas políticos

Funcionalidades del prototipo interactivo:

- a) Visualizar la lista de partidos políticos.
- b) Seleccionar un programa político.
- c) Navegar por el índice de un programa polítcico.
- d) Visualizar secciones de un programa, valorarla y comentarla.

Implementación en Android:

- a) Descargar la lista de partidos políticos del servidor.
- b) Descargar el programa seleccionado.
- c) Navegar por el índice del programa.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

- d) Descargar el contenido de las secciones.
- e) Añadir comentarios al servidor.
- f) Modificar parámetros sociales en el servidor (like, dislike, views, etc).

2. Bocetos de Propuestas Ciudadanas

A partir de lo desarrollado hasta el momento, se procedió a diseñar las propuestas en la aplicación. Manteniendo la coherencia con el diseño anterior, contuamos desarrollando los prototipos que añadían la funcionalidad de las propuestas ciudadanas.



Figura 5.12: Prototipos para el desarrollo de propuestas.

Elementos del protipo a papel:

- a) Propuestas ciudadanas.
- b) Listado de propuestas por tops.
- c) Listado de secciones de programas por tops.
- d) Desarrollar una propuesta.
- e) Pantalla principal rediseñada.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Funcionalidades del prototipo interactivo con POP:

- a) Visualizar propuesta.
- b) Listado de propuestas por tops.
- c) Listado de secciones de programas por tops.
- d) Desarrollar una propuesta.
- e) Pantalla principal rediseñada.

Implementación en Android:

- a) Visualizar propuesta almacenada en el servidor.
- b) Listado de propuestas por tops sincronizada con el servidor.
- c) Agregar una nueva propuesta al sistema.
- d) Pantalla principal rediseñada.

3. Bocetos de Propuestas Colaborativas

Escenarios *key path* Escenarios de validación

5.3.6. Principios de Diseño

Para verificar los principios de diseño que cumple el diseño de la aplicación, utilizaremos los 10 principios de diseño propuestos por Jakob Nielsen [101] para evaluar la calidad del sistema.

1. **Visibilidad del estado del sistema:** Siempre que el sistema ejecuta tareas o no puede continuar por algún motivo, se mantiene informado al usuario sobre lo que está pasando. Esto podemos encontrarlo cuando el usuario accede a los programas políticos y el sistema procede a descargar la información. Mientras se descarga la información, se muestra un pequeño diálogo indicando la descarga de los datos para mantener al usuario informado.
2. **Relación entre el sistema y el mundo real (Metáforas y lenguaje familiares):** El sistema mantiene un lenguaje próximo al usuario, evitando términos más formales del sistema. Cuando visualizamos una sección de un programa o una propuesta, el sistema emplea el lenguaje común de términos como *like*, *dislike*, etcétera, para valorar las secciones de un programa o una propuesta.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

3. **Control y libertad del usuario:** La aplicación permite deshacer acciones fácilmente como puede ser cambiar la opinión de un *like* o *dislike*, o cancelar la acción de publicar una nueva propuesta o comentario.
4. **Consistencia y estándares:** La aplicación sigue las pautas de diseño propuestas por el framework de diseño *Material Design* [102]. Este estilo de diseño fue propuesto por Google en 2014, y en la actualidad la mayor parte de aplicaciones móviles utilizan este framework a la hora de diseñar sus pantallas. Por lo que los usuarios no deberían tener problema navegando y utilizando los controles de la aplicación, ya que con meramente conocidos en un amplio catálogo de aplicaciones. Agregar una propuesta desde un botón inferior situado a la derecha, representar un listado de objetos mediante *tabs* deslizables, son algunas de las características destacables de *Material Design*.
5. **Prevención de errores:** La aplicación muestra mensajes de error cuando algo le impide funcionar con normalidad. De esta forma se avisa al usuario de que algo no está funcionando como debería. Cuando ejecutamos la aplicación y el sistema detecta que no hay conexión, la aplicación notifica al usuario en un sencillo diálogo la necesidad de tener conexión a internet para operar con la aplicación.
6. **Reconocimiento mejor que recuerdo:** Para agregar una nueva propuesta o comentario, tan sólo se necesita una transición entre dos pantallas. De esta forma facilitamos al usuario la generación de nuevos contenidos sin saturarle la memoria u obligarle a recordar mucha información para llegar a su objetivo. Este principio se encuentra en la mayor parte de las aplicaciones móviles, pues el usuario suele interactuar con el dispositivo en pequeños intervalos de tiempo y además no suele tener una postura cómoda que le facilite la concentración. Por lo que se requieren pocas transiciones entre pantallas, tareas sencillas y elementos sencillos y distinguibles.
7. **Flexibilidad y eficiencia:** Se utilizan atajos a funcionalidades disponible desde la mayor parte de las pantallas del sistema. Haciendo uso del botón *hamburguesa* en *material design* nos da la posibilidad de abrir un menú con accesos directos a las principales partes de la aplicación. De tal forma que no necesitamos volver hacia atrás para seleccionar otra opción. No obstante, ambos caminos pueden ser utilizados en función de la habilidad o conocimiento del usuario con la aplicación.
8. **Estética y diseño minimalista:** Tanto los diálogos que muestra el sistema como la información visible en la mayoría de las pantallas del

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

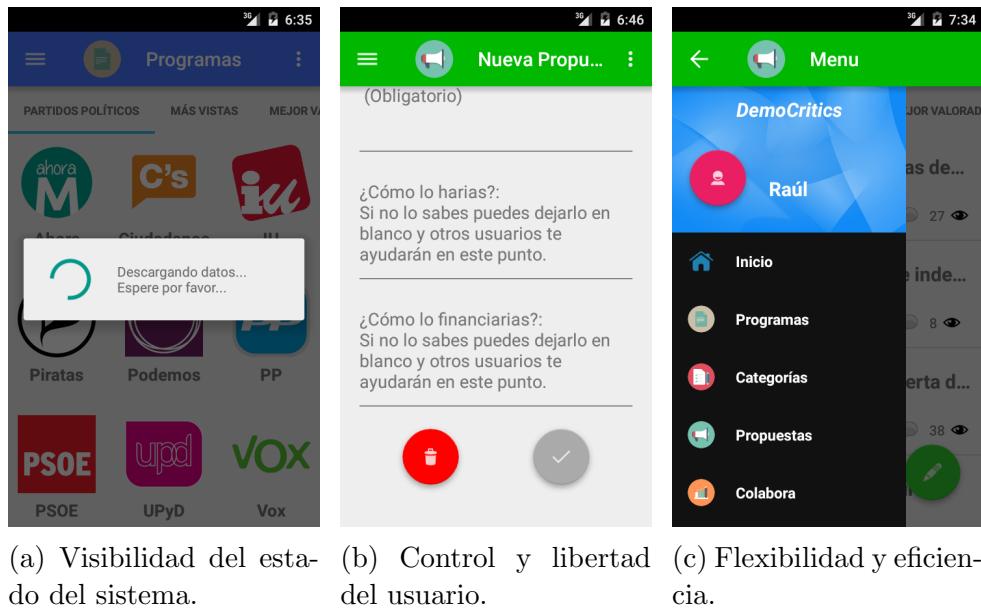


Figura 5.13: Principios de diseño en la aplicación.

sistema. Así evitamos dar información innecesaria al usuario o distraer su atención para evitar su confusión.

9. **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores:** Cuando el usuario da con un error inesperado, el sistema informa de lo ocurrido mediante un sencillo mensaje de error. Simplificando el mensaje mostrado al usuario para que pueda resolverlo sin grandes problemas.
10. **Ayuda y documentación:** Cuando el usuario genera contenido, el sistema va indicando al usuario los pasos que tiene que seguir para completar su tarea de forma satisfactoria. Esto sucede en el caso de que un usuario construye una nueva propuesta, donde el sistema le indica los campos que tiene que llenar y la información que debe ir en cada campo.

5.4. Implementación de DemoCritics

5.4.1. Metodología

Durante el desarrollo del código de la aplicación hemos utilizado las metodologías que se detallan a continuación.

Control de versiones

Para mantener una copia de las versiones locales, hemos utilizado GIT como control de versiones. Realizando *commits* y subiendo los cambios de cada versión en un repositorio público alojado en GitHub.

GitHub

Para compartir el código de la aplicación y visualizar los cambios de cada commit, utilizamos una organización pública donde se almacenan todos los repositorios de la aplicación. La organización está disponible en el siguiente enlace: <https://github.com/Zorbel>.

Reparto de tareas

Durante el desarrollo del proyecto, hemos dividido las tareas en dos grupos principales. Por un lado teníamos la parte de *back-end*, formada por la API Rest desarrollada en Laravel y la base de datos. Mientras que por otro lado se encontraba la parte de *front-end* formada por la aplicación en Android. Estas fueron las dos ramas principales en las que dividimos el trabajo en un primer momento. El desarrollo inicial y final en Wave se produjo en paralelo.

Revisiones de código

Cada vez que completábamos una funcionalidad y esta quedaba reflejada en *GitHub*, se realizaba una pequeña revisión de código para verificar cómo se había implementado el objetivo a desarrollar. De esta forma evitábamos malinterpretar algunos aspectos que pueden diferir cuando se concretan a la hora de realizar la implementación del sistema. También recibimos revisiones de código externas por parte de los directores del Trabajo de Fin de Grado, que aportaron una visión más eficiente del código desarrollado.

CAPÍTULO 5. METODOLOGÍA DEL PROYECTO

Evaluaciones de usabilidad

Recibimos una evaluación de usabilidad externa por parte de un director del Trabajo de Fin de Grado, que valoraba aspectos a mejorar de la UIX. Esta evaluación nos sirvió para modificar algunos aspectos de representación gráfica que mejorarían la interpretación de los elementos por parte del usuario.

5.5. Evaluación con Usuarios

Capítulo 6

Arquitectura del Proyecto

6.1. Arquitectura de Wave

La arquitectura que soporta Wave tal y como la diseñó originalmente Google está estructurada en forma de capas que interactúan entre sí. De forma general y de abajo hacia arriba disponemos de las siguientes capas:

- Capa de Conexión Cliente/Servidor: Se encarga de gestionar la conexión entre cliente y servidor mediante el protocolo XMPP.
- Capa de Control de OT: Se encarga de gestionar la consistencia en tiempo real de los datos mediante la utilización de Transformaciones Operacionales (OT).
- Capa de Modelo de Datos Wave: Se encarga de gestionar los datos a nivel de las Waves que forman parte de una Conversación.
- Capa de Modelo Conversacional: Se encarga de gestionar los datos a nivel de las Conversaciones, junto a los documentos que las componen y los usuarios participantes (Ver Sección 6.1.1). En el caso de SwellRT se extendió este modelo a otro más generalizado llamado Modelo de Contenidos (Ver Sección 6.1.2).
- Capa de Interfaz de Cliente: API con operaciones para que el cliente interactue con el Modelo Conversacional subyacente mediante eventos. En el caso de SwellRT se extendió este API para utilizar el Modelo de Contenidos de esta tecnología.

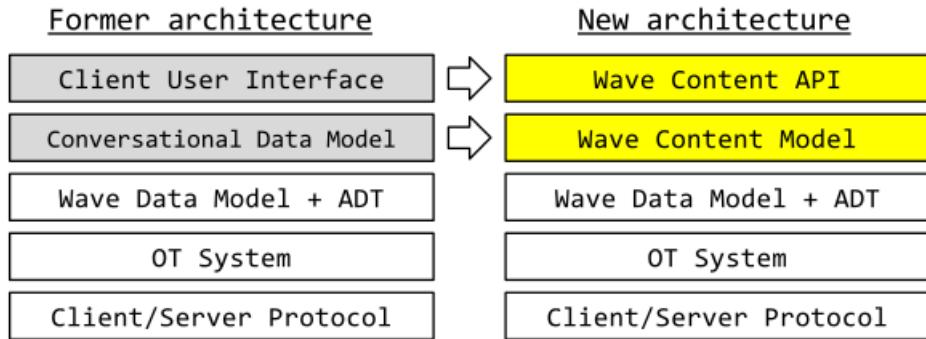


Figura 6.1: Arquitectura de Wave

6.1.1. Modelo Conversacional Wave

Además de definir el protocolo del que hace uso Wave, Google definió un Modelo de Datos Conversacional [103] 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 [104] [30]:

- **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** [105]: 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.

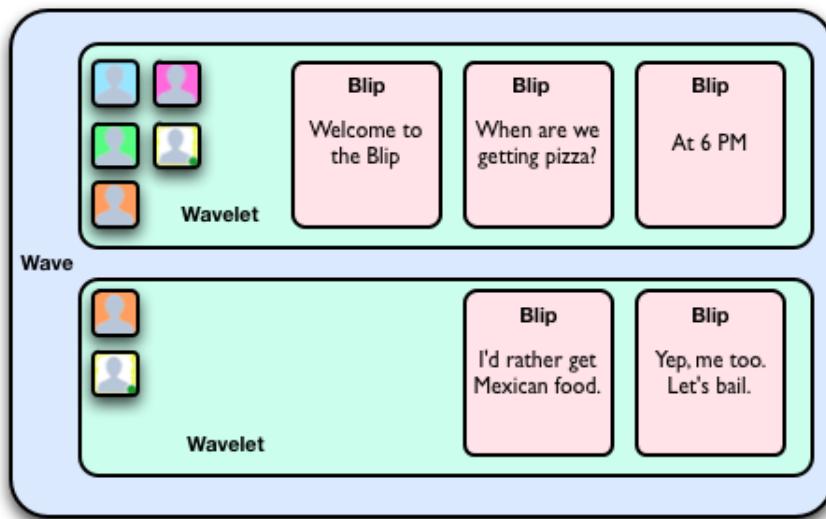


Figura 6.2: Modelo Conversacional de Wave

6.1.2. Modelo de Contenidos SwellRT

SwellRT [38] (Swell Real Time) es un framework de desarrollo para Wave que forma parte del proyecto europeo P2P Value [37]. Está basado en el servidor Wave In A Box [34] (WIAB) y extiende sus funcionalidades cambiando el Modelo Conversacional original creado por Google por uno nuevo de propósito más general llamado Modelo de Contenidos. En este modelo nuevo podemos intercambiar estructuras de datos de forma colaborativa, federada y en tiempo real que combinan los siguientes 3 tipos: **Mapas, Listas y Strings**.

De esta manera la estructura de datos intercambiada constará siempre de un Mapa inicial (llamado "root") del cual colgarán a modo de árbol el resto de

estructuras (de cualquiera de los tres tipos que maneja SwellRT) cuyos datos se gestionarán en tiempo real. Para ello existe también una API nueva que se encarga de la gestión de dicho arbol mediante la utilización de eventos que notifican de cambios en las estructuras de datos. Este modelo será el que utilicemos para la aplicación Android. El siguiente es un esquema de los cambios en el modelo original introducidos por SwellRT:

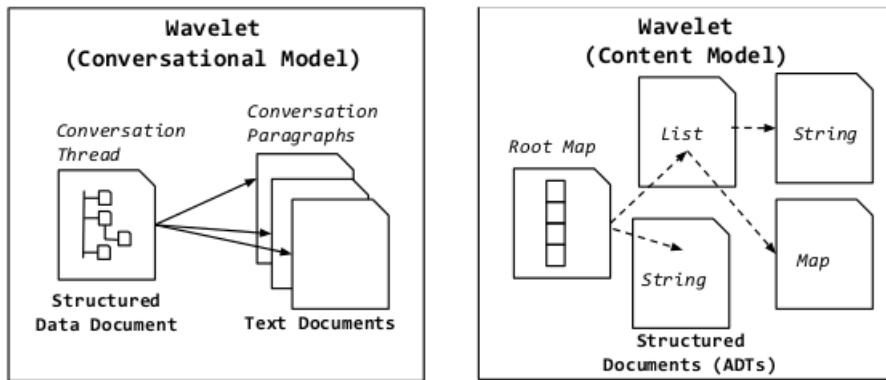


Figura 6.3: Cambio de Modelo de Wave

6.2. Arquitectura de la aplicación

La arquitectura de DemoCritics está compuesta por cuatro módulos principales de los que hablaremos en profundidad en las siguientes subsecciones. Como **Cliente móvil** tendremos la aplicación desarrollada en Android, que realizará peticiones HTTP al servicio web alojado en OpenShift [60], una plataforma que permite alojar servicios web de forma gratuita. Dentro del servicio contaremos con una **API RESTful** que será quien gestione las peticiones de la aplicación móvil mediante el protocolo HTTP.

CAMBIAR FOTO PARA VER INTEGRACION CON WAVE

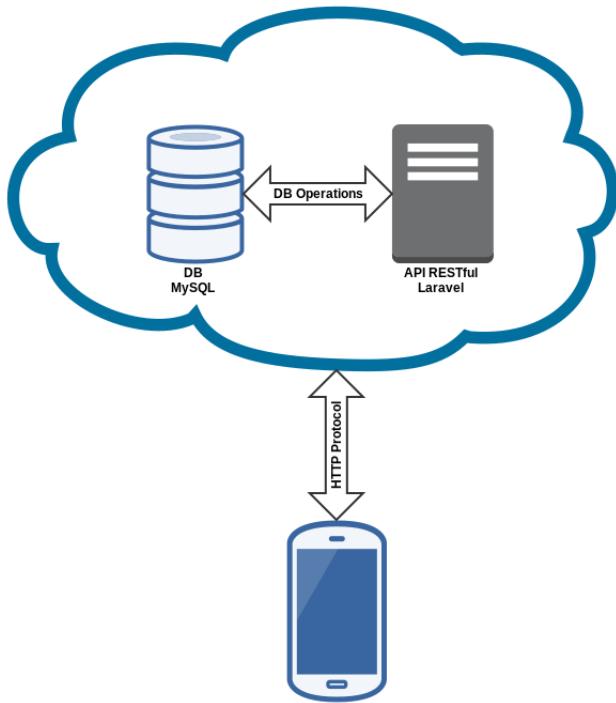


Figura 6.4: Arquitectura de la aplicación.

Por otro lado, la aplicación hará uso del **Servicio Android** desarrollado en SwellRT-Android [96] para conectarse con el servidor Wave alojado en <https://wave.p2pvalue.eu/> e intercambiar los datos que sea necesario mantener en tiempo real, es decir, la edición de una Propuesta de forma colaborativa entre varias personas.

Por último, la **Base de Datos MySQL** alojada en el servidor de OpenShift almacenará toda la información relacionada con la aplicación. La API RESTful será quien actúe de intermediario entre las peticiones de la aplicación y las operaciones en Base de Datos.

6.2.1. Base de datos

En la implementación de la Base de Datos se ha utilizado un Modelo Relacional para la definición de las tablas. Utilizando MySQL [55] como sistema de gestión de base de datos (SGBD) y phpMyAdmin [56] como herramienta de gestión gráfica de la base de datos.

La base de datos está formada por un total de 12 tablas donde se almacena

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

toda la información relacionada con los programas de los partidos políticos, las propuestas ciudadanas, encuestas, comparativas... y otros datos más técnicos como la gestión de los usuarios, la relación de los comentarios o la relación entre las secciones y comparativas entre otras.

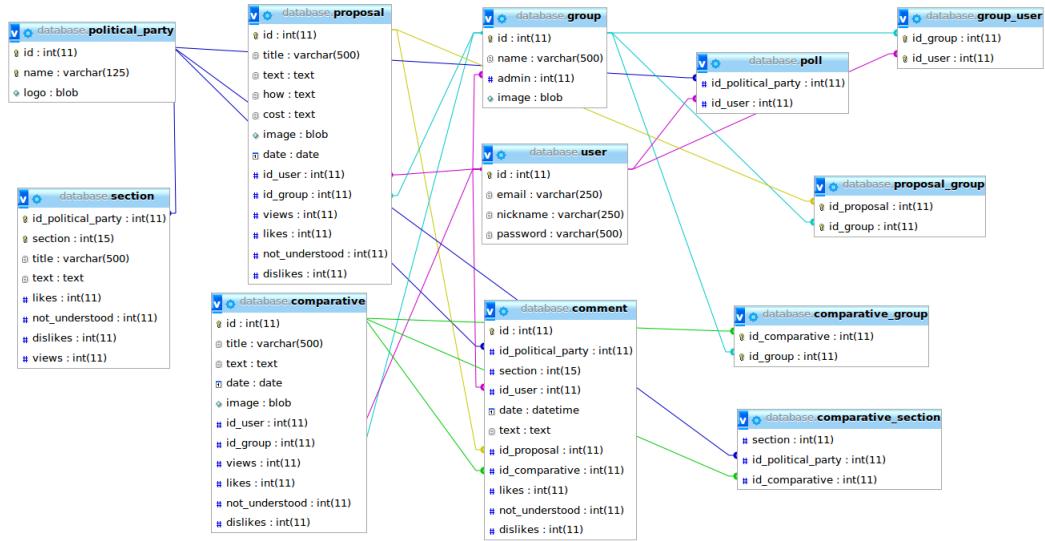


Figura 6.5: Modelo entidad-relación de la base de datos.

Las tablas *section* y *political-party* son utilizadas para guardar información estática en la aplicación. Es decir, en la tabla *political-party* se almacenan los partidos políticos que se presentan a unas elecciones, y en la tabla *section*, las diferentes secciones de un programa electoral. Tan sólo modificaremos las columnas de *likes*, *dislikes*, *not-understood* y *views* para obtener estadísticas de uso de cada sección. El resto de las columnas permanecerán intactas.

EXPLICAR RESTO DE TABLAS. DATOS ESTATICOS VS DINAMICOS

Las demás tablas serán utilizadas para guardar datos dinámicos de la aplicación. Datos que normalmente genera un usuario visitando una sección de un programa, creando una propuesta, haciendo un comentario, etc.

6.2.2. Service REST

Para establecer la conexión de la aplicación desarrollada en Android con la Base de Datos hemos utilizado **Laravel** como Servicio Web. Laravel [58] es

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

un framework de código abierto para desarrollar aplicaciones web con PHP 5. Laravel permite además montar una API REST (Representational State Transfer), un estilo de arquitectura software para sistemas hipermédia distribuidos como la World Wide Web. Este término se originó en una tesis doctoral sobre la web escrita por Roy Fielding [106]. La elección de Laravel como framework PHP fue debida a su flexibilidad, la facilidad para programar la aplicación y el gran soporte que tiene de la comunidad. Además Laravel cuenta con una licencia de software libre MIT, lo que nos permite continuar utilizando software open-source en todo el proyecto, y es uno de los frameworks más populares actualmente.

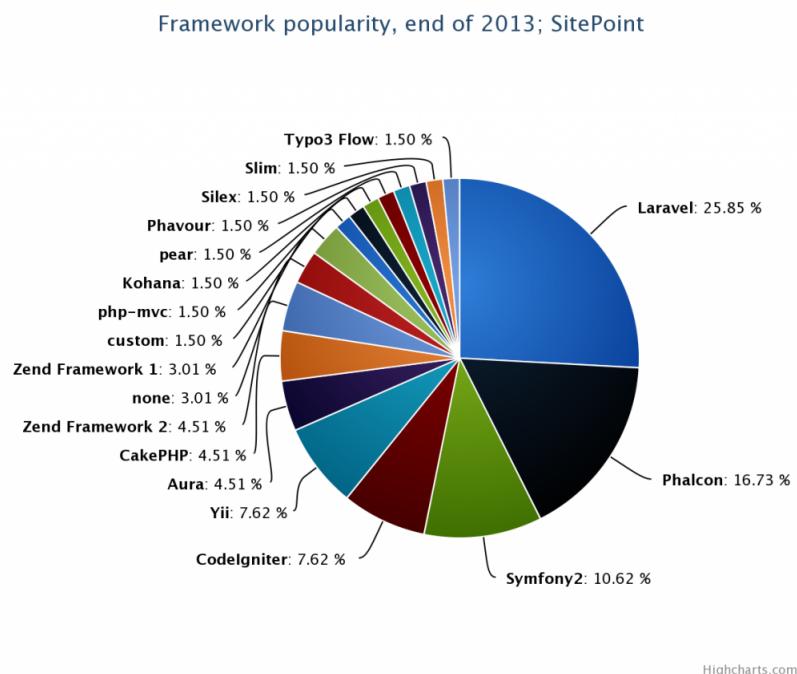


Figura 6.6: Frameworks PHP más populares. Fuente: SitePoint

Laravel nos permite implementar un sistema RESTful para que el cliente móvil pueda hacer peticiones al servicio web y que dicho servicio responda a éstas de la forma que queramos. Estas peticiones se realizan mediante el protocolo HTTP. En función de la operación que deseemos hacer, clasificaremos estas funciones en el acrónimo **CRUD** [107] (del original en inglés: **C**reate, **R**ead, **U**pdate and **D**elete).

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

Petición	Operación	SQL	Utilidad
GET	Leer	SELECT	Obtener un recurso almacenado en el servidor.
POST	Crear	INSERT	Crear un nuevo recurso en el servidor.
PUT	Actualizar	UPDATE	Actualizar un recurso almacenado en el servidor.
DELETE	Borrar	DELETE	Eliminar un recurso almacenado en el servidor.

Cuadro 6.1: Funciones CRUD

En función de las peticiones que realicemos al servicio realizadas mediante el protocolo HTTP, el servidor nos devolverá un código de estado para obtener *feedback* de lo sucedido. Por tanto distinguiremos diferentes códigos de estado cuando queramos obtener un recurso, actualizar un recurso, crear uno nuevo o borrarlo. En la siguiente tabla se definen los códigos de estado que puede devolvernos el servidor en función de nuestras peticiones.

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

Petición	Status Code	Descripción
GET	200 (OK)	El recurso solicitado ha sido devuelto correctamente.
GET	404 (Not Found)	El recurso solicitado no ha sido encontrado.
POST	201 (Created)	El nuevo recurso ha sido creado correctamente.
POST	404 (Not Found)	No se ha especificado el nuevo recurso a crear.
POST	409 (Conflict)	No se ha podido crear el recurso porque ya existe.
PUT	200 (OK)	El recurso ha sido actualizado correctamente.
PUT	204 (No Content)	No se ha especificado el recurso que pretende ser actualizado.
PUT	404 (Not Found)	El recurso a actualizar no ha sido encontrado.
DELETE	200 (OK)	El recurso solicitado ha sido borrado.
DELETE	404 (Not Found)	El recurso solicitado para borrar, no ha sido encontrado.

Cuadro 6.2: Códigos de estado de la respuesta del servidor

Usar un servicio RESTful nos proporciona una gran flexibilidad a la hora de independizar la tecnología del servidor de la del cliente. Mediante la arquitectura basada en peticiones HTTP no solo podremos hacer peticiones desde el cliente en Android, sino que más adelante podríamos desarrollar una versión web o incluso un cliente para iOS sin tener que modificar el servicio, ya que las peticiones HTTP serán las mismas.

Arquitectura

Laravel utiliza e implementa un funcionamiento basado en el patrón **MVC** (Model–view–controller), separando el modelo de la vista, y delegando la gestión al controlador. En nuestro caso tendríamos un modelo almacenado en las tablas de la base datos, donde se guardará toda la información dinámica y estática de la aplicación. El controlador sería encargado de gestionar las peticiones del cliente en Android en función del tipo de operación que requiera. Y por último, tendríamos como vista el resultado devuelto por el servidor con los datos solicitados por el cliente, que a su vez los representaría en la

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

aplicación móvil. Esta estructura de comportamiento hace independiente el modelo controlado por el servidor, de la vista que obtiene el usuario final en su cliente móvil. Con esto conseguimos eleborar un código más claro y sencillo, así como también evitar conflictos entre el modelo y la vista.

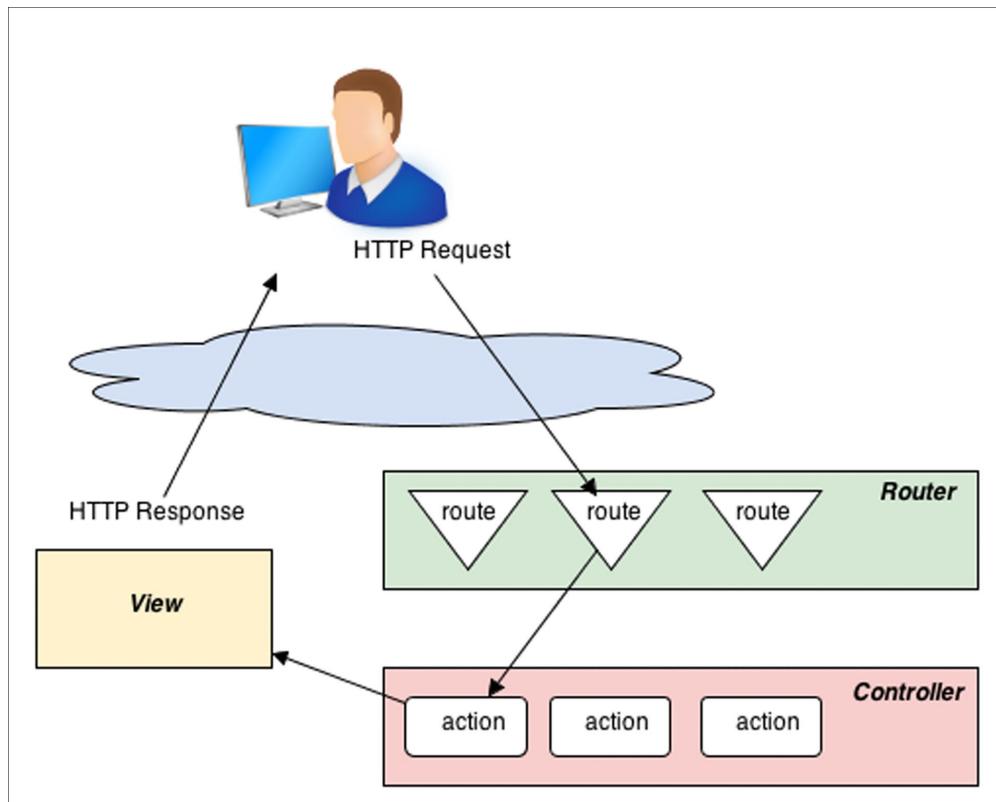


Figura 6.7: Arquitectura de Laravel

Para comprender la arquitectura de Laravel vamos a destacar tres componentes clave. En primer lugar tenemos una primera capa que gestiona las peticiones del cliente. Esta capa estará compuesta por una serie de **rutas** a las que el cliente realizará la petición en función de la operación que quiera realizar. Después, en función de la ruta que haya realizado la petición, se adentrará en una nueva capa compuesta por **controladores** que se encargarán de procesar la petición y devolver al cliente lo que ha solicitado. Éste último componente será la **vista**, que puede ser representada de muchas formas. En nuestro caso, la vista estará compuesta en una serie de datos en formato JSON, que posteriormente el cliente móvil se encargará de representar la vista. Visto esto, procederemos a detallar un poco más estos tres componentes fundamentales.

Rutas

Cómo citábamos en anteriormente, la primera capa de Laravel es un *mapa de rutas* que definirá todos los caminos posibles para llegar a la funcionalidad requerida de los controladores. Todas las rutas estarán definidas en el fichero **routes.php**, alojado en la carpeta *app/Http/routes.php* de nuestra instalación Laravel. El archivo *routes.php* es un fichero muy simple escrito en PHP donde definiremos todas las rutas posibles. La definición de una ruta irá acompañada de tres valores. El primero de ellos será el tipo de petición que realizará el cliente (ver tabla 6.1), seguido de la definición de la url por la que será accesible, y por último, indicaremos el controlador responsable de procesar la petición. A parte de indicar el controlador, también hay que indicar el método que procesará la petición dentro del controlador como veremos más adelante.

Para definir el conjunto de urls que forman el archivo *routes.php*, hemos seguido una serie de buenas prácticas [108] habituales a la hora de desarrollar una RESTful API. Normalmente utilizaremos nombres en lugar de verbos para acceder a los recursos. Por ejemplo, si queremos visualizar el listado de propuestas de la aplicación, utilizaremos la url */proposal*. Que será definida como:

```
Route::get('/proposal', 'ProposalController@index');
```

Así queda definida la ruta que accede a todo el listado de propuestas como una petición GET, que se encargará de procesarla el controlador *ProposalController* en su método *index()*.

Para acceder a una propuesta en concreto, se pasará un *id* para obtener el recurso adecuado. Este *id* irá a continuación de la url anteriormente definida. Por ejemplo, si quisiéramos acceder a la propuesta cuyo id es igual a 5, definiríamos la siguiente ruta:

```
Route::get('/proposal/{id}', 'ProposalController@show');
```

Nótese ruta es la misma que la anterior, a la que hemos añadido un nuevo parámetro que deberá especificar el usuario. Así la petición *GET /proposal/5*, nos devolvería la propuesta con id 5. Si nos fijamos, el controlador también es el mismo, pero esta vez el método encargado de procesar la petición será *show()*. Además este método recibirá el parámetro *id* que envíe el usuario en la petición.

Para crear una nueva propuesta en la aplicación, deberemos realizar una petición POST. Nuestra ruta será exactamente igual que la primera, pero

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

antes deberemos especificar que se trata de una ruta que atenderá a una petición POST:

```
Route::post('/proposal', 'ProposalController@create');
```

Donde una vez más en controlador *ProposalController* se encargará de procesar la petición en el método *create*. Si nos fijamos en la primera ruta dónde obteníamos las propuestas (*GET /proposal*, es exactamente igual que la que acabamos de definir. Sólo les diferencia el tipo de petición que está realizando el cliente. Para definir otros tipos de peticiones como PUT o DELETE, usaremos la misma metodología. El resultado final de las peticiones que podrían realizarse a una propuesta quedaría definida de la siguiente forma:

```
// Posibles operaciones con el objeto "propuesta"
Route::get('/proposal', 'ProposalController@index');
    // Obtiene el listado de todas las propuestas
Route::get('/proposal/{id}', 'ProposalController@show');
    // Obtiene la propuesta pasada por el campo {id}
Route::post('/proposal', 'ProposalController@create');
    // Crea una nueva propuesta en el servidor
Route::put('/proposal/{id}', 'ProposalController@update');
    // Actualiza la propuesta pasada por {id}
Route::delete('/proposal/{id}',
            'ProposalController@destroy');
    // Borra la propuesta pasada por {id}
```

Para organizar el código y poder visualizarlo de forma clara utilizaremos grupos de rutas. Esto nos resultará especialmente útil cuando tengamos varias operaciones que realizar dentro de una misma dirección, definiendo una nueva ruta cuya función será definida a continuación, incluyendo las rutas que vienen dentro del grupo. Si nos fijamos en el ejemplo anterior, podemos observar como todas las url comienzan con */proposal*. Por ello no es necesario definir en cada línea la ruta */proposal*, si no que bastará con definirla en un único grupo que adjuntará todas las rutas que comienzen de la misma forma. Así nuestro ejemplo anterior quedaría más claro y ordenado de la siguiente forma:

```
Route::group(['prefix' => 'proposal'], function()
{
    Route::get('/', 'ProposalController@index');
    Route::get('/{id}', 'ProposalController@show');
    Route::post('/', 'ProposalController@create');
    Route::put('/{id}', 'ProposalController@update');
    Route::delete('/{id}', 'ProposalController@destroy');
});
```

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

De esta forma podemos visualizar de un vistazo todas las posibilidades agrupadas dentro de la dirección *proposal*. Lo que nos permitirá utilizar grupos para ordenar funciones comunes dentro de un mismo prefijo, y además crear subgrupos dentro de los anteriores si tenemos algún prefijo repetido varias veces. Esto podría aplicarse a nuestro ejemplo anterior, agrupando las peticiones que requieren como parámetro un *id* de la propuesta:

```
Route::group(['prefix' => 'proposal'], function()
{
    Route::get('/', 'ProposalController@index');
    Route::post('/', 'ProposalController@create');
    Route::group(['prefix' => '/{id}'], function()
    {
        Route::get('/', 'ProposalController@show');
        Route::put('/', 'ProposalController@update');
        Route::delete('/', 'ProposalController@destroy');
    });
});
```

HACE FALTA ALGO MÁS...?

Controladores

Los controladores son los encargados de procesar todas las operaciones que intervienen en el modelo, es decir, en la información almacenada en la base de datos. Un controlador es un fichero escrito en PHP que será almacenado en la ruta */app/Http/Controllers* de la aplicación. Este fichero estará formado por una métodos y atributos que serán llamados desde las rutas definidas en la aplicación. Cada controlador hereda de una clase abstracta llamada *Controller*, que a su vez esta hereda de la clase abstracta *BaseController*. La cual implementa la funcionalidad básica de los controladores.

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

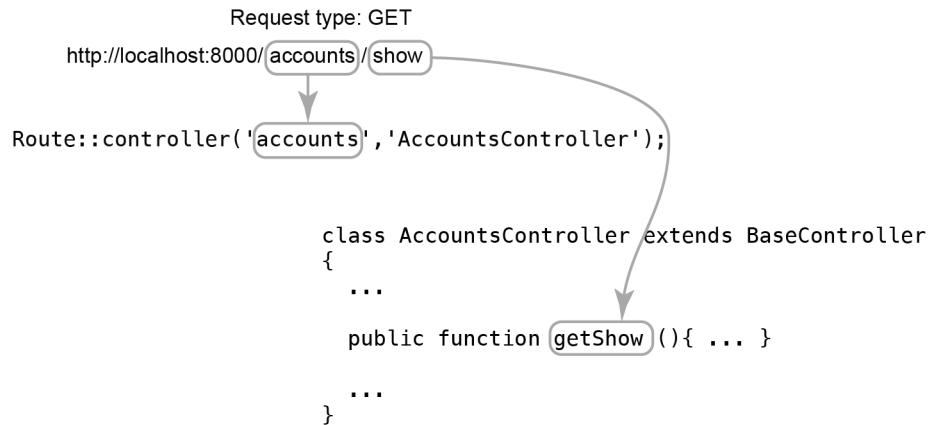


Figura 6.8: Camino desde la ruta al método del controlador.

Dentro de un controlador definiremos aquellos métodos que correspondan con las rutas de la aplicación. Estos métodos procesarán la información del cliente, operando con los parámetros que haya obtenido, accediendo a la base de datos, y devolviendo una respuesta en función del éxito de la operación. El siguiente ejemplo muestra una implementación básica del método *index()* del *ProposalController*:

```
<?php namespace App\Http\Controllers;  
  
use App\Http\Requests;  
use App\Http\Controllers\Controller;  
use DB;  
  
class ProposalController extends Controller {  
    /**  
     * Display a listing of the resource.  
     *  
     * @return Response  
     */  
    public function index()  
    {  
        return DB::select('SELECT * FROM `proposal`');  
    }  
}
```

La conexión a la base de datos se encuentra configurada en el archivo */config/database.php*, por lo que la gestión de la conexión y desconexión será controlada por Laravel. Lo que nos proporciona más independencia y versatilidad a la hora de programar. Como respuesta del método, Laravel utiliza por de-

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

fecto un fichero JSON con los resultados de la variable a la que asignemos el resultado, o a la sentencia SQL de la consulta que estemos devolviendo. El siguiente ejemplo muestra cuál sería el resultado de realizar una petición GET a la dirección */proposal*:

```
[{"id":1,"title":"Reducir la contaminaci\u00f3n en Madrid","text":"Actualmente la contaminaci\u00f3n en Madrid est\u00e1 llegando a unos 1\u00edmites por encima de la media de las principales ciudades de la Uni\u00f3n Europea. Por ello deber\u00e1mos reducir esa contaminaci\u00f3n ambiental acerc\u00e1ndonos a la media europea.", "how":"Cerrando el tr\u00f3fico en determinadas zonas de Madrid. Distrito: Zona Centro.", "cost":"Hacer 7 km de calles peatonales: 750.000 \u20ac", "id_image":4, "date":"2015-05-06 00:00:00", "id_category":4, "id_user":"6d823fa54c6d1a12", "views":4, "likes":3, "not_understood":0, "dislikes":0}, {"id":2,"title":"Restaurar el plan de estudios del '98","text": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent lobortis, sapien eget dignissim efficitur, augue eros molestie purus, et u gravida erat quam ual lacus. Fusce vel eros cursus, u venenatis udi uac, aliquet uris u. Fusce euismod ex uat u varius u lacinia. Sed at urna condimentum orci u volutpat eleifend nec id uerat. Sed at vestibulum neque. Maecenas u vehicula hendrerit felis, quis condimentum arcu u commodou eu. Sed u regestas, orci u eget u ultricies mollis, est u leo u porttitor sem, u at u elementum enim u arcu ut magna. ", "how": "Suprimiendo el plan Bolonia de los grados de 4 a 00f1os u licencias u y diplomaturas.", "cost": "Coste 0.", "id_image":2, "date": "2015-05-18 00:00:00", "id_category":2, "id_user": "d2d115f79ab3a7a4", "views": 12, "likes": 0, "not_understood": 1, "dislikes": 0}], ]
```

6.2.3. Servicio Android de Conexión con Wave

REORDENAR LO DE WAVE

6.2.4. Cliente Android

La plataforma Android divide el desarrollo de una aplicación en dos partes: la implementación de la lógica detrás de la aplicación mediante el uso de

Activities [77] o Services [93], y la configuración del aspecto de la interfaz mediante Layouts XML [52]. En las siguientes secciones hablaremos de algunos de los aspectos más destacados de este proyecto en referencia a dichas características de Android.

Interfaz Gráfica

En el desarrollo de esta aplicación quisimos utilizar un diseño plano basado en las últimas guías de diseño de Android disponibles en su última versión 5.0. Google denominó a este diseño plano "Material Design".

En Android, cuando creamos una Actividad (en su método "onCreate()") asignamos una Vista ("View") a la actividad que representa la interfaz del usuario. Esta vista está definida en un archivo de Layout XML que debemos modificar para crear la interfaz gráfica añadiendo componentes gráficos [109] ("widgets") y modificando sus atributos mediante etiquetas XML para adaptarlos a nuestras necesidades. Además podemos asignarles un identificador único para poder movernos por dichos widgets luego desde el código de la aplicación mediante una navegación en árbol utilizando el método "findViewById()" proporcionado por Android.

Estos componentes gráficos pueden ser los básicos proporcionados por Android (como cajas de texto, botones, listas, etc.) o se pueden crear *Vistas Personalizadas* en caso de que los widgets básicos no aporten la funcionalidad deseada.

En este último caso será necesario definir un Layout para la vista personalizada (que puede a su vez estar compuesto por widgets básicos o no) y una clase que posea referencias a los datos y a los componentes del layout que los albergarán.

Sin embargo a menudo querremos usar varias de estas vistas, ya sean básicas o personalizadas, juntas dentro de una lista, por lo que necesitaremos un *Adaptador*[110] ("Adapter") que haga de intermediario entre el layout y la clase que lo define para "adaptar" los datos que posee la clase a los componentes del layout que los muestran.

Veamos un ejemplo de esto con nuestra vista (widget) personalizada llamada **PartyWidget**, creada para mostrar cada uno de los elementos de la lista de Partidos Políticos que se visualiza en la aplicación. El funcionamiento de este widget personalizado está formado por los siguientes tres elementos:

- **party_widget.xml**: Layout XML que define la disposición de los dos

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

componentes gráficos básicos que conforman esta vista: un ”Image-View” para el logo del partido y un ”TextView” para el nombre del partido.

- **PartyWidgetView.java:** Clase que define el objeto contiene referencias a los dos componentes anteriores (mediante sus IDs) y los datos de la imagen (de tipo Bitmap) y el nombre (String) del partido.
- **PartyWidgetAdapter.java:** Clase que extiende de un ”BaseAdapter” de Android y define el adaptador que se encargará de gestionar una lista de elementos gráficos de un mismo tipo (en nuestro caso del tipo ”PoliticalParty”) y de crear las vistas que muestran los datos almacenados en dicha lista. El método más importante de un Adapter es el ”getView”, que crea la vista personalizada (en nuestro caso de tipo ”PartyWidgetView”) y le asigna los datos que hay dentro de cada elemento de la lista de la forma que nosotros le definamos.

De esta manera, solo es necesario decirle al elemnto que contiene la lista de Partidos Políticos (en nuestro caso un ”GridView” que permite visualizarlos en forma de rejilla que se adapta al tamaño de pantalla) que use como adaptador el PartyWidgetAdapter con los datos de la lista de Partidos Políticos que previamente nos hemos descargado.

Navegación por Tabs

Tras la fase de investigación decidimos que sería interesante incluir en la aplicación distintos filtros para ver tanto las secciones como las propuestas más valoradas positivamente, más vistas, más comentadas, etc. Para implementar la vista de esta funcionalidad decidimos utilizar un diseño muy parecido al que utiliza Google en su tienda de aplicaciones Google Play para navegar de forma lateral mediante pestañas (”tabs”) deslizantes. En nuestro caso cada una de las tabs mostraría una lista (elemento ”ListView” de Android) de elementos ordenados por distinto filtro.

No obstante, el método utilizado por Google en anteriores versiones de Android basado en añadir Tabs a la barra superior (”Action Bar”)[111] de la aplicación está actualmente obsoleto en el API 21, por lo que tuvimos que recurrir a la actual implementación de Google, que no se encuentra todavía dentro del API 21 de desarrollo. Es por eso que hubo que utilizar un par de clases (”SlidingTabLayout” y ”SlidingTabStrip”) que definen las tabs utilizadas por Google en el diseño con ”Material Design”. Estas clases fueron extraídas del GitHub[112] de la app que Google mostró en su aplicación de

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

prueba cuando presentó "Material Design" a finales de 2014. Este elemento trabaja definiendo el contenido de cada Tab mediante la utilización de Fragmentos[113] ("Fragment") de Android, que representan una porción de lo que se muestra dentro de una Actividad.

Sin embargo, la vista (Layout) de una Sección y de una Propuesta dentro de esta lista sería muy similar (con su foto, título e indicadores sociales de número de likes, vistas, etc.) por lo que ¿cómo hacer para utilizar la misma vista en dos objetos que son de tipos distintos? La solución que encontramos para utilizar el mismo elemento de visualización (definido en el layout llamado "top_ranking_item") en ambos casos fue utilizar una interfaz que implementarían tanto la clase "Section" como la clase "Proposal": la interfaz "TopItem".

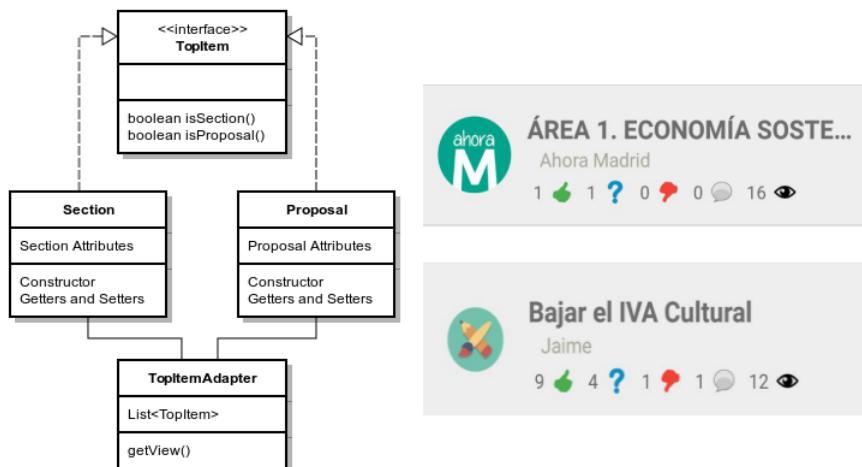


Figura 6.9: Esquema de Implementación de TopItem (Con vista de Section y Proposal)

De esta forma, las clases `Section` y `Proposal` implementan los métodos `"isSection()"` e `"isProposal()"` de la interfaz devolviendo true en cada caso respectivamente. Así, cuando dentro de nuestro Adaptador ("`TopItemAdapter`") tratemos los elementos de su lista de `TopItems` en su método `"getView()"`, podremos preguntar si se trata de un `Section` o un `Proposal` para llenar la vista del `ListView` con los datos correspondientes a cada caso. Solo hace falta por tanto crear listas de Secciones o Propuestas y pasárselas al nuestro adaptador personalizado después de definirlo como adaptador para el `ListView`.

Menús de Navegación

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

Con el objetivo de proporcionar al usuario una navegación por la aplicación que permita ir a cualquiera de sus secciones independientemente de en qué pantalla te encuentres actualmente, se decidió utilizar un menú lateral que se puede deslizar desde el lado izquierdo de la pantalla. Android permite añadir menús [114] a las Actividades mediante métodos específicos de éstas, aunque en este caso son menús fijos sencillos a los que se accede mediante el botón de opciones del móvil o de la app.

Nosotros queríamos utilizar el menú lateral deslizante presente en la mayoría de aplicaciones del estilo "Material Design" introducido por Google en su última versión de Android 5.0 (API 21), por lo que utilizamos un componente gráfico llamado "Navigation Drawer" para ello. Este componente está presente en todos los layouts de la aplicación y contiene dentro un "ListView" con las entradas del menú. Cada una de estas entradas es asimismo un componente personalizado por nosotros, definido en el layout `menu_left_item.xml` (contiene un logo y el nombre de la entrada de menú). En consecuencia, y tal y como se ha hablado antes en esta sección, existirá también un Adapter "MenuLeftListAdapter" que se encargará de gestionar la vista de la lista de entradas del menú. Además, se le ha añadido una cabecera ("Header") al menú, que contiene la información de nombre de Usuario y un botón para cambiar dicho nombre.

Sin embargo, si todas las Actividades que componen la aplicación disponen de este menú, ¿Cómo hacer para evitar repetir el código de generación y configuración de dicho menú en todas las Actividades? La solución encontrada fue utilizar una Actividad padre llamada "MenuActivity" que se encargaría de generar y de albergar los métodos que configuran el menú. De esta manera, cualquier Actividad de la aplicación extendería de "MenuActivity" y llamaría a sus métodos de generación configuración de menú con una referencia a su propia vista de Layout (recordemos que cada Layout contiene el elemento "Navigation Drawer" necesario para mostrar el menú).

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

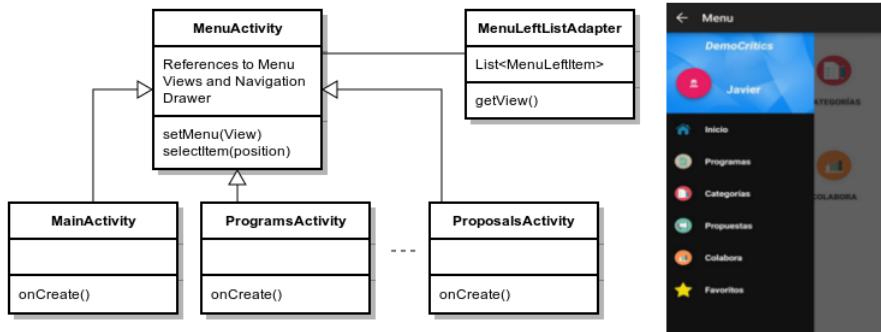


Figura 6.10: Esquema de Implementación de Menú Izquierdo (Con vista del Menú)

Existe también en la aplicación un Menú derecho deslizable que muestra un Índice del Programa de un Partido Político para poder navegar por él de manera más sencilla (similar al que utiliza la aplicación de la Wikipedia para navegar por un artículo). Este menú solo está presente en la actividad "SectionViewerActivity", ya que solo es necesario acceder al índice del programa cuando nos encontramos navegando por sus secciones. Utiliza también para ello un "Navigation Drawer" similar al del otro menú antes explicado. No obstante, en este caso decidimos utilizar un tipo distinto de lista ("ExpandableListView") que se diferencia de la lista normal en que posee dos niveles de navegación que se pueden expandir. De esta forma es necesario definir los layouts de los distintos niveles ("groups") y los subniveles ("childrens") dentro de cada uno.

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

Adapter	Layout	Clase	Descripción
PartyWidgetAdapter	party_widget.xml	PartyWidgetView	Widget para mostrar un elemento de la lista de Partidos Políticos
MenuLeftListAdapter	menu_left_item.xml	MenuItemLeft	Widget para mostrar las entradas del Menú lateral izquierdo.
TopItemAdapter	top_ranking_item.xml	TopItem	Widget para mostrar un elemento de la lista con los Tops de Secciones y Propuestas.
ListIndexAdapter	(solo muestra texto)	No es necesario	TextView (Android) que muestra los nombres del índice de un Programa Político.
ExpandableListAdapter	list_child_item.xml list_group_item.xml	No es necesario	TextView que contienen los títulos de las distintas secciones y subsecciones (desplegables) del índice de un Programa Político.
CommentListAdapter	comment_item.xml	Comment	Widget para mostrar un comentario de un usuario dentro de la lista de comentarios de una Sección o una Propuesta.
SampleFragmentPagerAdapter	tab_page_*.xml	TabPageFragment	Widget para definir lo que contiene cada tab mediante Fragmentos cuya vista es generada de forma dinámica.

Cuadro 6.3: Adapters, Views y Clases utilizadas en el proyecto

Estructuración de Programas Políticos

A la hora de almacenar los Programas Políticos en la Base de Datos tuvimos que estudiar la forma más adecuada de hacerlo para poder estructurar dichos programas en distintos niveles de secciones y subsecciones. En definitiva, el problema radicaba en cómo codificar la información de a qué nivel de anidamiento pertenecía cada Sección. Además, teníamos dos opciones: o bien el service REST devolvía un programa ya estructurado o la propia aplicación se encargaba de estructurarlos.

Codificación en Base de Datos

En principio queríamos elegir una codificación que nos permitiera realizar una consulta a la base de Datos que devolviera las distintas secciones y subsecciones **en el orden en el que aparecen en el Programa**. Tras varios intentos con distintos campos de Strings y enteros como atributo de la tabla de Secciones en Base de Datos, decidimos que lo mejor sería utilizar

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

un único número entero que codificara dicha información. Tras estudiar la composición de diversos programas identificamos que **no serían necesarios más de 4 niveles de anidamiento en subsecciones**. En consecuencia, el número estaría formado por 8 cifras, siendo cada dos cifras un nivel distinto de anidamiento. Veamos varios ejemplos de cómo funciona esta codificación:

01020302	<u>Ejemplos:</u>
Nivel: 1 2 3 4	
	01040000 -> 1.4
	10050200 -> 10.5.2
	02130607 -> 2.13.6.7
	01100101 -> 1.10.1.1

Figura 6.11: Ejemplos de Codificación de Secciones y Subsecciones

Utilizando esta codificación conseguimos que mediante una simple consulta SQL la Base de Datos nos devuelva todas las secciones del programa de un determinado partido de forma ordenada, siendo luego el Service REST el que pasará los resultados de esta consulta en JSON a la aplicación Android.

Estructuración y Almacenamiento de Secciones

Con dicha codificación conseguimos ordenar las secciones en el orden e el que aparezcan en el programa, pero ahora debíamos guardarlas en la aplicación en una estructura que nos permitiera navegar por el programa de forma estructurada pudiendo en cada momento ver las subsecciones o volver a la sección anterior. Decidimos por tanto **guardar el programa en una estructura en árbol formada por listas de listas**. De esta forma cada elemento de la lista sería un objeto del tipo Section que contendría la información de dicha sección (título, texto, etc.) y otra lista con las subsecciones.

Pero, ¿cómo construir esta estructura a partir de la lista ordenada que nos devuelve el Service REST? La solución encontrada fue utilizar una función que construyera dicho árbol de forma recursiva (Ver clase "GetProgramsData.java") sobre una Sección "root" vacía que pertenecería a cada Partido Político. Dicha función recorrería la lista de secciones sin estructurar y para cada sección comprobaría su nivel de anidamiento para saber si colocarlo en el nivel actual, en el siguiente o en el anterior.

```
protected int createIndex(Section parent, List<Section>
    JsonResult, int index) {
    currentSection
    n currentSection = JsonResult.get(index);
```

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

```
rent.getSections() == null) {  
    rent.setSections(new ArrayList<Section>());  
  
    .addSubSection(currentSection);  
    nextIndex  
    xtIndex = index + 1;  
    ((nextIndex < JSONResult.size()) && (getLevel(JSONResult.get(  
        nextIndex)) >= getLevel(currentSection))) {  
        (getLevel(JSONResult.get(nextIndex)) == getLevel(  
            currentSection)) {  
            currentSection = JSONResult.get(nextIndex);  
            nextIndex++;  
            parent.addSubSection(currentSection);  
        }  
        else if (getLevel(JSONResult.get(nextIndex)) > getLevel(  
            currentSection))  
            nextIndex = createIndex(currentSection, JSONResult,  
                nextIndex);  
  
    nextIndex;  
  
    int getLevel(Section sec) {  
        _sec = sec.getmSection(), level;  
        _sec % 100 != 0) {  
            vel = 4;  
            if (id_sec % 10000 != 0) {  
                vel = 3;  
                if (id_sec % 1000000 != 0) {  
                    vel = 2;  
  
                    vel = 1;  
                    level;
```

Además, como no queríamos tener que descargarnos el programa continuamente, esta estructura en forma de arbol se guardaría en una clase llamada "PoliticalGroups" que alamacenaría el listado de Partidos Políticos y para cada uno de ellos almacenaría la estructura en árbol de su programa (su nodo "root"). Esta clase implementa el **patrón de diseño Singleton** para asegurar que solo exista una instancia de ella, accesible desde cualquier punto de la aplicación.

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

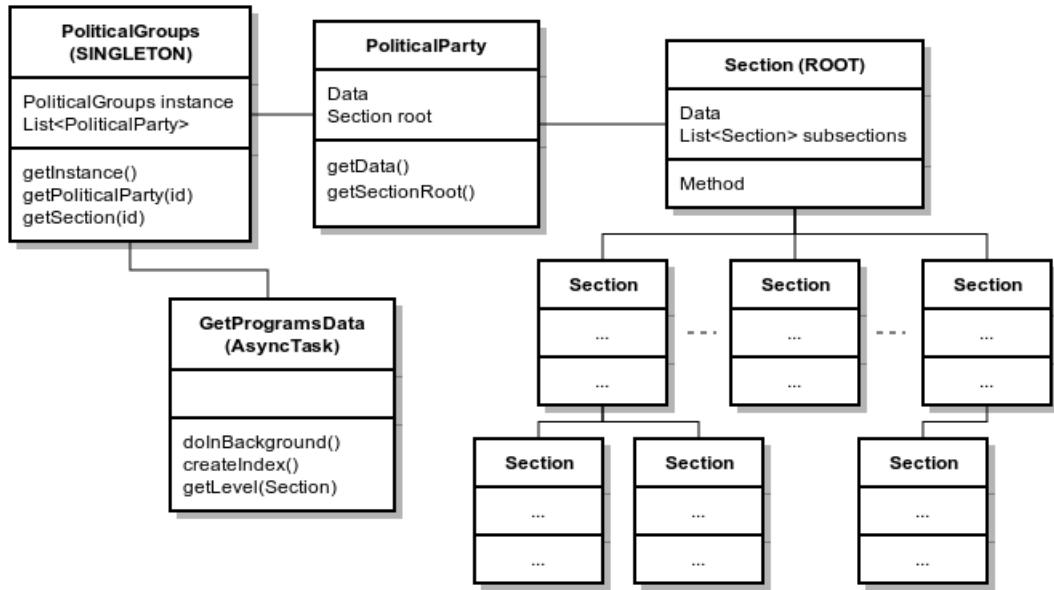


Figura 6.12: Esquema de estructura en árbol de Programa Político

Conexión y Peticiones a Service REST y Base de Datos

Las peticiones de información a Base de Datos (ya sea para crear datos, descargarlos o modificarlos) se realizan mediante el protocolo HTTP enviando dichas peticiones al Service REST (Ver Sección 6.2.2). Para enviar dichas peticiones es necesario por tanto realizar una conexión HTTP al Service REST desde la aplicación Android. Aprovechando la experiencia previa adquirida durante la migración de Wave (Ver Sección 5.2.3), decidimos utilizar la librería "HttpURLConnection" de Android y un esquema basado en AsyncTask que ejecutan el proceso de conexión y descarga de datos en un thread separado del UI Thread tal y como recomienda Google hacer para trabajar con conexiones de red[87].

Así, en función de la operación que se quiera hacer en Base de Datos, para cada tipo de consulta existirá un AsyncTask encargado de conectarse al servidor con la URL apropiada y tratar los datos en JSON que este le pueda devolver, ya sea para mostrarlos o guardarlos en algún objeto de la aplicación. De forma general podemos identificar cuatro tipos de peticiones al servidor (Ver tabla 6.1).

Con el objetivo de no repetir código decidimos estructurar el esquema de

CAPÍTULO 6. ARQUITECTURA DEL PROYECTO

peticiones HTTP mediante herencia de AsyncTasks. El siguiente diagrama esquematiza la estructura:

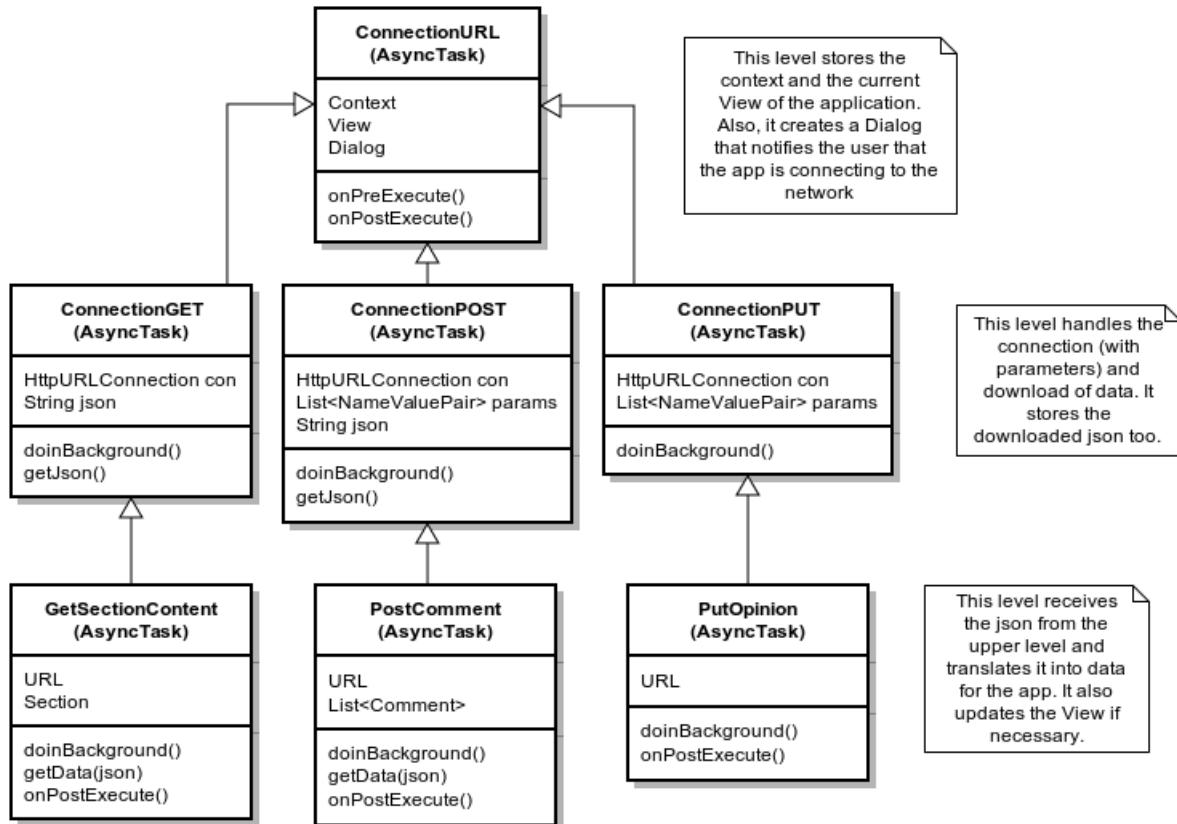


Figura 6.13: Esquema de Clases de Conexión HTTP mediante AsyncTask

Hace falta también actualizar la vista del usuario cuando se reciban los datos. Como un AsyncTask se ejecuta en un hilo aparte diferente del UI Thread (para no bloquear la interacción con el usuario).

Gestión de Usuarios

ID de teléfono

Peticiones a Wave

ARQUITECTURA DE PETICIONES A WAVE

Capítulo 7

Resultados y Conclusiones

7.1. Discusion de Resultados

7.1.1. Evaluación con usuarios

7.2. Conclusiones

Capítulo 8

Trabajo a Futuro

8.1. Mejoras

Bibliografía

- [1] Inc. Google. Google Wave Federation Protocol Over XMPP.
<http://wave-protocol.googlecode.com/hg/spec/federation/wavespec.html>.
- [2] Inc. Google. Meet Google Wave.
<http://googlecode.blogspot.com.es/2009/05/hello-world-meet-google-wave.html>.
- [3] Inc. Google. End of Google Wave.
<https://support.google.com/answer/1083134?hl=en>.
- [4] Google Inc. Google Docs.
<https://drive.google.com/>.
- [5] Apache. Apache License 2.0.
<http://www.apache.org/licenses/LICENSE-2.0>.
- [6] Apache. WIAB Repository.
<https://github.com/apache/incubator-wave>.
- [7] Apache. Install WIAB.
<https://cwiki.apache.org/confluence/display/WAVE/Install+WIAB>.
- [8] WIAB Server Example.
<http://waveinabox.net/>.
- [9] Google Inc. Realtime API: Add Realtime collaboration to your app.
<https://developers.google.com/google-apps/realtime/>.
- [10] Microsoft. RTC Client API.
<https://msdn.microsoft.com/en-us/library/ms775893%28v=vs.85%29.aspx>.
- [11] Google Inc. WebRTC.
<http://www.webrtc.org/>.
- [12] Mozilla Labs. TogetherJS: Collaboration made easy.
<https://togetherjs.com/>.

- [13] Joseph Gentle. ShareJS.
<http://sharejs.org/>.
- [14] Chianwu Tian. Goodow: Google Docs–style collaboration via the use of operational transforms.
<https://github.com/goodow realtime>.
- [15] The Etherpad Foundation. Etherpad.
<http://etherpad.org/>.
- [16] TitanPad Authors. TitanPad: lets people work on one document simultaneously.
<https://titanpad.com>.
- [17] Alexander Kirk. Colorillo.
<http://colorillo.com/>.
- [18] Henry Oswald and James Allen. ShareLaTex: LaTeX, Evolved.
<https://www.sharelatex.com/>.
- [19] Samepage Labs Inc. Samepage — The Num. 1 Online Collaboration Tool.
<https://www.samepage.io/>.
- [20] Quip. Quip: Trabaja con personas, no con archivos.
<https://quip.com/>.
- [21] Plaza Podemos: ¡Sí se puede!
<https://www.reddit.com/>.
- [22] Reddit: the front page of the internet.
<https://www.reddit.com/r/podemos/>.
- [23] Change.org · La mayor plataforma de peticiones del mundo.
<https://www.change.org/>.
- [24] Programa colaborativo de Zaragoza en Común.
<http://programa.ganemoszaragoza.com/>.
- [25] S.A. APPGREE. Appgree.
<http://www.appgree.com/que-es-appgree/>.
- [26] S.A. APPGREE. DemoRank: El Algoritmo.
<http://www.appgree.com/el-proceso-paso-a-paso/>.

- [27] Google+.
<https://plus.google.com/>.
- [28] Apache. Apache Wave (Incubating).
<http://incubator.apache.org/wave/about.html>.
- [29] Inc. Google. Wave Federation.
<http://www.waveprotocol.org/federation>.
- [30] Google. Google Wave Federation Architecture White Paper.
<http://wave-protocol.googlecode.com/hg/whitepapers/google-wave-architecture/google-wave-architecture.html>.
- [31] Peter Saint-Andre. Extensible messaging and presence protocol (xmpp): Core. 2011.
RFC 6120 Available at <http://tools.ietf.org/html/rfc6120>.
- [32] Understanding Operational Transformation.
<http://www.codecommit.com/blog/java/understanding-and-applying-operational-transformation>.
- [33] Inc. Google. Google Wave Operational Transformation.
<http://www.waveprotocol.org/whitepapers/operational-transform>.
- [34] Apache. Wave In A Box.
<http://www.waveprotocol.org/wave-in-a-box/>.
- [35] Oracle Corporation. OpenJDK.
<http://openjdk.java.net/>.
- [36] Google Inc. Google Web Toolkit.
<http://www.gwtproject.org/>.
- [37] P2PValue. P2P Value European Project.
<http://www.p2pvalue.eu/>.
- [38] P2PValue. SwellRT, a real-time federated collaboration framework.
<https://github.com/P2Pvalue/swellrt>.
- [39] Google Inc. Android Developers Site.
<http://developer.android.com/index.html>.
- [40] Eclipse Foundation. Eclipse IDE.
<https://eclipse.org/ide/>.

- [41] Oracle Corporation. Java.
<https://www.java.com/es/>.
- [42] Mozilla Foundation. JavaScript.
<https://developer.mozilla.org/es/docs/Web/JavaScript>.
- [43] Roy Fielding. Hypertext transfer protocol (http/1.1): Message syntax and routing. 2014.
RFC 7230 Available at <https://tools.ietf.org/html/rfc7230>.
- [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] Linus Torvalds. Git.
<https://git-scm.com/>.
- [46] GitHub Inc. GitHub · Build software better, together.
<https://github.com/>.
- [47] Google Inc. Android Studio IDE.
<https://developer.android.com/tools/studio/index.html>.
- [48] JetBrains. IntelliJ IDEA IDE.
<https://www.jetbrains.com/idea/>.
- [49] Google Inc. Android 5.0 APIs.
<https://developer.android.com/about/versions/android-5.0.html>.
- [50] Google Inc. Android SDK.
<https://developer.android.com/sdk/index.html>.
- [51] World Wide Web Consortium. Extensible Markup Language (XML).
<http://www.w3.org/XML/>.
- [52] Google Inc. Android Layouts.
<https://developer.android.com/guide/topics/ui/declaring-layout.html>.
- [53] Introducing JSON.
<http://json.org/>.

- [54] ISO. ISO/IEC 9075-1:2011 - Information technology – Database languages – SQL.
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=53681.
- [55] Oracle Corporation. MySQL :: The world's most popular open source database.
<https://www.mysql.com/>.
- [56] The PhpMyAdmin Project. PhpMyAdmin: Bringing MySQL to the web.
<http://www.phpmyadmin.net/>.
- [57] The PHP Group. PHP: Hypertext Preprocessor.
<https://php.net/>.
- [58] Taylor Otwell. Laravel - The PHP Framework For Web Artisans.
<http://laravel.com/>.
- [59] Taylor Otwell. The Laravel PHP Framework.
<https://github.com/laravel>.
- [60] Inc. Red Hat. OpenShift by Red Hat.
<https://www.openshift.com/>.
- [61] Jon Skinner. Sublime Text: The text editor you'll fall in love with.
<https://www.sublimetext.com/>.
- [62] WooMoo Inc. POP - Prototyping on Paper — Mobile App Prototyping Made Easy.
<https://popapp.in/>.
- [63] Free Software Foundation. The GNU General Public License v3.0 - GNU Project - Free Software Foundation.
<https://www.gnu.org/licenses/gpl.html>.
- [64] Google Inc. Eclipse Android Development Tools (ADT).
<https://developer.android.com/tools/help/adt.html>.
- [65] Google Inc. Android API Reference.
<http://developer.android.com/reference/packages.html>.
- [66] Google Inc. Android Virtual Device Manager.
<http://developer.android.com/tools/devices/managing-avds.html>.

- [67] Google Inc. Using the Android Emulator.
<http://developer.android.com/tools/devices/emulator.html>.
- [68] Google Inc. Android SDK Manager.
<https://developer.android.com/tools/help/sdk-manager.html>.
- [69] Google Inc. Android Dalvik Debug Monitor Server.
<http://developer.android.com/tools/debugging/ddms.html>.
- [70] Google Inc. Android Versions Distribution.
<https://developer.android.com/about/dashboards/index.html>.
- [71] Google Inc. Building Apps with Over 65K Methods.
<https://developer.android.com/tools/building/multidex.html>.
- [72] Google Inc. ProGuard.
<http://developer.android.com/tools/help/proguard.html>.
- [73] Apache Foundation. Apache Ant.
<http://ant.apache.org/>.
- [74] Google Inc. Building and Running from the Command Line.
<https://developer.android.com/tools/building/building-cmdline-ant.html>.
- [75] Google Inc. Setting up a Device for Development.
<https://developer.android.com/tools/device.html#setting-up>.
- [76] Google Inc. Android Debug Bridge.
<http://developer.android.com/tools/help/adb.html>.
- [77] Google Inc. Activities.
<https://developer.android.com/guide/components/activities.html>.
- [78] Roy Fielding. Hypertext transfer protocol (http/1.1): Authentication. 2014.
RFC 7235 Available at <https://tools.ietf.org/html/rfc7235>.
- [79] Google Inc. System Permissions.
<https://developer.android.com/guide/topics/security/permissions.html>.

- [80] Google Inc. App Manifest.
<https://developer.android.com/guide/topics/manifest/manifest-intro.html>.
- [81] Google Inc. Emulator Networking: Network Address Space.
<https://developer.android.com/tools/devices/emulator.html#emulatornetworking>.
- [82] Google Inc. Android LogCat.
<https://developer.android.com/tools/help/logcat.html>.
- [83] Google Inc. Apache HTTP Library.
<https://developer.android.com/reference/org/apache/http/package-summary.html>.
- [84] Google Inc. Android's HTTP Clients.
<http://android-developers.blogspot.com.es/2011/09/androids-http-clients.html>.
- [85] Google Inc. HttpURLConnection Library.
<https://developer.android.com/reference/java/net/HttpURLConnection.html>.
- [86] Google Inc. Processes and Threads.
<https://developer.android.com/guide/components/processes-and-threads.html>.
- [87] Google Inc. Connecting to the Network.
<https://developer.android.com/training/basics/network-ops/connecting.html>.
- [88] Async-IO.org. Atmosphere: The Asynchronous WebSocket/Comet Framework.
<https://github.com/Atmosphere/atmosphere>.
- [89] Async-IO.org. wAsync: A WebSockets/HTTP Client Library for Asynchronous Communication.
<https://github.com/Atmosphere/wasync>.
- [90] Wave Client for Android.
<https://github.com/Zorbel/swell-android>.
- [91] QOS.ch. SLF4J for Android.
<http://www.slf4j.org/android/>.

- [92] Google Inc. Android AsyncTask.
<https://developer.android.com/reference/android/os/AsyncTask.html>.
- [93] Google Inc. Android Services.
<https://developer.android.com/guide/components/services.html>.
- [94] Async-IO.org. Async Http Client.
<https://github.com/AsyncHttpClient/async-http-client>.
- [95] Project Grizzly. Project Grizzly: NIO Event Development Simplified.
<https://grizzly.java.net/index.html>.
- [96] P2PValue. SwellRT Client for Android.
<https://github.com/P2Pvalue/swellrt/tree/master/android>.
- [97] Alan Copper. *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity*. Sams Publishing, 2004.
- [98] Labodemo.
<http://labodemo.net/es/>.
- [99] Javier de la Cueva. Javier de la Cueva: abogado, doctor en Filosofía y estudiioso de las relaciones entre el Derecho y la Tecnología.
<http://www.javierdelacueva.es/bio/>.
- [100] Javier de la Cueva. *Manual del ciberactivista. Teoría y práctica de las acciones micropolíticas*. Bandaàparte Editores, 2015.
- [101] Jakob Nielsen. 10 Heuristics for User Interface Design: Article by Jakob Nielsen.
<http://www.nngroup.com/articles/ten-usability-heuristics/>.
- [102] Google Inc. Material design - Google design guidelines.
- [103] Inc. Google. Google Wave Conversation Model.
<https://wave-protocol.googlecode.com/hg/spec/conversation/convspec.html>.
- [104] Inc. Google. Google Wave API Overview.
<http://www.waveprotocol.org/wave-apis>.

- [105] Inc. Google. Google Wave Extensions.
<http://www.waveprotocol.org/wave-apis/extensions>.
- [106] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.
- [107] Martin Heller. REST and CRUD: the Impedance Mismatch — InfoWorld.
<http://www.infoworld.com/article/2640739/application-development/rest-and-crud--the-impedance-mismatch.html>.
- [108] Stefan Jauker. 10 Best Practices for Better RESTful API — Thinking Mobile.
<http://blog.mwaysolutions.com/2014/06/05/10-best-practices-for-better-restful-api/>.
- [109] Google Inc. Android Widget.
<https://developer.android.com/reference/android/widget/package-summary.html>.
- [110] Google Inc. Android Adapter.
<https://developer.android.com/reference/android/widget/Adapter.html>.
- [111] Google Inc. Android Action Bar.
<https://developer.android.com/guide/topics/ui/actionbar.html>.
- [112] Google Inc. IOsched app Sliding Tabs.
<https://github.com/google/iosched/blob/master/android/src/main/java/com/google/samples/apps/iosched/ui/widget/SlidingTabLayout.java>.
- [113] Google Inc. Android Fragments.
- [114] Google Inc. Android Menus.
<https://developer.android.com/guide/topics/ui/menus.html>.