UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática

Universitat Politècnica de València

Redes sociales y discapacidad visual: Desarrollo de un cliente de Twitter para gente con problemas de baja visión en sistemas iOS.

Proyecto Final de Carrera

Ingeniería Técnica en Informática de Gestión

Autor: Héctor Darío Arroyo Gómez

Director: Antonio Garrido Tejero

Director: Javier Sánchez Sierra

Junio 2012

Agradecimientos

Durante mi estancia en Raylight Soluciones Tecnológicas he aprendido muchas cosas a nivel técnico, pero también he crecido como persona. En estas líneas quiero aprovechar para agradecer a todas las personas que han hecho posible la consecución de este proyecto: en primer lugar, a mis padres, por la paciencia que han tenido conmigo y el apoyo que me han dado y me siguen dando día a día; a Joaquín Selva Roca de Togores, "Quino", CEO de Raylight, discapacitado visual, excelente líder y mejor persona; a Enric Montesa, impresionante en el marketing y la estrategia de negocio, además de una gran persona; a Javier Sánchez Sierra, brillante tutor, increíble profesional y , gracias a esta experiencia, también amigo; al resto del equipo de Raylight (Voro, Iván, Maite, Davinia y Pau), qué son algo más que compañeros; a Antonio Garrido Tejero, por aceptar ser el tutor en la universidad de este proyecto, por su paciencia, y por todos los consejos que me ha dado; y por si me olvido de alguien, a todas las personas que me han apoyado durante este periodo, y a todas aquellas que trabajan día a día para normalizar y facilitar lo máximo posible la vida a la gente que sufre cualquier tipo de discapacidad.

Resumen

iLoowi es el nombre comercial de la suite de aplicaciones desarrolladas por la compañía Raylight Soluciones Tecnológicas S.L. (a partir de ahora, Raylight) para dispositivos Apple con el sistema operativo iOS (iPhone y iPad en sus distintas versiones). Esta suite trata de acercar la tecnología actual a las personas que padezcan alguna discapacidad visual, sea parcial o total. Con un objetivo ambicioso, Raylight quiere ir más allá del desarrollo de las aplicaciones básicas de este tipo de dispositivos, intentando romper las barreras que, aún a día de hoy, siguen existiendo para este grupo de personas.

Este documento trata de dar una visión técnica del proceso llevado a cabo para la realización de un cliente de Twitter para personas con discapacidad visual acorde con la filosofía de la compañía. En él se detallarán todas las fases que se han ido sucediendo hasta la finalización del producto, desde los estudios preliminares con usuarios discapacitados, el diseño de un nuevo tipo de interacción basado en los resultados de dicho estudio (y qué ha sido registrado en la "United States Patent and Trademark Office"), el desarrollo de una librería de controles que permita el uso de este nuevo tipo de interacción en dispositivos con el sistema operativo iOS, y, finalmente, el desarrollo de una aplicación, un cliente de Twitter que permita el acceso, la comunicación y el uso por parte de los discapacitados visuales de esta red social tan extendida y famosa, dando la oportunidad de conectar a este grupo de personas con los más de 550 millones [1] de usuarios registrados en este servicio.

Palabras clave: iOS, Twitter, discapacidad visual, baja visión, smartphones

Tabla de contenidos

1. Introducción 7
1.1 Motivación 7
1.2 Situación y problemática actual 8
1.3 Soluciones actuales 9
1.3.1 Android TalkBack 9
1.3.2 VoiceOver 10
1.3.3 Siri 11
1.4 Objetivos del proyecto 12
2. Conceptos previos 13
2.1 Conceptos la experiencia de usuario 13
2.2 Planteamiento teórico de la solución 15
3. Herramientas, tecnologías y licencias 17
3.1 Herramientas 17
3.1.1 Herramientas hardware 17
3.1.2 Herramientas software 18
3.2 Tecnologías 20
3.2.1 Paradigma de programación orientada a objetos 20
3.2.2 Lenguaje de programación Objective-C 21
3.2.3 Patrón de diseño de aplicaciones Model-View-Controller 23
3.2.4 iOS SDK 24
3.2.5 Frameworks 25
3.2.6 API de servicios REST de Twitter 26
3.2.7 JSON 27
3.2.8 Text-To-Speech 28
3.2.9 UML 28
3.3 Licencias 29
3.3.1 Licencia de desarrollador iOS de Apple 29
3.3.2 Licencia para el uso de la librería de TTS de Acapela 30
3.3.3 Licencia de desarrollador de Twitter 30

4. Especificación de requisitos 31
4.1 Introducción 31
4.1.1 Propósito 31
4.1.2 Ámbito del sistema 32
4.1.3 Definiciones, acrónimos y abreviaturas 32
4.1.4 Referencias 33
4.1.5 Visión global 33
4.2 Descripción general 33
4.2.1 Perspectiva del producto 34
4.2.2 Funciones del producto 34
4.2.3 Características del usuario 35
4.2.4 Restricciones generales 35
4.2.5 Supuestos y dependencias 36
4.2.6 Requisitos futuros 36
4.3 Requisitos específicos 36
4.3.1 Interfaces externas 36
4.3.2 Funciones 37
4.3.3 Requisitos de rendimiento 38
4.3.4 Restricciones de diseño 38
4.3.5 Atributos del sistema 39
4.3.6 Otros requisitos 39
5. Análisis 40
5.1 Diagrama de casos de uso 40
5.2 Diagrama de actividades 45
5.3 Diagrama de clases 46
5.4 Servicios REST 49
5.5 Prototipos de las interfaces gráficas de usuario 52
6. Desarrollo 56
6.1 Estudio preliminar 56
6.1.1 Aplicaciones de test 56
6.1.2 Conclusiones y resultados del test preliminar 60

6.2 Diseño y desarrollo de la librería de controles accesibles 61
6.2.1 Diseño de la librería de controles accesibles 61
6.2.2 Desarrollo de la librería de controles accesible 64
6.3 Pruebas de la librería de controles accesibles 67
6.4 Desarrollo de un cliente de Twitter accesible 68
6.5 Pruebas de la aplicación 71
6.6 Distribución de la aplicación en el AppStore 72
6.7 Ampliaciones 73
7. Conclusiones 74
Anexo I. Manual de usuario 77
Anexo II. Referencias 82

1. Introducción

Un smartphone, o teléfono inteligente, es un dispositivo que combina la capacidad computacional de un ordenador corriente con la conectividad e independencia de un teléfono móvil. Ambos, teléfono y ordenador, son inventos del siglo XX (en realidad el teléfono es un invento del siglo XIX, pero fue en el siglo XX cuando se extendió al gran público), pero la combinación de ambos ha sido, sin lugar a dudas, el avance tecnológico más importante desde Internet. En el mundo de la informática, eso fue ya hace muchos años.

Este avance no viene dado por el simple hecho de unir un teléfono móvil y un ordenador. Tim Cook, CEO de Apple, dijo en una conferencia que "puedes combinar una tostadora con un frigorífico, pero eso no significa que le vaya a gustar al usuario". Con esta frase, aparte de mandar un mensaje a la competencia acerca de sus estrategias comerciales, nos da a entender porqué los smartphones han tenido tanto éxito: tecnología, usabilidad y utilidad. Eso, combinado con diseños elegantes, y excelentes campañas de marketing han logrado que, según Gartner Group (figura 1), al final del período entre enero y marzo de 2012 se hayan distribuido alrededor de 420 millones de estos dispositivos [2].

Worldwide Mobile Device Sales to End Users by Vendor in 1Q12 (Thousands of Units)

Company	1Q12 Units	1Q12 Market Share (%)	1Q11 Units	1Q11 Market Share (%)
Samsung	86,567.6	20.7	68,782.0	16.1
Nokia	83,162.5	19.8	107,556.1	25.1
Apple	33,120.5	7.9	16,883.2	3.9
ZTE	17,439.3	4.2	10,788.7	2.5
LG	14,720.4	3.5	23,997.2	5.6
Huawei Device	10,796.1	2.6	7,002.9	1.6
Research in Motion	9,939.3	2.4	13,004.0	3.0
Motorola	8,368.2	2.0	8,789.7	2.1
Sony Mobile Communications	7,898.4	1.9	7,919.4	1.9
HTC	7,703.4	1.8	9,313.5	2.2
Others	139,392.6	33.3	153,809.0	35.9
Total	**419,108.3**	**100.0**	**427,845.7**	**100.0**

Source: Gartner (May 2012)

Figura 1: Venta de smartphones hasta el primer trimestre de 2012.

Por otro lado, la aparición de las redes sociales ha revolucionado la forma de comunicarse y de estar conectado con el resto del mundo. Ya no sólo se suben fotos o videos a la red, si no que se comparten ideas, sensaciones y momentos con aquellos que son tus "amigos" o que son tus *followers*, y que, en definitiva, son personas que te interesan de uno u otro modo. Todo esto en tiempo real y sin importar el lugar (al menos esa es la teoría, pero por motivos políticos hay países que tienen restringido el acceso a estas redes).

1.1 Motivación

Está claro que todos queremos disfrutar de los avances que las nuevas tecnologías ofrecen, pero no siempre estas tecnologías están pensadas y adaptadas para todos. En este contexto, la accesibilidad para personas con discapacidad visual ofrecida por las principales plataformas, o no es todo lo adecuada que debería de ser (VoiceOver de iOS, TalkBack de Android), o es, simplemente inexistente (WindowsPhone 7.5).

En el sistema operativo iOS de Apple se ofrece VoiceOver (Siri es un proyecto en fase beta y que no tiene versión en castellano, se hablará de esta herramienta más adelante), aplicación dependiente del sistema, como solución de accesibilidad en dispositivos con el sistema operativo iOS. Es una herramienta bastante completa, pero, como se observa en el ejemplo de la figura 2, al ser una adaptación de un sistema diseñado para personas sin discapacidad para personas con discapacidad, es una herramienta mejorable, ya que algunas de las características que se ofrecen por defecto en el dispositivo dificultan la accesibilidad (como la entrada de texto mediante teclado QWERTY o la navegación entre distintas ventanas).

Figura 2: Diseño adaptativo en lugar de un diseño centrado en el usuario (UCD).

De la unión de estos dos conceptos, redes sociales y accesibilidad en iOS, nace la idea de realizar un cliente de Twitter que sea capaz de interactuar con esta red social en un smartphone (en este caso, en un iPhone con una versión de iOS 5.0 o superior) mediante unos recursos de accesibilidad diseñados específicamente para personas con cualquier discapacidad visual, abriendo de esta manera el campo de las redes sociales a un sector que según cifras de la Organización Mundial de la Salud, ronda los 285 millones de personas en todo el mundo, de los cuales 39 millones son ciegos totales [3].

1.2 Situación y problemática actual

Con la aparición de los smartphones, la forma en la que se trabaja para el desarrollo de un proyecto ha sufrido algunos cambios con respecto a la forma en la que se afrontaban nuevos productos software destinados exclusivamente a escritorio. El cambio en la forma de interactuar con los dispositivos, el tamaño y forma de los mismos, y la globalización del mercado del software mediante plataformas propietarias (Google Play, AppStore) han hecho que profesiones con un papel menos relevante en tiempos pasados, en los cuales se primaba la funcionalidad por encima de todo, sean ahora quiénes marquen el ritmo y estrategia a seguir en los procesos de desarrollo de software. La funcionalidad se presupone, y los diseñadores y los desarrolladores de experiencia de usuario (a partir de ahora, UX) son los encargados de transmitir al equipo, no sólo cómo hacer una aplicación útil para el usuario, si no también de cómo hacer una aplicación que enganche al usuario y que haga que este no utilice los desarrollos de la posible competencia.

En este sentido, y dirigido por los mercados potenciales de un producto, el diseño de dispositivos móviles y el desarrollo de aplicaciones han sido pensados para personas que no sufren ningún tipo de discapacidad que les dificulte o inhabilite para un uso estándar del producto.

Por suerte (aunque, en algunos casos, por ley) existen empresas y equipos de desarrollo cuyo nicho de mercado o actuación es el fabricar herramientas de accesibilidad que permitan un uso lo más cercano posible al normal en determinadas tecnologías para personas que sufren algún tipo de discapacidad.

En el siguiente apartado se describirán algunas de estas herramientas destinadas al uso de smartphones por personas con discapacidad visual.

1.3 Soluciones actuales

En este punto se van a describir las soluciones de accesibilidad que existen actualmente en el mercado, mostrando cuáles son sus aciertos y cuáles son sus carencias. Primero se hablará de herramientas en la plataforma Android, con el único objetivo documentar el trabajo realizado en otros sistemas, y posteriormente se describirán las herramientas disponibles en el sistema operativo iOS. No se describirá ninguna herramienta del sistema operativo Windows Phone porque carece de características específicas de accesibilidad en el momento de la redacción de este documento.

1.3.1 Android TalkBack

TalkBack [4] es una herramienta nativa de los sistemas Android a partir de la versión 4.0, que proporciona respuestas a modo de vibración y sonido como complemento de las acciones que se suceden en el dispositivo.

Tanto la vibración como el sonido se puede configurar y adaptar para diferenciar los distintos tipos de acciones (llamada entrante, notificación de una aplicación, recibo de mensajes…) Además, TalkBack proporciona información sobre dónde está pulsando el usuario en cada momento. Permite a los programadores usar sus características y posibilidades en las aplicaciones que se desarrollen.

Aunque es nativo, activar TalkBack necesita de una navegación por los ajustes del dispositivo, como se observa en la figura 3, con lo que es complicado para usuarios con discapacidad visual. La aplicación no permite la modificación de aspectos gráficos en Android que ayuden a las personas que no tienen una discapacidad visual completa a tener referencias visuales. TalkBack no realiza ningún cambio en la interacción y tampoco permite la configuración de la verbosidad (la cantidad de información en las frases de respuesta) ni la velocidad en la que se "recitan" las respuestas.

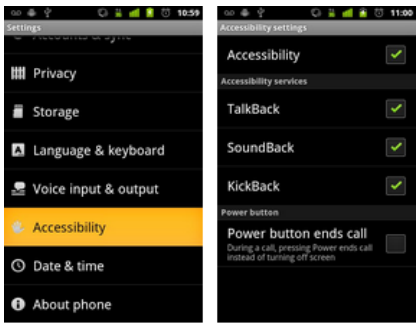


Figura 3: Acceso a la activación de la accesibilidad y TalkBack en Android.

1.3.2 VoiceOver

VoiceOver [5] es la primera solución presentada de forma nativa en dispositivos con el sistema operativo iOS a partir de la version 3.2. Esta aplicación proporciona respuestas auditivas a la interacción con el dispositivo, guiando y ayudando al usuario en la navegación por el sistema.

Figura 4: Activación de VoiceOver y guía de uso en iPad.

VoiceOver es una herramienta que introduce unas ligeras modificaciones en la interacción, que junto a la introducción de tabulaciones en el sistema, facilita la navegación para las personas que padecen alguna discapacidad visual. Tal y como vemos en la figura 4, el sistema permite su configuración para el acceso rápido al modo accesible (triple click en el botón "Home" para activar/desactivar), la velocidad del habla, el tipo de voz y cómo se leen los textos, si por frases completas, palabra por palabra o, incluso, letra a letra. Además, es completamente compatible con otras características de accesibilidad nativas de iOS como son la inversión de colores, textos grandes y zoom en las pantallas. Permite el uso y explotación de sus características por parte de los programadores a la hora de desarrollar sus aplicaciones.

VoiceOver es una buena adaptación de un sistema claramente basado en la imagen y el diseño para personas discapacitadas. A pesar de ello, es una adaptación, con lo que existen puntos mejorables. No permite la configuración de la verbosidad, con lo que se deja la responsabilidad en este sentido al programador. Los cambios en la interacción en algunos casos ayudan pero, en otros casos, pueden resultar complicados, como a la hora de introducir texto, navegar por una lista o pasar páginas del escritorio o de un documento. La inversión de colores es eso, una inversión, y no una transformación a blanco sobre negro, con lo que la gama de colores es superior, causando efectos confusos.

1.3.3 Siri

Siri [6] es el asistente inteligente presentado por Apple para sus dispositivos (a partir del iPhone 4S y iPad 2) con la versión del sistema operativo iOS 5.0 o superior. Actualmente es un sistema en versión Beta y no está disponible en castellano. Está basado en el sistema de VoiceRecognition desarrollado por Nuance Communications [7] y un sistema de inteligencia artificial desarrollado por Apple.

Figura 5: Distintos usos de Siri y su interacción con aplicaciones nativas del sistema iOS 5.0.

Siri es capaz de reconocer el habla del usuario y proporcionar respuestas asociadas a las peticiones del mismo (figura 5). El sistema se puede configurar para el acceso rápido, manteniendo pulsado el botón “Home” o con tan sólo inclinar el teléfono hacia el usuario. La aplicación interactúa con otras aplicaciones nativas del sistema, con lo que es capaz de realizar las tareas básicas del dispositivo sin necesidad de navegar ni interactuar de forma táctil con él y también se comunica con internet, con lo que puede dar información disponible en la red.

Esta herramienta ha sido una revolución en cuanto a la forma de interactuar con los dispositivos móviles. A pesar de ello, a día de hoy es una versión Beta y tiene muchas mejoras pendientes. Siri falla en muchas ocasiones en el reconocimiento del habla, con lo que en muchas ocasiones cuesta demasiado entenderse con él, sobre todo cuando las peticiones son complejas o elaboradas. Este sistema no está en todos los idiomas, con lo que dificulta aún más el entendimiento con el asistente. Tampoco se ha lanzado un framework que posibilite a los programadores de aplicaciones utilizar la herramienta, con lo que las posibilidades actuales de la herramienta son limitadas.

1.4 Objetivos del proyecto

Como hemos visto, el mercado actual ofrece buenas herramientas, pero aún están muy lejos de solucionar el problema que supone el manejo de los dispositivos móviles por personas con alguna discapacidad visual. Además, las redes sociales, usadas por tantos millones de personas en el mundo, no tienen una versión accesible de las mismas, marginando tecnológicamente a las personas que padecen este problema.

En este proyecto se propone la realización, en primer lugar, de una librería de controles accesibles para personas con alguna discapacidad visual en la plataforma iOS de Apple. Estos controles se basarán en la interacción patentada por Javier Sánchez Sierra, Joaquín Selva Roca de Togores y por mí, llamada "Controles gestuales accesibles para la navegación en dispositivos electrónicos con pantalla multitáctil". Esta patente fue elaborada después de un estudio y una investigación sobre la interacción con estos dispositivos por parte de personas que padecen una discapacidad visual, con lo que la realización de esta librería tiene como objetivo hacer accesible el acceso a la información que proporcionan las librerías de controles estándar en sistemas iOS y aumentar notablemente la experiencia de usuario por parte de este grupo de personas en dispositivos táctiles de la compañía Apple.

En segundo lugar, y tras la realización de esta librería, se propone la realización de un cliente para la plataforma Twitter que, mediante llamadas a la API pública que ofrece la red social, pueda conectar a los usuarios discapacitados con los usuarios de esta red. De esta manera se proporciona un nuevo canal de comunicación completamente nuevo a este grupo de usuarios que, mediante otras herramientas, sólo encuentran dificultades en el acceso a estas nuevas plataformas sociales.

Cumplir los objetivos de este proyecto supone eliminar la barrera existente entre la discapacidad visual y la tecnología actual, además de la creación de las bases para que futuros desarrollos tengan en cuenta a este sector de la población. También abre una nueva vía de negocio para las plataformas sociales, aumentando su audiencia potencial, y creando una imagen positiva de las mismas (muy útil para su promoción) a los posibles futuros usuarios concienciados con los problemas que la discapacidad genera.

A continuación se trata de documentar y explicar todo el proceso llevado a cabo para la consecución de este proyecto.

2. Conceptos previos

A continuación se recopilan una serie de conceptos previos que ayudan a comprender cuáles son los puntos clave a la hora de la realización de este tipo de proyectos en cuanto a la experiencia de usuario se refiere. Además, se ofrecerá un planteamiento teórico de las fases que se van a seguir para alcanzar el objetivo del proyecto.

2.1 Conceptos la experiencia de usuario

Todas las soluciones de accesibilidad presentadas en la sección anterior tienen un común denominador: son adaptaciones de sistemas concebidos para personas sin ninguna deficiencia visual. Es por ello qué estos sistemas, aunque ayudan, no solucionan el problema que plantea la discapacidad visual en cuanto al uso de estas tecnologías.

Raylight se propone entrar en el mercado de los teléfonos inteligentes mediante la realización de aplicaciones diseñadas desde el principio para personas que sufren este tipo de discapacidad. Utilizando los estándares disponibles en el mercado del desarrollo de aplicaciones para móviles (iOS y Android principalmente), y haciendo uso de algunos de los conceptos teóricos empleados en algunas de las herramientas anteriormente vistas, se busca realizar una suite de aplicaciones accesibles. Estos conceptos son:

- **Interacción:** qué los nuevos terminales inteligentes posean pantallas táctiles ha modificado la forma de interactuar con los mismos. En un ordenador, hacíamos uso del ratón y el teclado para interacción con el sistema. En los dispositivos móviles, esta interacción se basa en tocar sobre una pantalla que muestra una serie de elementos software que representan las distintas acciones y posibilidades del sistema. Esta interacción tiene una alta carga visual (navegación entre elementos del sistema, recorrer una lista…), por lo que es fundamental realizar modificaciones en la interacción que se adapten a las necesidades de los usuarios.

Figura 6: Interfaces gráficas en dispositivos Android, iOS y solución propuesta en la aplicación iLoowi de Raylight.

- **Aspecto visual:** cuando se habla de que una persona padece una discapacidad visual se tiende a pensar en la ceguera como problema. En realidad, y según la Organización Mundial de la Salud, las personas pueden ser divididas en varios grupos según su salud visual [3]. Estos grupos son visión normal, baja visión (que está dividida en dos subgrupos, discapacidad visual moderada y discapacidad visual grave), y ceguera. De los 285 millones de discapacitados visuales en todo el mundo, 246 millones pertenecen al grupo de baja visión, con lo que a la hora de desarrollar aplicaciones para estos grupos hay que tener en cuenta que la gran mayoría de usuarios sobre los que se pone el foco tienen algún resto visual que hay que intentar aprovechar para facilitar lo máximo posible el uso de estos dispositivos y mejorar su UX.

Los dispositivos móviles inteligentes hacen un uso colorista en el diseño de sus interfaces gráficas con el objetivo de llamar la atención y agradar al usuario. Para el desarrollo de aplicaciones orientadas a personas con baja visión, es muy útil reducir la gama de colores a utilizar, basándose en el diseño de interfaces oscuras que superpongan elementos y textos claros, obteniendo un alto contraste, que los usuarios con discapacidad visual son capaces de distinguir (figura 6). En algunas ocasiones, el uso de grises también es recomendado.

- **Respuesta auditiva:** por motivos obvios, la respuesta auditiva es un punto fundamental sobre el qué trabajar, ya que de ella depende que el usuario sea o no capaz de interactuar con el sistema. De esta respuesta auditiva hay que controlar varios aspectos. Las metáforas a utilizar de interacción con los controles (sonidos al entrar, salir y actuar con los controles en el área donde han sido situados), la verbosidad en los distintos controles de la aplicación (la cantidad de información que proporcionan al usuario, que indican que control es y qué acción pueden realizar), la velocidad en la que se recitan los textos o incluso la voz que narra dichos textos (mediante herramientas de Text-To-Speech) pueden provocar que el usuario tenga una gran UX o que, por el contrario, esta UX sea baja y por tanto deje de usar tu aplicación.

- **Otros aspectos de la UX:** también hay que fijarse en que el usuario siempre sepa donde está y qué está haciendo el sistema, que la información más importante sea accesible de forma rápida y eficaz, que el contenido de las interfaces gráficas no sea dinámico y que entre las distintas interfaces sea parecido, para que el usuario aprenda rápido la distribución de las mismas.

2.2 Planteamiento teórico de la solución

Una vez se conocen cuáles son los elementos claves sobre los que se tiene que trabajar, el proyecto se divide en fases. Estas fases son un estudio preliminar del problema, diseño y desarrollo de los componentes a realizar, pruebas unitarias de los componentes desarrollados, diseño y desarrollo de la aplicación, pruebas de la aplicación y distribución de la misma. A continuación se hace una descripción teórica de estas fases:

- Estudio preliminar del problema: en esta fase se propone realizar un estudio sobre cómo mejorar la experiencia de usuario en dispositivos móviles con personas con distinto grado de discapacidad visual. Para ello se realizarán unas aplicaciones sencillas que cuenten con una colección de controles que sustituyan a la colección básica de controles disponibles en cualquier plataforma (botones, sliders, etiquetas, campos de texto…). Además, en las aplicaciones realizadas para hacer las pruebas se combinan estos controles con distintas tonalidades, principalmente basadas entre blanco, negro y escala de grises. También se prueban distintas velocidades de texto, uso y longitud de la verbosidad en los textos y distintos feedbacks auditivos y vibratorios. Durante toda esta fase se han de recoger datos sobre la respuesta de los usuarios a las distintas pruebas, además de registrarlas en video.

- Diseño y desarrollo de componentes: una vez recopilados los datos se procederá a diseñar y desarrollar los componentes que van a formar parte de la librería de controles accesibles. Algunos componentes estarán basados en los mostrados en el estudio y otros se realizarán según los datos obtenidos del mismo, algunos básicos, y otros más avanzados (estos últimos son componentes basados en la combinación de componentes básicos). Los componentes diseñados y desarrollados tendrán como objetivo sustituir los componentes básicos en el desarrollo de interfaces gráficas, en este caso, en dispositivos iOS, por componentes que consigan realizar las mismas acciones qué los básicos pero de una manera accesible para las personas con discapacidad visual.

- Pruebas unitarias de los componentes realizados: después de la primera fase de desarrollo los componentes se tendrán que probar con distintos usuarios con el objetivo de certificar que el trabajo que se ha sido realizado durante el desarrollo ha sido el correcto y encaja con las conclusiones extraídas del estudio preliminar. Si en estas pruebas se detectan problemas o comportamientos inesperados, durante esta misma fase se incluye un periodo de corrección de errores para solucionar estos problemas.

- **Diseño y desarrollo de la aplicación:** una vez desarrollados todos los componentes, se plantean que aplicaciones deberían de ser desarrolladas. Además de las aplicaciones básicas, que consoliden la plataforma, mi proposición es la de desarrollar una aplicación accesible que interactúe con alguna de las redes sociales más extendidas. Las principales opciones eran Facebook, que tiene 901 millones de usuarios, o Twitter, con alrededor de 555 millones de usuarios al comienzo de 2012 [1]. Esta decisión se tomará en base a la complejidad de la posible aplicación y su posible integración en sistemas iOS.

- **Pruebas de la aplicación:** antes de distribuir la aplicación en el mercado, la aplicación tiene que pasar una serie de pruebas que den la seguridad de que la aplicación es adecuada al objetivo planteado y estable como solución software. Esta serie de pruebas serán de nuevo llevadas a cabo por usuarios con baja visión.

- **Distribución:** una vez superadas todas las anteriores fases, la aplicación se distribuirá en el AppStore (figura 7), la plataforma de distribución de aplicaciones para dispositivos iOS de Apple a nivel mundial. Gracias a este tipo de plataformas, se consigue llegar a cualquier usuario en cualquier lugar del mundo. La aplicación será traducida en al menos 3 idiomas, inglés, catalán y castellano, aunque tendrá como mínimo 4 versiones, ya que la versión en inglés será distribuida en inglés británico e inglés americano.

Figura 7: Logo del AppStore de Apple.

En los siguientes capítulos se entrará en detalle qué trabajo se realizó en cada una de estas fases y cómo se realizó. Antes de eso, se explicarán las tecnologías usadas en todo el proceso llevado a cabo para la consecución de este proyecto.

3. Herramientas, tecnologías y licencias

Durante la realización de este proyecto se han utilizado una serie de herramientas, tanto hardware como software, además de hacer uso de diversas tecnologías necesarias, no sólo para acometer el objetivo, si no también para realizar un buen proyecto software, bien documentado, escalable y con componentes reutilizables para facilitar el trabajo futuro de mantenimiento, mejora y construcción de nuevas aplicaciones. Además, también se ha requerido el uso de ciertas licencias para el uso de alguna de las herramientas. A continuación se procede a detallar primero las herramientas, más tarde las tecnologías y el capítulo terminará detallando cuáles son las licencias adquiridas y utilizadas en este proyecto.

3.1 Herramientas

Obviamente, para la realización de cualquier proyecto software se requiere de una serie de herramientas hardware, llamémosle básicas, que sirvan de plataforma para el desarrollo del proyecto. Los desarrollos en iOS y, en general, en cualquier producto de la compañía Apple, están bastante limitados a un uso de sus propios dispositivos para desarrollar para los mismos. En esta sección se van a comentar en primer lugar los dispositivos utilizados para estos desarrollos.

Por otro lado, no sólo se necesitan herramientas hardware, si no también herramientas software para la realización de los proyectos. Después de la sección de herramientas hardware se detallará cuáles han sido las herramientas software utilizadas en este proyecto.

3.1.1 Herramientas hardware

Las herramientas hardware [8] utilizadas han tenido principalmente dos funciones: plataforma dónde realizar el desarrollo y dispositivos dónde testear y probar las soluciones realizadas.

- MacBook Pro: ordenador portátil de la compañía Apple (figura 8) que cuenta con el sistema operativo Mac OS X Lion. En el mismo se instalan las diversas herramientas software necesarias para el desarrollo del proyecto. No es la máquina necesaria mínima para el desarrollo de aplicaciones para dispositivos iOS, pero por mi experiencia personal, es una buena máquina donde realizar los desarrollos.

Figura 8: MacBook Pro.

- iPhone 4S: es la última versión disponible en el mercado del archiconocido dispositivo de la marca de Cupertino. Dispone de la última versión del sistema operativo iOS, con lo que es compatible con todos los desarrollos realizados para esta plataforma.

- iPhone 4: este dispositivo es el modelo anterior al iPhone 4S. A pesar de que puede ser actualizado, cuenta con una versión anterior del sistema operativo iOS (4.2) para la realización de pruebas en versiones anteriores a iOS 5.0.

Figura 9: iPad 2 y iPhone 4S.

- iPad 2: penúltimo modelo de la tableta desarrollada por la compañía, útil para comprobar el posible uso de los desarrollos, diseñados para teléfonos, en este tipo de dispositivos.

3.1.2 Herramientas software

Los anteriores dispositivos cuentan con una serie de herramientas software, tanto para su uso y disfrute como para su utilización para el desarrollo de soluciones software. Apple es una compañía que se esfuerza mucho en qué el diseño de sus productos, tanto hardware como software, sean vistosos y atractivos para el consumidor, pero también tiene muy claro que la robustez de sus herramientas software es un factor clave para el éxito de la compañía, por lo que trabajar con sus herramientas, aunque caro, proporciona una experiencia de usuario de alto nivel.

A continuación se van a describir las herramientas software utilizadas durante el proyecto, desde los sistemas operativos de los dispositivos utilizados hasta el IDE de desarrollo. El objetivo no es detallar todas las características de estas herramientas, si no las relevantes en desarrollos de aplicaciones para dispositivos móviles, y mas concretamente, en el desarrollo de nuestra aplicación.

Figura 10: Dispositivos Mac con el sistema operativo MacOS X Lion.

- MacOS X Lion [8]**:** es el último sistema operativo para ordenadores Mac (figura 10). Permite el uso de gestos en el trackpad para la interacción con el sistema, en un claro acercamiento hacia un sistema común para dispositivos táctiles y ordenadores. Sobre él se instalarán las herramientas software sobre las que se va a diseñar y desarrollar el proyecto.

- iOS 5.0 [8]**:** el sistema operativo de los dispositivos móviles de Apple. Este sistema tiene siempre el control por encima de las aplicaciones, con lo que a la hora de desarrollar aplicaciones impone algunas restricciones en cuanto al uso de funcionalidades del sistema. Permite la gestión de eventos referentes a la pantalla táctil y los gestos multitáctiles. Además, cuenta con la integración de la plataforma Twitter, con lo que es relativamente sencillo realizar aplicaciones que interactúen con la red social.

- XCode 4.3.1 [9]**:** es el IDE (Integrated Development Environment) para el desarrollo de aplicaciones para el sistema operativo iOS y MacOS. Tal y como vemos en la figura 11, cuenta con una interfaz limpia y cuidada, proporciona las herramientas para desarrollar aplicaciones en estos sistemas. Cuenta con Interface Builder, un editor gráfico de interfaces gráficas muy potente, pero qué no utilizaremos en el desarrollo de este proyecto al utilizar componentes gráficos propios. El compilador Apple LLVM entiende C, C++ y Objective-C y es capaz de compilar código hasta dos veces más rápido que GCC. Además, cuenta con un potente motor de debug de las aplicaciones, tanto en el ordenador como en los dispositivos, y con un conjunto de herramientas, conocidas como Instruments, para monitorizar las aplicaciones y sacarles el mayor rendimiento posible a las mismas.

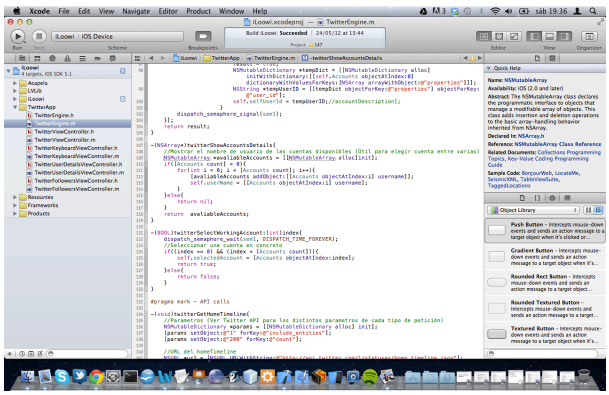


Figura 11: Entorno de desarrollo XCode 4.3.1.

- Otras: hay otras herramientas utilizadas para la realización del proyecto que no han sido usadas por el equipo de desarrollo y que merecen ser nombradas aquí. Cornerstone ha sido utilizado como cliente para realizar el proyecto utilizando un control de versiones con el sistema SVN. Por otro lado, los diseños gráficos han sido diseñados con Photoshop.

3.2 Tecnologías

Durante el proceso de desarrollo de este proyecto se han utilizado diversas tecnologías que han hecho posible la consecución del mismo. Los paradigmas de programación, el lenguaje de programación, los frameworks de desarrollo, los patrones de diseño o la arquitectura de desarrollo de aplicaciones promovida para las aplicaciones para dispositivos iOS van a ser descritas para comprender un poco mejor el proceso llevado a cabo. Todas estas herramientas irán apareciendo de manera incremental, por lo que cada tecnología aquí mostrada no podrá entenderse sin las descritas anteriormente.

3.2.1 Paradigma de programación orientada a objetos

El paradigma de programación orientada a objetos [10] o POO es un paradigma de programación por el cual todas las entidades que forman un sistema se consideran objetos. En cierto estado de la ejecución de un programa, estos objetos se encontrarán con un determinado estado (datos con una serie de valores), poseerán un determinado comportamiento (métodos) y contarán con una determinada identidad que los diferencian del resto de objetos. El origen de este paradigma se remonta a finales de los años 60, cuando Ole Johan Dahl y Kristen Nygaard crearon el lenguaje de programación Simula67, un lenguaje diseñado para hacer simulaciones.

Aunque su primera aparición fue en el año 1967, fue a mediados de los años 80 cuando se popularizó gracias a la aparición del lenguaje de programación C++, una extensión orientada a objetos del leguaje C, creado por Dennis Ritchie [11].

Los conceptos mas importantes que caracterizan el paradigma de programación orientada a objetos son los siguientes:

- **Abstracción:** consiste en aislar un elemento de su contexto o del resto de elementos que lo acompañan. El objetivo es mostrar qué hace un objeto sin mostrar como lo hace.

- **Encapsulamiento:** agrupación de todos los componentes que puedan componer una misma entidad en el mismo nivel de abstracción. Este concepto permite controlar el comportamiento de los objetos, ya que los datos pertenecientes a estos sólo pueden ser cambiados mediante las operaciones definidas para ese objeto.

- Herencia: el mecanismo de herencia es aquél por el cual se crean nuevas clases partiendo de clases ya existentes, sumándole a las funcionalidades que ya posee la clase padre, ciertas funcionalidades y comportamientos extra definidos en la clase hija. La herencia fomenta la jerarquía de clases y es un requisito fundamental para el empleo del polimorfismo.

- Polimorfismo: es la capacidad de enviar mensajes a grupos de objetos distintos, con el único prerrequisito de saber contestar a estos mensajes. Para ello, una clase padre crea una interfaz por la cual las clases hijas tengan que definir individualmente los métodos descritos en esta interfaz, teniendo para cada clase un comportamiento que no tiene porque ser igual. El polimorfismo no es igual a la sobrecarga de métodos, ya que esta última se da siempre dentro de una misma clase y no entre clases. Además, el polimorfismo se resuelve en tiempo de ejecución y la sobrecarga se resuelve en tiempo de compilación.

Estos son los principales conceptos que definen el paradigma de programación orientada a objetos. Son muchos los lenguajes de programación que hace uso de este paradigma. A continuación se hablará del utilizado durante el desarrollo de este proyecto.

3.2.2 Lenguaje de programación Objective-C

Objective-C [12] es un pequeño pero poderoso conjunto de extensiones al lenguaje estándar ANSI C. Este conjunto de extensiones están basados en el lenguaje de programación SmallTalk, uno de los primeros lenguajes de programación orientada a objetos. Fue diseñado por Brad Cox en el año 1980 y en 1988 fue adoptado como lenguaje de programación de NEXTSTEP, sistema operativo diseñado por NeXT Computer Inc., empresa que fue absorbida por Apple en 1997, y qué fue el precursor del sistema operativo MacOS X.

El lenguaje tiene como objetivo proporcionar, de una manera sencilla y poderosa, capacidades de la orientación a objetos al lenguaje C. Estas capacidades hacen que todo el desarrollo de aplicaciones en este lenguaje este centrado en los objetos, identificados mediante el tipo de datos `id`. Se basa en el paso de mensajes a instancias de objetos, de la manera `[objeto método:parámetro];`. Al igual que C++, las clases suelen estar separadas en un archivo de cabecera, qué definirá la interfaz de dicha clase, cuya extensión es `.h`, y un archivo de implementación, denotado con la extensión `.m` (figura 12).

```
//
//  UAPelicula.m
//  Filmoteca
//
//  Created by Héctor Dario Arroyo Gómez on 09/12/11.
//  Copyright (c) 2011 __MyCompanyName__. All rights reserved.
//

#import "UAPelicula.h"


@implementation UAPelicula
@synthesize titulo = _titulo;
@synthesize director = _director;
@synthesize edad = _edad;
@synthesize fecha = _fecha;
@synthesize puntuacion = _puntuacion;
@synthesize actores = _actores;
@synthesize portada = _portada;


#pragma mark - inicializadores

-(id) init{
    if (self = [super init]) {
        self = [self initWithTitle:@"Sin titulo"];
    }
    return self;
}

-(id) initWithTitle:(NSString *)newTitulo{
    if (self = [super init]) {
        NSCalendar  *calendario =   [NSCalendar currentCalendar];
        NSDateComponents    *componentes    =   [[NSDateComponents  alloc]  init];
        [componentes    setDay: 01];
        [componentes    setMonth:01];
        [componentes    setYear:1970];
        NSDate  *fecha  =   [calendario dateFromComponents: componentes];
        self = [self initWithTitle:newTitulo director:@"Desconocido" ageRating:TP mark:-1
            withDateOfRelease:fecha];
    }
    return self;
}
```

Figura 12: Implementación de una clase con sus incializadores en Objective-C.

Otra de las peculiaridades del lenguaje es la gestión de memoria. En la última revisión del lenguaje se ha incluido un mecanismo llamado ARC (Automatic Reference Counting) por el cual esta gestión de memoria es realizada automáticamente por el compilador. Para la realización de aplicaciones para el sistema operativo iOS, esta versión esta disponible en la versión 5.0 y posteriores del sistema, con lo que si se desea que la aplicación pueda funcionar en dispositivos con versiones anteriores del sistema operativo, la gestión de memoria se ha de realizar de manera manual. Los objetos tienen que ser retenidos (`[objeto retain]`) y liberados (`[objeto release]`) según las necesidades de la aplicación y de manera controlada, ya que de otra manera se pueden producir problemas por filtraciones de memoria (uso descontrolado e innecesario de memoria) que pueden provocar problemas de rendimiento de las aplicaciones e incluso cierres inesperados de las mismas.

Objective-C es la base sobre la que se construyen las aplicaciones para dispositivos de la compañía Apple, sean dispositivos móviles con el sistema operativo iOS, como ordenadores con el sistema operativo MacOS X. Además, una vez conocida la sintaxis y las características especiales del lenguaje, es un lenguaje que ofrece al programador un gran control sobre el comportamiento de las aplicaciones.

3.2.3 Patrón de diseño de aplicaciones Model-View-Controller

En el desarrollo de aplicaciones para dispositivos con el sistema iOS el patrón o la arquitectura de software que se utiliza es la conocida como Model-View-Controller [13] (a partir de ahora, MVC). Este patrón divide las aplicaciones en conjuntos de MVCs que se comunican entre sí (figura 14) para realizar las diversas tareas que la aplicación requiera en un determinado momento.

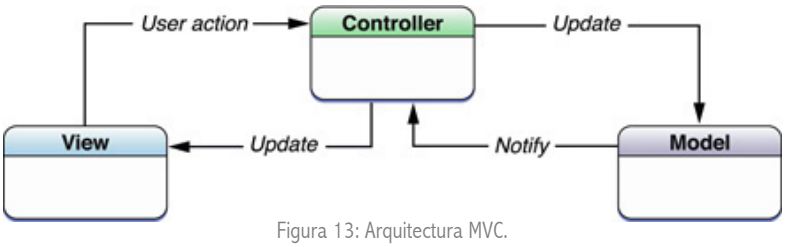


Figura 13: Arquitectura MVC.

Como vemos en la figura 13, los objetos toman uno de los siguientes roles:

- **Modelo:** los objetos cuyo rol es el de ser modelo son los encargados de encapsular los datos de una aplicación y su comportamiento básico. Estos objetos definen la lógica y los procesos que manipulan dichos datos. Estos objetos no se comunicarán directamente con la vista, si no que serán los controladores los que se comuniquen con los modelos para la creación o manipulación de los datos del modelo, y estos notificarán a los primeros cuando se finalicen las distintas tareas.

- **Vista:** los objetos que adquieren el rol de vistas son aquellos objetos que el usuario ve en las aplicaciones. Estos objetos conocen cómo mostrarse visualmente en los dispositivos y cómo actuar en respuesta a los diversos eventos provocados por la interacción con el usuario. Muestran datos, pero no se comunican directamente con los objetos modelo, si no que se comunican con un intermediario, qué es un objeto de tipo controlador. Son configurables, y son los responsables de generar consistencia visual entre aplicaciones, por lo que son elementos fundamentales en el diseño de una buena UX.

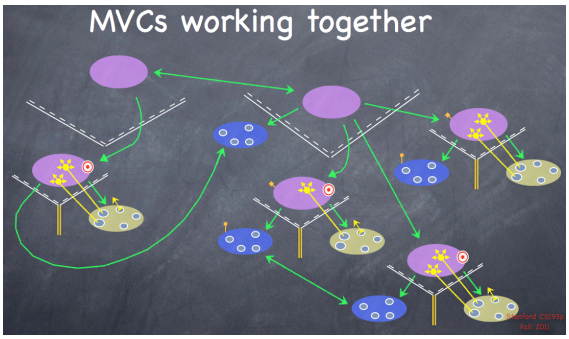


Figura 14: Diversos conjuntos MVC y su comunicación. Los modelos están representados de azul, los controladores están representadas de morado y las vistas de gris.

- **Controlador:** son los objetos encargados de la comunicación entre la vista y los modelos. Para ello, interpreta los datos recibidos desde los objetos vista, hace peticiones a los objetos modelo con los datos interpretados y comunica los resultados a la vista de una u otra manera.

Gracias a este patrón se fomenta la reutilización de componentes de una aplicación y se proporcionan interfaces mejor definidas. Además, los cambios ha realizar son mucho más sencillos, gracias a la independencia de los componentes debido a la separación por capas (por ejemplo, un cambio en la vista no afecta al modelo).

Estos componentes del patrón se pueden combinar por diversas razones, pero de una manera lógica. Esa ha sido la estrategia en el diseño de nuestra aplicación, en la cuál se han combinado las vistas con los controladores, ya que las primeras no podían ser diseñadas de forma gráfica, si no que todas las interfaces han sido programadas.

3.2.4 iOS SDK

iOS Software Development Kit [14] (iOS SDK) es el kit de desarrollo de aplicaciones para los dispositivos con el sistema iOS. Provee al programador de todas las herramientas, compiladores y frameworks necesarios para el desarrollo de aplicaciones. Además, incluye las herramientas necesarias para testear las aplicaciones aunque no se disponga de dispositivos físicos.

La primera versión beta de este kit de desarrollo fue liberada para el uso por desarrolladores en marzo de 2008 y ha tenido múltiples revisiones desde entonces. Actualmente, la última versión es la 5.1, que corresponde también a la última versión del sistema operativo.

Como todos los productos de desarrollo para plataformas de Apple, y en aras de tener siempre el control sobre las aplicaciones, esta SDK proporciona ciertos límites en cuanto a la programación se refiere. Por ejemplo, si se desea enviar un SMS, no existe o no es pública una clase o un método que envíe directamente ese SMS, si no que, con las herramientas del iOS SDK, el usuario tiene que abrir el diálogo de envío de mensajes estándar de la plataformas, escribir ahí el mensaje y enviarlo desde ese mismo diálogo. En posteriores secciones se verá cómo ha afectado esta limitación en el desarrollo de este proyecto.

Las novedades más importantes de la última versión de esta SDK son la posibilidad de integrar las aplicaciones con iCloud, el servicio en la nube de Apple, un nuevo y moderno servicio de notificaciones, la posibilidad de diseñar el flujo de navegación de las vistas que forman parte de la aplicación mediante Storyboard y la integración con la red social Twitter, este último muy útil para nuestro proyecto.

3.2.5 Frameworks

Los frameworks [15] son aquellos conjuntos de clases o librerías de clases necesarios o útiles para el desarrollo de determinada aplicación.

El iOS SDK proporciona una colección muy amplia de frameworks que dan la posibilidad al programador hacer uso de todas las posibilidades que los dispositivos proporcionan. Por otro lado, y por razones obvias, sería completamente desproporcionado y desmedido incluir en una aplicación todos los frameworks disponibles para una aplicación que probablemente sólo haga uso de un par de ellos.

Los frameworks más importantes utilizados en el desarrollo de este proyecto son los siguientes:

- UIKit: es el framework con los componentes básicos para la interfaz de usuario. Aunque nosotros hemos creado nuestros propios controles, estos descienden de la clase LVControl (se hablará de todo ello más adelante, en el capítulo de desarrollo), que a su vez desciende de la clase UIControl, disponible en UIKit. Además, en muchos momentos se hace uso de algunos componentes de este framework para apoyar a los componentes realizados (por ejemplo, los iconos de los controles son del tipo UIImage).

- AudioToolbox: framework que contiene las clases necesarias para la gestión de de tareas relacionadas con el audio en dispositivos iOS. Entre los recursos disponibles en este framework se encuentra el System Sound Service, el cual proporciona una interfaz en C para reproducir sonidos de duración menor a 30 segundos y para provocar el efecto de vibración del teléfono. Gracias a su ayuda hemos logrado crear los efectos de sonido al entrar o salir de un control y también hemos habilitado la vibración del teléfono cuando un usuario se encuentra en el centro del teclado diseñado, de forma que tenga una referencia de dónde se encuentra la zona central del mismo.

- Accounts: este framework tiene las clases encargadas de la gestión y el manejo de diversos tipos de cuentas de usuario en dispositivos iOS. Todas las cuentas utilizadas para acceder a diversos servicios están ubicadas en un almacén de cuentas que permite la obtención y el acceso a ellas de forma sencilla y segura, ya que siempre que una aplicación quiere hacer uso de estas cuentas se muestra un dialogo modal solicitando permiso para el acceso a los datos de la cuenta. En nuestro caso, combinado con el framework Twitter qué describiremos a continuación, permite acceder a la plataforma de forma sencilla y evitando la complejidad que supone programar el acceso a cuentas protegidas mediante OAuth (protocolo de acceso a servicios de suscripción de forma segura y controlada).

- Twitter: es el encargado de todas las gestiones con la red social Twitter. Este framework, en combinación con los datos obtenidos por las clases del framework Accounts, encapsula las peticiones a la API de servicios REST (descritas en la siguiente sección) que Twitter pone a disposición de los programadores (y que veremos más a fondo en la próxima sección). De una forma muy sencilla se puede hacer peticiones a esta API, añadirle parámetros en forma de diccionario o pares clave-valor y obtener una respuesta en formatos XML o JSON (hay más formatos, pero estos son los más interesantes para nuestro proyecto). Aunque no se usan en este proyecto, proporciona también una serie de clases gracias a las cuales cualquier aplicación es capaz de enviar tweets con apenas un par de líneas de código.

Estos son los principales frameworks utilizados en el desarrollo de la aplicación, aunque también se han utilizado otros, como Foundation, qué encapsula los objetos básicos comunes en todas las aplicaciones hechas para iOS o MacOS X (objetos de tipo NSArray, NSDictionary y muchos otros), o CoreImage, framework para el tratamiento (qué no el uso) de imágenes.

3.2.6 API de servicios REST de Twitter

Representational State Transfer o REST [16] es una técnica para la arquitectura de software en sistemas basados en la web. Estos sistemas describen una interfaz que combina documentos XML o JSON (se hablará de este tipo de documentos en la siguiente sección) con el protocolo http sin las abstracciones adicionales que proporcionan otros protocolos de servicios web, como SOAP.

Twitter pone a disposición de los programadores una API (Application Programming Interface, es el conjunto de instrucciones que sirve de abstracción para solicitar servicios a un sistema) para la creación de aplicaciones basadas o relacionadas con los servicios que Twitter ofrece. Esta API recibe peticiones del tipo GET (solicitar datos) o POST (enviar datos) mas la URL de la petición (con el formato de la respuesta) y un conjunto de pares clave-valor con diversos parámetros y ofrece respuestas asociadas a la petición realizada. Por ejemplo, para solicitar qué se devuelva un archivo JSON con los últimos 200 tweets del usuario @HecArroyoGomez, la petición a realizar sería:

```
GET
https://api.twitter.com/1/statuses/user_timeline.json?screen_name=HecA
rroyoGomez&count=200
```

Este tipo de peticiones tienen qué estar firmadas o autentificadas. Gracias a la integración de la última versión de iOS con el servicio Twitter, esta operación es muy sencilla y no supone mayor problema.

La petición daría como respuesta un documento JSON, con un error si no ha podido obtener el documento solicitado o con un diccionario de pares clave-valor con toda la información de los 200 tweets, no sólo el texto de los mismos, si no también si ha sido retwiteado, cuando se twiteo y otra serie de datos de interés acerca de los tweets. En el capítulo de desarrollo se explicará con mayor detalles las peticiones realizadas al servicio Twitter en este proyecto.

3.2.7 JSON

JSON [17] (JavaScript Object Notation) es un formato ligero de texto de intercambio de datos. Es independiente del lenguaje, aunque con utiliza convenciones muy similares a algunas presentes en lenguajes de la familia de C, por lo que es fácilmente entendible por los programadores.

Las estructuras que conforman un documento JSON son una colección de pares clave-valor (diccionario) y listas ordenadas de valores. Estos dos tipos de estructuras están presentes de una u otra forma en todos los lenguajes de programación, por lo que se puede usar como fuente de datos en cualquiera de ellos sin importar cuál sea.

La API de Twitter puede devolver, a petición del usuario, una respuesta con formato JSON, tal y como vemos en la figura 15. Esta respuesta será *parseada* en la aplicación (transformada a un formato entendible por la aplicación) para su uso y manejo según sea necesario.

Además de su universalidad, algunas de las ventajas de este formato son qué las operaciones realizadas sobre él son muy veloces (más que las operaciones sobre XML) ya que no es un formato estructurado, sólo son datos estructurados. Por la misma razón, suelen ser documentos menos pesados que los documentos XML.


```
[
  {
    "name": "Twitter",
    "id_str": "783214",
    "id": 783214,
    "connections": [
      "following",
      "followed_by"
    ],
    "screen_name": "twitter"
  },
  {
    "name": "Twitter API",
    "id_str": "6253282",
    "id": 6253282,
    "connections": [
      "following",
      "followed_by"
    ],
    "screen_name": "twitterapi"
  },
  {
    "name": "Twitter Engineering",
    "id_str": "6844292",
    "id": 6844292,
    "connections": [
      "following"
    ],
    "screen_name": "TwitterEng"
  }
]
```


Figura 15: JSON respuesta a una petición a la API de servicios REST de Twitter.

3.2.8 Text-To-Speech

Uno de los puntos clave del desarrollo de este proyecto ha sido la utilización de la librería de Text-To-Speech (TTS a partir de ahora) desarrollada por la compañía Acapela.

TTS hace referencia a la tecnología que transforma el texto escrito a texto hablado. El proceso inverso se conoce como VoiceRecognition, y combinados pueden ofrecer, junto a otras tecnologías, herramientas como Siri en los dispositivos iOS.

A diferencia de otras plataformas, y por motivos comerciales (Apple quiere diseñar su herramienta y no competir con terceras herramientas en sus dispositivos), no existe ningún framework que proporcione estos servicios de forma nativa, con lo que hay que utilizar recursos propios o de terceros. En este contexto, la compañía Acapela ofrece una librería para dispositivos iOS capaz de "leer" los textos en múltiples idiomas y de forma nativa. No es la única compañía que ofrece estos servicios, pero fue la elegida para el desarrollo de este proyecto por no necesitar conexión a la red para realizar las transformaciones y por su relación calidad/precio.

Gracias a esta tecnología, todos los componentes de la interfaz del proyecto y todas las acciones que se ejecutan y que afectan a la aplicación proporcionan información a modo de texto hablado, de tal manera que el usuario pueda conocer en todo momento dónde está, qué está haciendo él, qué está haciendo la aplicación y, además, conocer la información que necesite sin necesidad de tener una referencia visual de la misma.

3.2.9 UML

En este documento, para apoyar las especificaciones que se van a definir en el capítulo en el qué se analizará el proyecto, se va a hacer uso de varios diagramas definidos por el Unified Modeling Language o UML [18]. Este lenguaje de modelado, respaldado por Object Management Group, es el estándar para la realización de diagramas y documentes gráficos para la documentación de sistemas software.

Existen diversos tipos de modelos que forman parte de este lenguaje. Para este proyecto utilizaremos los siguientes diagramas:

- Diagrama de casos de uso: en él se mostrará quién usará el sistema (actores), para qué lo usarán (funcionalidad) y las distintas relaciones entre los actores que interactúan con la aplicación.

- Diagrama de clases: este diagrama mostrará cómo son las relaciones entre distintas clases, cuáles son sus atributos, cuáles son sus operaciones y que tipo de visibilidad tienen cada uno de estos componentes.
- Diagrama de actividades: con este diagrama veremos claramente cómo será el flujo de actividades de la aplicación según se cumplan o no diversas situaciones que afectan al sistema.

3.3 Licencias

Para el desarrollo de este proyecto se han tenido que usar algunas herramientas desarrolladas por terceros que necesitan de licencia para poder ser usadas. Además, para poder probar las aplicaciones en dispositivos físicos de Apple y para su publicación en el AppStore también es necesario contar con una licencia proporcionada por la compañía. A continuación se describen las licencias que han sido necesarias en este proyecto.

3.3.1 Licencia de desarrollador iOS de Apple

Apple ofrece varios tipos de licencias para desarrolladores [19] dependiendo de las necesidades del desarrollador o desarrolladores que vayan a hacer uso de esa licencia. Estas licencias, válidas para el año 2012, son: iOS Developer Program, para desarrolladores o equipos, con un precio de 80 € por año; iOS Enterprise Program, orientado a empresas, con un precio de 240 € al año; y un programa dedicado a universitarios, el iOS University Program, qué es gratuito, pero que tiene que ser adquirido por docentes de algún centro universitario.

Estas licencias proporcionan el acceso a una serie de recursos de documentación y de video para apoyar al programador y guiarle en el desarrollo de aplicaciones. Estos recursos son iguales para los tres tipos de licencia. Las variaciones en los programas de desarrolladores se basan en la cantidad de dispositivos que se pueden asociar a la licencia y hacia quién están dirigidas las aplicaciones que se van a desarrollar. En la licencia iOS Developer Program, se pueden asociar hasta 100 dispositivos para probar sobre ellos y las aplicaciones que se creen están destinadas a ser publicadas en el AppStore. En el iOS University Program se pueden crear hasta 200 perfiles de usuario con su propio dispositivo y el fin de las aplicaciones tiene que ser educacional. El iOS Enterprise Program tiene como objetivo el desarrollo de aplicaciones que van a ser utilizadas de forma interna en una compañía.

En este proyecto se ha utilizado la licencia iOS Developer Program, se han creado 2 perfiles de programadores y se han asociado 5 dispositivos con los cuales se han realizado diversas pruebas.

3.3.2 Licencia para el uso de la librería de TTS de Acapela

Para el uso de la librería de TTS de Acapela [20], además del pago de 250€ para la descarga y utilización del SDK, hay que contratar una licencia de uso, dependiente del número de descargas que se realicen de la aplicación desarrollada. Como las cantidades y los porcentajes a pagar son muy variables, y no es el objetivo de este documento el de realizar un *business plan*, no se detallarán mucho más estas cifras, pero a modo orientativo, para una aplicación con una gran cantidad de descargas la tasa a pagar puede ser de entre el 20% y el 30% de los ingresos de la aplicación.

Esta licencia también proporciona el acceso a los recursos de documentación proporcionado por la compañía para el desarrollo de aplicaciones que hagan uso de su SDK.

3.3.3 Licencia de desarrollador de Twitter

Para acceder a los recursos disponibles para desarrolladores de la plataforma Twitter hay que registrarse como desarrollador de la plataforma [21]. A partir de ese momento se podrán desarrollar aplicaciones para cualquier sistema haciendo uso de la API que Twitter proporciona.

Gracias a la integración de Twitter en el sistema iOS 5.0 no es necesario qué registremos la aplicación a desarrollar en la plataforma para obtener su API key que habilite a la aplicación autentificarse por medio del protocolo OAuth. Sin embargo, sí que son útiles todos los recursos de documentación y acceso a foros con los que resolver cualquier tipo de problemas a la hora de programar e implementar la plataforma dentro de las aplicaciones.

4. Especificación de requisitos

En ingeniería de software, la especificación de requisitos es un paso fundamental previo al desarrollo que ayuda a entender y a comprender el problema que se plantea. Es un documento (en este caso, un capítulo del documento) en el cuál se va define el proyecto y sus requisitos a modo de información y contrato entre el cliente y el equipo de desarrollo.

4.1 Introducción

Para la realización de esta especificación de requisitos vamos a seguir la organización definida por un estándar en la elaboración de este tipo de documentos conocido como IEEE 830 [22] y qué está reconocido internacionalmente.

4.1.1 Propósito

El planteamiento de este proyecto, un cliente de Twitter para dispositivos Apple enfocados a personas que padezcan alguna discapacidad visual, surge de la falta de productos en los nuevos dispositivos móviles que proporcionen herramientas de accesibilidad para interaccionar con las nuevas plataformas sociales existentes en Internet. Esta aplicación estará destinada a usuarios que padezcan un grado de discapacidad lo suficientemente alto como para necesitar herramientas especiales para el uso de dispositivos móviles (cambios de contraste, magnificadores, herramientas de audio y/o vibración…) y usuarios con ceguera. Las herramientas, tecnologías y licencias a utilizar han sido descritas en el capítulo anterior.

Para la realización de este proyecto se seguirán una serie de fases muy claramente diferenciables. En primer lugar, se realizará un estudio con usuarios discapacitados sobre unos componentes básicos y mínimos para que, según el feedback recibido, se avance a una segunda fase de desarrollo de una biblioteca de componentes accesibles. Una vez finalizados, estos componentes pasarán una serie de pruebas para comprobar que el trabajo realizado ha sido acertado y cumple con el objetivo propuesto. Cuando esta fase este validada, se procederá al desarrollo del cliente de Twitter usando los componentes desarrollados. Por último, se realizarán pruebas de uso y se liberará la aplicación para su distribución en el AppStore.

4.1.2 Ámbito del sistema

El sistema está claramente orientado a un sector concreto de la población, tanto local como mundial, ya que es un proyecto con carácter internacional, qué sufre o padece algún tipo de discapacidad, le interesa la tecnología y los nuevos dispositivos y quiere estar conectado con lo que sucede a su alrededor y en cualquier parte mediante la información que proporcionan las redes sociales.

El beneficio que se busca obtener con este proyecto afecta a diversos actores: los usuarios, los principales afectados, que obtendrán una nueva forma de comunicación y una plataforma que les permita interactuar con millones de personas; la red social Twitter, que abrirá su mercado a un nuevo sector de la población de un tamaño potencial de más de la mitad de su alcance actual; y, obviamente, la empresa Raylight, que comercializará esta herramienta.

Para maximizar los beneficios anteriormente descritos, lo más importante es garantizar un UX de alta calidad a los usuarios a los cuales esta destinado este desarrollo.

4.1.3 Definiciones, acrónimos y abreviaturas

UX: experiencia de usuario. Define el grado de satisfacción y la calidad que la herramienta ofrece en su uso y disfrute por parte de los usuarios.

Deficiencia visual: pérdida parcial de las habilidades visuales por parte de una persona.

Ceguera: pérdida total de las capacidades visuales por parte de una persona.

Twitter: red social de microblogging. http://www.twitter.com

Tweet: entrada de blog de 140 caracteres en la red social Twitter.

Retweet: referencia en el timeline personal a tweets publicados por otras personas.

Timeline: línea temporal en la que se representan los últimos tweets de uno o varios usuarios.

API: Application Programming Interface, son un conjunto de instrucciones que proporcionan los mecanismos necesarios para solicitar servicios a un sistema.

REST: arquitectura de sistemas software basados en la web que sistemas describen una interfaz de documentos XML o JSON con el protocolo http.

Localización de textos: estrategia de programación por la cual los textos se disponen en archivos aparte en los cuales, en cada uno de ellos, están los textos en un determinado idioma, y por el cual el dispositivo trata de seleccionar el idioma en función de la configuración del dispositivo, o selecciona uno de ellos por defecto.

4.1.4 Referencias

Para esta especificación de requisitos se han utilizado el tema 4 de los apuntes de la asignatura "Ingeniería del Software de Gestión" de la Universidad Politécnica de Valencia, del curso académico 2010-2011, otros proyectos anteriormente realizados para diversos tipos de consultas y una guía del IEEE 830 para la Especificación de Requisitos Software publicada por el profesor de la Universidad Complutense de Madrid Gonzalo Méndez en el curso académico 2008-2009 [22].

El resto de referencias y la bibliografía utilizada en la redacción de este documento serán presentadas en una sección aparte al final del documento.

4.1.5 Visión global

Este documento trata de explicar los procesos llevados a cabo para la consecución del proyecto abordado.

En primer lugar se ha intentado dar una visión general del proyecto, qué herramientas, tecnologías y licencias se van a utilizar y cuáles son los aspectos a tener en cuenta a la hora de afrontar un proyecto de este calibre. Después de esta especificación de requisitos se abordará el capítulo donde se analizará el proyecto y que se verá apoyado por distintos diagramas definidos en el lenguaje de modelado estándar UML. Por último, se describirá todo el proceso de desarrollo llevado a cabo, se proporcionará un manual de usuario y se comentarán diversas mejoras a desarrollar en el futuro y las conclusiones extraídas de la realización de este proyecto.

4.2 Descripción general

En esta sección procederemos a detallar las características de nuestro producto para que sean entendidas de una manera mas sencilla.

4.2.1 Perspectiva del producto

Este proyecto tiene como objetivo mejorar herramientas ya existentes, como VoiceOver, para proporcionar una UX de alta calidad a los usuarios que sufran algún tipo de discapacidad visual.

En el desarrollo de este proyecto se busca crear una librería de controles accesibles para personas con discapacidad visual y que estos controles sean reutilizables en futuras aplicaciones. Además, también se pretende crear una aplicación que permita realizar las tareas más importantes disponibles en la red social Twitter, como son visualizar el timeline propio y de terceros, enviar tweets y retweets, ver los detalles de los usuarios o seguir a otros usuarios.

4.2.2 Funciones del producto

En cuanto a la interfaz, el producto contará con las siguientes funcionalidades:

- Todas las interfaces tienen que ofrecer la posibilidad de volver a la interfaz anterior, excepto la interfaz que sea asumida como la interfaz principal.
- En todo momento, el usuario debe saber qué está haciendo, qué información tiene disponible, qué puede hacer, y donde está. Esto se conseguirá informando con sonido y voz en respuesta a la interacción del usuario con la pantalla táctil.

Por otro lado, en el contexto de la aplicación, el usuario dispondrá de la siguiente funcionalidad:

- Debido a las limitaciones que la plataforma impone, el usuario deberá estar registrado con anterioridad en la plataforma Twitter y tener asociada su cuenta al teléfono donde va a usar la aplicación. La aplicación le permitirá acceder a la plataforma con su cuenta de usuario.
- El usuario podrá comprobar los últimos tweets enviados por los usuarios que sigue desde su propio timeline.
- El usuario podrá visitar el timeline de otros usuarios.
- El usuario podrá enviar tweets, podrá retwittear tweets de otros usuarios.
- El usuario podrá contestar los tweets de otros usuarios.
- El usuario podrá ver sus detalles, incluyendo su avatar, y el de otros usuarios de la plataforma.
- Se podrán enviar mensajes directos (privados) si la relación entre usuarios es bidireccional.

- Se podrá comprobar la lista de usuarios que siguen al usuario o que son seguidos por el usuario. Además, también se podrá ver los mismos tipos de relaciones de otros usuarios.
- Se podrá seguir o dejar de seguir a otros usuarios de la plataforma.

4.2.3 Características del usuario

El perfil de usuario está bastante definido: usuarios con alguna discapacidad visual, que sea curioso tecnológicamente hablando, y que le guste mantener relaciones y estar informado por medio de las plataformas sociales disponibles en internet. No se espera que sean usuarios experto de dispositivos móviles, pues el objetivo es realizar una plataforma sencilla e intuitiva. Tampoco se espera que sean usuarios expertos de la red social Twitter, ya que toda la interacción con la plataforma estará explicada en los diversos cuadros de información disponibles en cada interfaz.

4.2.4 Restricciones generales

Para la realización de este proyecto, se han detectado las siguientes restricciones:

- El lenguaje de programación con el que desarrollar este proyecto es Objective-C, el normal en desarrollos para dispositivos Apple.
- La interacción con la red social Twitter se realizará mediante peticiones a la API de servicios REST que la plataforma facilita a los desarrolladores.
- Debido a limitaciones de esta API pública de la plataforma Twitter, no se realizarán las tareas de registro del usuario en la plataforma.
- Debido a las limitaciones impuestas por la compañía Apple en el desarrollo de aplicaciones para sus dispositivos, puede existir algún componente que no sea accesible.
- En aras de la sencillez y la usabilidad de la aplicación, esta aplicación no incluirá la visualización de tweets favoritos, visualización de “trending topics”, proposición de sugerencias para seguir a nuevos usuarios ni la lectura de mensajes directos (aunque sí que se permitirá el envio de los mismos.)
- Por motivos de marketing, todos los textos estarán localizados, al menos, en inglés, catalán y castellano.

4.2.5 Supuestos y dependencias

La aplicación está inicialmente pensada para dispositivos iPhone con el sistema operativo iOS 5.0 o superior. El usuario tiene que contar con conexión wifi o 3G en su dispositivo. Pueden existir problemas de disponibilidad de la plataforma, externos a la aplicación. La información vendrá dada en el idioma en el que el usuario tenga configurado su dispositivo, o, en su defecto, en inglés, por lo que los tweets que el usuario quiera "leer" se recitarán en el idioma en el que se encuentre la aplicación.

4.2.6 Requisitos futuros

En cuanto a la librería de controles accesibles, en un futuro se deberá plantear la inclusión de comandos controlados por voz. En futuras mejoras, la aplicación podrá sufrir cambios en la interfaz dependiendo de la respuesta de los usuarios y sus propuestas. También podrán implementarse más funcionalidades ofrecidas por la red social Twitter. Además, se puede plantear el traslado del proyecto a otro tipo de dispositivos iOS o a otras plataformas como Android o WindowsPhone.

En un futuro también se planteará la distribución de la aplicación en más idiomas.

4.3 Requisitos específicos

A continuación se procede a detallar los requisitos que se deben cumplir durante el desarrollo de este proyecto.

4.3.1 Interfaces externas

La interfaz de usuario tiene que estar orientado a un sistema táctil. Para ello, todos los controles básicos disponibles en las interfaces gráficas estarán representados de una u otra manera con los nuevos controles accesibles. Además, por el perfil de los usuarios a los que está destinada esta aplicación, tiene que ser un sistema sencillo e intuitivo. Gráficamente, las interfaces tienen que estar basadas en la combinación de colores de blanco sobre negro. Otra cosa a tener en cuenta es que todos los controles, acciones y posibilidades tienen que poder informar al usuario en forma de voz, haciendo uso de las librerías de Text-To-Speech.

Respecto a la interfaz con otros sistemas, se tienen que proporcionar los mecanismos necesarios para que el usuario no detecte la independencia de la aplicación con respecto a la red social Twitter. Además, se debe de interaccionar con una librería de terceros de Text-To-Speech de forma transparente.

Por último, la interfaz de comunicaciones será la encargada de comunicarse con la plataforma social Twitter. Esta interfaz, desde el punto de vista de los desarrolladores, debe de ser independiente para su posible mejora, crecimiento y reutilización en futuros sistemas.

4.3.2 Funciones

En esta sección se detallan las funciones del sistema a desarrollar. Vamos a desgranar estas funciones por el tipo de objetivos que se desean:

- Objetivos de usabilidad: en cuanto a la usabilidad y accesibilidad del sistema, las funcionalidades a cubrir son :

- Interacción adaptada a los usuarios basada en la localización y confirmación de las acciones.
- Interfaces basadas en combinaciones de colores de alto contraste, blanco sobre negro.
- Información sonora sobre el contenido de las distintas interfaces.
- Posibilidad de volver a la pantalla anterior.
- Información de audio sobre qué está haciendo el sistema en cada momento.
- Información de audio en cada uno de los controles de la interfaz.
- Información de audio sobre el contenido de los controles.
- Si se hace algún uso de imágenes no vectoriales, se deben mostrar en un tamaño lo más grande posible.

- Objetivos de utilidad: además, el sistema tiene que proporcionar unas herramientas que sean útiles para el usuario. Estas son:

- El sistema tiene que mostrar los últimos tweets (la mayor cantidad posible).
- Tiene que estar habilitado el envío de tweets.
- Posibilidad de retweet.
- Posibilidad de conterstar tweets de otros usuarios.
- Posibilidad de refrescar el timeline.

- Posibilidad de entrar en el timeline de otros usuarios y mostrar los últimos tweets de los mismos.
- Acceso a los detalles de usuario que muestren su nombre, su nick, el número de tweets escritos, el número de seguidores qué tiene y el número de usuarios a los qué sigue. Además, si es posible, se mostrará su avatar o imagen de usuario.
- Acceso a los detalles de otros usuarios de la misma manera que a los propios detalles.
- Acceso a la lista de seguidores que un usuario tenga.
- Acceso a la lista de usuarios a los que un usuario sigue.
- Si la relación entre el usuario de la aplicación y otro usuario es, al menos, de entrada (que ese otro usuario siga al usuario de la aplicación) , posibilidad de envío de mensajes directos (privados).
- Posibilidad de seguir a un usuario.
- Posibilidad de dejar de seguir a un usuario.

4.3.3 Requisitos de rendimiento

En cuanto a los requisitos de rendimiento, la aplicación esta diseñada para dispositivo móviles iPhone con la versión 5.0 o superior del sistema operativo iOS. Esta versión del sistema operativo no funciona en dispositivos anteriores al iPhone 4, y este ofrece las características mínimas para un correcto funcionamiento de la aplicación, aunque pueden aparecer diferencias mínimas entre los efectos visuales de la interacción y los efectos de audio por el peso de las voces de la librería de Text-To-Speech.

Por otro lado, la aplicación deberá estar conectada a internet desde el dispositivo móvil, y para su correcto funcionamiento, la conexión mínima debe ser de 3G para no tener tiempos de carga excesivos. Con conexiones wifi la aplicación proporciona unos tiempos de carga reducidos.

4.3.4 Restricciones de diseño

Debido a las restricciones impuestas por Apple en el desarrollo de aplicaciones para dispositivos iOS [24], puede que algún componente no pueda ser diseñado de forma accesible.

Debido a las restricciones impuestas por Twitter en el número de peticiones a su servicio, el sistema tiene que intentar obtener la máxima información posible en el menor número de llamadas a la API. Si se realizan más 350 llamadas a la API en un periodo de tiempo menor que una hora, la API devolverá de respuesta ERROR durante la siguiente hora desde el momento en que se supere esta cifra. Siempre que se tenga que realizar una llamada a la API y esta proporcione un parámetro en el que se pueda indicar la cantidad de información de respuesta, este parámetro deberá adoptar el máximo valor permitido por la API. Siempre se primará el realizar menos llamadas antes que obtener más información.

4.3.5 Atributos del sistema

Los atributos con los que contará nuestro sistema serán:

- Usabilidad: Será accesible para personas con cualquier grado de discapacidad visual.
- Seguridad: El sistema accederá a la cuenta de Twitter almacenada de forma segura (esa gestión de la seguridad la controla Apple) en el terminal y tras la confirmación del usuario.
- Fiabilidad: La aplicación no se podrá utilizar si no carga correctamente los datos por motivos de conexión a la red o de permisos en el acceso a la cuentas de Twitter del usuario. En todo caso, la aplicación informará de cualquier situación anómala al usuario.
- Mantenimiento: El mantenimiento de las aplicaciones para dispositivos iOS es muy sencilla, al ser versiones de la aplicación que se van publicando en el AppStore como extensiones de la aplicación base.

4.3.6 Otros requisitos

La aplicación deberá de ser pensada en, al menos, tres idiomas, inglés, castellano y catalán. A la hora de distribuir la aplicación, por petición de Acapela, qué es quién proporciona la librería de Text-To-Speech, se distribuirá una aplicación por idioma.

5. Análisis

Para un correcto entendimiento del trabajo a realizar en cualquier proyecto, antes de comenzar la fase de desarrollo se debe de realizar una fase de análisis que determine las pautas a seguir durante la elaboración del mismo. En este capítulo se mostrará una visión previa de las interfaces, funcionalidades y estructura del proyecto, apoyándose en algunos diagramas descritos por el lenguaje de modelado UML. Además, se describirá cómo se realizan las llamadas a la API de servicios REST de Twitter y se hablará de los prototipos sobre los cuales basar las interfaces gráficas de usuario.

5.1 Diagrama de casos de uso

Los diagramas de casos [24] de uso muestran gráficamente cómo interaccionan los usuarios (actores) con el sistema y qué funcionalidad ofrece este. Además, para fortalecer el entendimiento de las distintas relaciones y acciones que se muestran en el diagrama, se incluirán una serie de plantillas descriptivas.

Como podemos observar en la figura 16, el diagrama nos muestra dos actores, el usuario, una persona discapacitada visual que se haya registrado previamente en la red social Twitter, y un segundo actor que es la red social Twitter en sí misma. A grandes rasgos, el usuario se relaciona de diversas formas con la aplicación, que a su vez, y según la acción requerida, realiza llamadas a la red social. Twitter le ofrece respuestas a la aplicación, con datos o con un mensaje de error, y estas son tratadas y ofrecidas al usuario para que las utilice como crea conveniente.

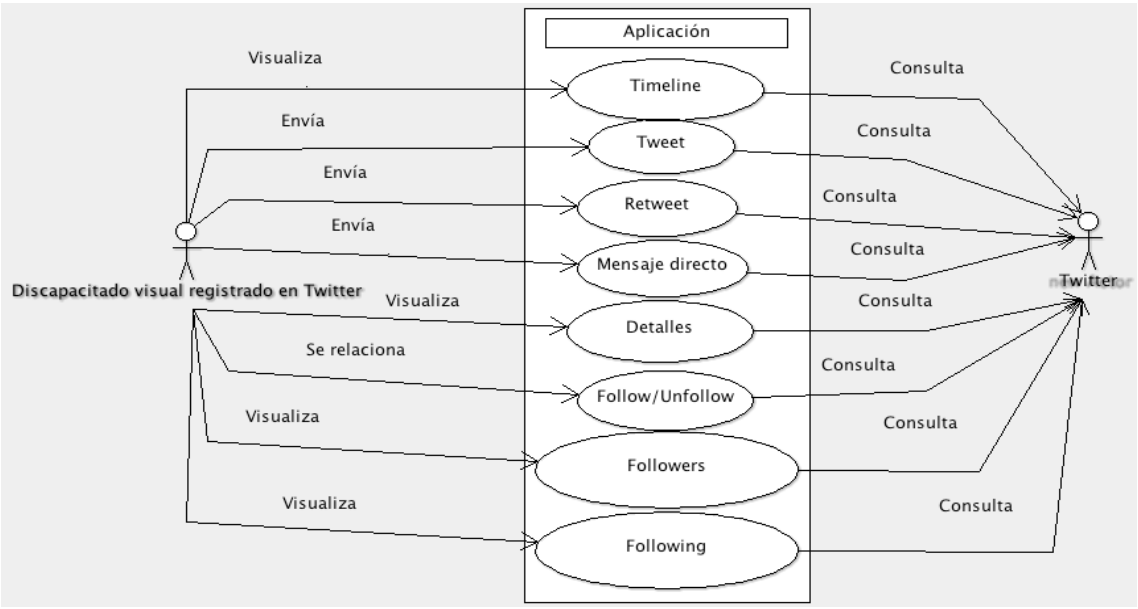


Figura 16: Diagrama de casos de uso.

A continuación describiremos los distintos casos de usos mediante el uso de plantillas.

El primer caso de uso es la visualización del timeline:

<table>
<tr><td><u>Caso de uso</u></td><td>Visualizar timeline.</td></tr>
<tr><td>Actores</td><td>Usuario (iniciador), Twitter.</td></tr>
<tr><td>Resumen</td><td>El usuario puede consultar su timeline o el de otros usuarios, mostrando, como máximo, los últimos 200 tweets.</td></tr>
<tr><td>Precondiciones</td><td>Es un usuario válido y hay conexión a la red.</td></tr>
<tr><td>Postcondiciones</td><td>Si es el timeline personal, se puede enlazar con el timeline de los usuarios que hayan publicado alguno de los, como máximo, últimos 200 tweets.</td></tr>
<tr><td>Incluye</td><td>--</td></tr>
<tr><td>Extiende</td><td>--</td></tr>
<tr><td>Hereda de</td><td>--</td></tr>
<tr><td>Flujo de eventos</td><td>
<table>
<tr><th><u>Usuario</u></th><th><u>Sistema</u></th></tr>
<tr><td>1. El usuario entra en el sistema.</td><td>2. El sistema consulta las credenciales con Twitter.</td></tr>
<tr><td></td><td>3. El sistema muestra el timeline personal.</td></tr>
<tr><td>4. El usuario pide el acceso al timeline de otro usuario.</td><td>5. El sistema realiza la petición a Twitter.</td></tr>
<tr><td></td><td>6. El sistema muestra el nuevo timeline.</td></tr>
</table>
</td></tr>
</table>

Gracias al sistema también podremos consultar los detalles de un usuario:

<table>
<tr><td><u>Caso de uso</u></td><td>Visualizar detalles de usuario</td></tr>
<tr><td>Actores</td><td>Usuario (iniciador), Twitter</td></tr>
<tr><td>Resumen</td><td>El usuario puede ver los detalles propios o los de otros usuarios. Estos detalles son nombre, nick dentro de la red social, cantidad de tweets enviados, número de followers, número de following, si se sigue o no al usuario y el avatar.</td></tr>
<tr><td>Precondiciones</td><td>Es un usuario válido y hay conexión a la red.</td></tr>
<tr><td>Postcondiciones</td><td>--</td></tr>
<tr><td>Incluye</td><td>--</td></tr>
<tr><td>Extiende</td><td>--</td></tr>
<tr><td>Hereda de</td><td>--</td></tr>
<tr><td>Flujo de eventos</td><td>
<table>
<tr><th><u>Usuario</u></th><th><u>Sistema</u></th></tr>
<tr><td>1. El usuario accede a los detalles.</td><td>2. El sistema realiza las distintas consultas pertinentes para la obtención de todos los detalles.</td></tr>
<tr><td></td><td>3. Recibe confirmación y muestra los detalles al usuarios.</td></tr>
</table>
</td></tr>
</table>

Otra de las posibilidades será ver la lista de *followers* de un usuario:

Caso de uso	Visualizar lista de *followers*
Actores	Usuario (iniciador), Twitter
Resumen	El usuario puede consultar si lista de seguidores o la lista de seguidores de otros usuarios.
Precondiciones	Es un usuario válido y hay conexión a la red.
Postcondiciones	--
Incluye	Visualizar detalles.
Extiende	--
Hereda de	--
Flujo de eventos	

Usuario	*Sistema*
1. El usuario accede a la lista de seguidores.	2. El sistema consulta los identificadores de los usuarios.
	3. Recibe la respuesta con los identificadores.
	4. Procesa los identificadores y realiza la petición a Twitter de los nombres y nicks asociados a esos identificadores.
	5. Recibe la respuesta y se lo comunica al usuario.
	6. Se le muestra la lista al usuario.

Al igual que la lista de *followers*, se podrá visualizar la lista de usuarios que un usuario sigue:

Caso de uso	Visualizar lista de *following*
Actores	Usuario (iniciador), Twitter
Resumen	El usuario puede consultar si lista de usuarios a los que sigue o la lista de usuarios a los que siguen otros usuarios.
Precondiciones	Es un usuario válido y hay conexión a la red.
Postcondiciones	--
Incluye	Visualizar detalles.
Extiende	--
Hereda de	--
Flujo de eventos	

Usuario	*Sistema*
1. El usuario accede a la lista de usuarios seguidos.	2. El sistema consulta los identificadores de los usuarios.
	3. Recibe la respuesta con los identificadores.
	4. Procesa los identificadores y realiza la petición a Twitter de los nombres y nicks asociados a esos identificadores.
	5. Recibe la respuesta y se lo comunica al usuario.
	6. Se le muestra la lista al usuario.

Otro caso de uso que encontramos es el envío de tweets:

Caso de uso	Envío de tweets
Actores	Usuario (iniciador), Twitter
Resumen	El usuario publica un tweet en la red social.
Precondiciones	Es un usuario válido y hay conexión a la red.
Postcondiciones	El tweet se almacena satisfactoriamente en Twitter.
Incluye	--
Extiende	--
Hereda de	--
Flujo de eventos	

Usuario	*Sistema*
1. El usuario accede al teclado.	
2. El usuario redacta un tweet de hasta 140 caracteres.	
3. Confirma el envío del tweet.	4. El sistema procesa el tweet.
	5. El sistema envía el tweet a Twitter.
	6. El sistema recibe la confirmación de Twitter y se lo comunica al usuario.

También encontramos el caso de uso de envío de retweets:

Caso de uso	Envío de retweets
Actores	Usuario (iniciador), Twitter
Resumen	El usuario publica en su timeline el tweet escrito por otro usuario.
Precondiciones	Es un usuario válido y hay conexión a la red.
Postcondiciones	El retweet se almacena satisfactoriamente en la red social Twitter.
Incluye	Visualizar timeline.
Extiende	--
Hereda de	--
Flujo de eventos	

Usuario	*Sistema*
1. El usuario lee un tweet.	
2. Pulsa el botón de retweet.	3. El sistema avisa a Twitter de la acción.
	4. El sistema recibe la confirmación y se lo comunica al usuario.

El sistema será capaz de enviar mensajes directos:

Caso de uso	Envío de mensajes directos
Actores	Usuario (iniciador), Twitter
Resumen	El usuario envía un tweet privado a un usuario que le sigue.
Precondiciones	Es un usuario válido y hay conexión a la red. El usuario al que se le envía el mensaje directo tiene que ser seguidor del usuario que lo envía.
Postcondiciones	El mensaje directo se almacena satisfactoriamente en la red social Twitter.
Incluye	Envío de tweets. Visualización de detalles.
Extiende	--
Hereda de	--
Flujo de eventos	

Usuario	*Sistema*
1. El usuario accede al teclado.	
2. El usuario redacta un tweet de hasta 140 caracteres.	
3. Confirma el envío del tweet.	4. El sistema procesa el tweet.
	5. El sistema envía el mensaje directo a Twitter.
	6. El sistema recibe la confirmación y se lo comunica al usuario.

El último caso de uso será poder seguir o dejar de seguir a un usuario:

Caso de uso	Follow/Unfollow usuario
Actores	Usuario (iniciador), Twitter
Resumen	Se establece o se rompe una relación entre usuarios de la red social, de manera que se muestren o no los tweets del usuario en el timeline personal.
Precondiciones	Es un usuario válido y hay conexión a la red.
Postcondiciones	Se sigue o se deja de seguir a un usuario de Twitter.
Incluye	Visualizar detalles.
Extiende	--
Hereda de	--
Flujo de eventos	

Usuario	*Sistema*
1. El usuario presiona el botón.	2. El sistema comprueba si se ha pulsado follow si no se seguía al usuario o unfollow si era seguido.
	3. El sistema envía la petición a Twitter.
	4. Se recibe la confirmación y se le comunica al usuario.
	5. Se muestra el nuevo estado de la relación.

5.2 Diagrama de actividades

Este diagrama [24] trata de clarificar cuál es el flujo que hay que seguir para la realización de diversas acciones dentro de la aplicación. Este flujo se ve condicionado por diversas situaciones que se pueden dar durante el uso de la aplicación.

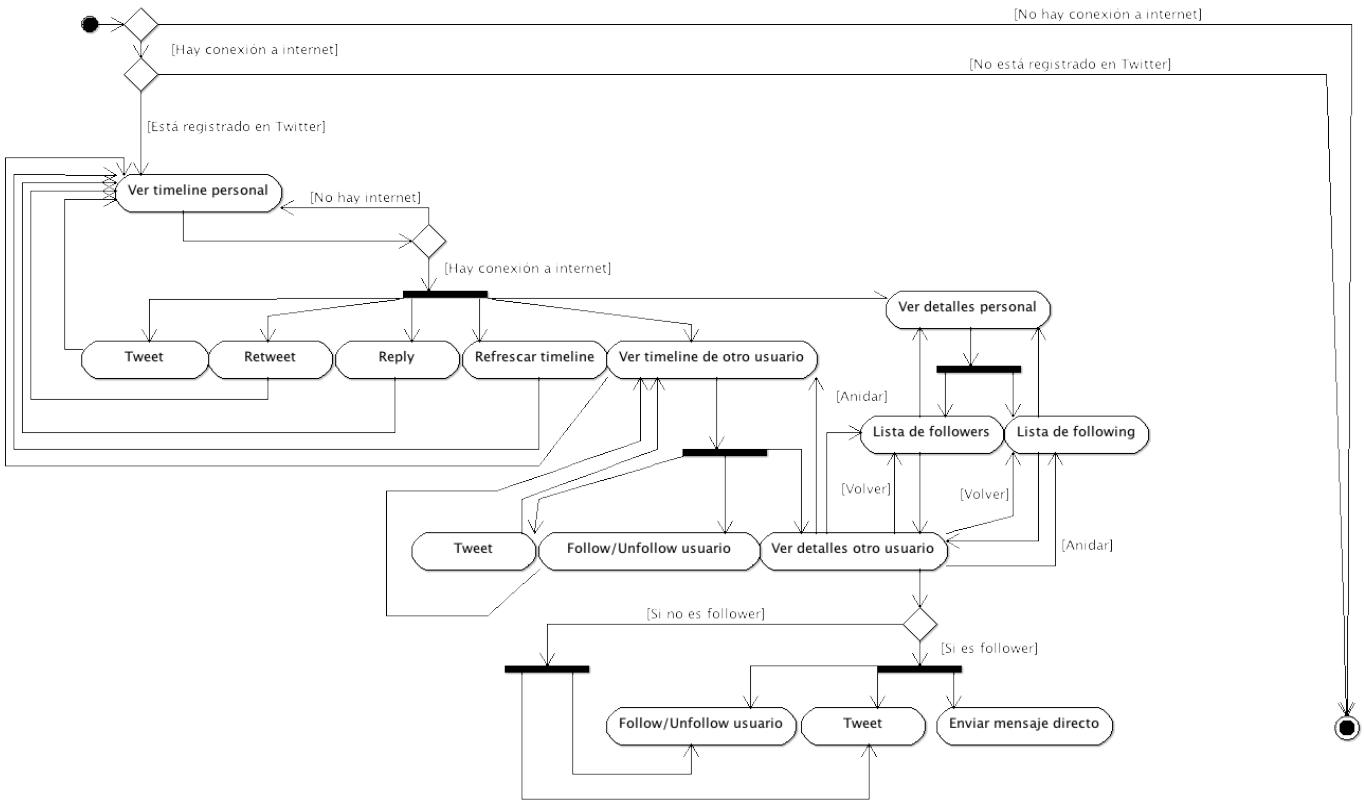


Figura 17: Diagrama de actividades de la aplicación.

En la figura 17 podemos ver el punto de inicio de la aplicación y el punto de destrucción de la misma. Aunque el punto de destrucción de la aplicación sólo se muestra en el caso inicial de que el usuario no esté registrado en la red social Twitter o no haya conexión a la red, puede ser alcanzado en cualquier otro momento si se pierde la conexión. Además, en realidad no es un punto de destrucción de la aplicación, porque según las normas establecidas por Apple para el desarrollo de aplicaciones para dispositivos iOS, no es conveniente destruir o finalizar mediante código las aplicaciones. Por este motivo, el punto de destrucción de la aplicación es en realidad un punto de bloqueo de la aplicación, en el cual, si es alcanzado, se mostrará la interfaz en la que nos encontrábamos antes de realizar la acción que nos ha llevado a ese punto, o a una interfaz de bloqueo si el punto es alcanzado al acceder a la aplicación.

Por otro lado, observamos que inicialmente la aplicación entra en el timeline del usuario y le otorga la funcionalidad básica de escribir un tweet, hacer un retweet, contestar el tweet de otro usuario, y refrescar el timeline para ver el mismo de forma actualizada. Además, también se le da la posibilidad al usuario de ver el timeline de otros usuarios y los detalles de la cuenta propia. Si elegimos la opción de ver el timeline de otro usuario podermos enviar un tweet a ese usuario, seguir o dejar de seguir a ese usuario y también podremos enlazar a los detalles de ese otro usuario. Otra manera para entrar a los detalles de otro usuario es enlazar desde los propios detalles del usuario principal, consultando sus listas de *followers* y *following*. Dentro de los detalles de terceros usuarios podemos seguir o dejar de seguir a ese usuario, enviarle un tweet y, si es uno de nuestros *followers*, enviarle un mensaje directo (un tweet privado) a ese usuario.

5.3 Diagrama de clases

En el diagrama de clases [24] se muestra gráficamente la estructura de la aplicación y su jerarquía de clases. Gracias a este diagrama, podemos observar el nombre de las clases, sus atributos y sus métodos, además de las reglas de visibilidad de estos componentes. También podemos ver cómo se relacionan las clases y sirve como guía para que los desarrolladores implementen el sistema.

En la figura 18 (página 48) podemos observar las clases y las relaciones que existen entre ellas. Además, vemos que existen dos paquetes, el paquete TTS y el paquete LVLib. El primer paquete proporciona las interfaces y las clases necesarias para que los componentes de la aplicación tengan propiedades de sistemas con Text-To-Speech. Estas interfaces y clases son privadas y han sido implementadas por la compañía Acapela, por lo que no se ofrecen más detalles de los mismos. El paquete LVLib es la librería de controles accesibles que se ha desarrollado. Este diagrama de clases se refiere a la aplicación y por ello se muestra así la librería, aunque se entrará en más detalles en el capítulo en el que se habla del desarrollo del proyecto.

Una de las cosas que se pueden observar en este diagrama, es, como decíamos, las clases. Tal y como se muestra, el sistema tendrá seis interfaces: la interfaz TwitterViewController, qué es la que nos va a mostrar los distintos timelines (propios y de otros usuarios); AboutViewController, qué mostrará los detalles de la compañía; TwitterUserDetailsViewController, donde se podrán observar los detalles de los distintos usuarios que se consulten; TwitterFollowersViewController, la encargada de representar las listas tanto de *followers* como de *following* de un usuario (el nombre de la clase se acortó poniendo sólo el nombre de una de las opciones, pero es válida para las dos); TwitterKeyboardViewController, accesible desde cualquier interfaz que ofrezca la posibilidad de escribir un tweet, mostrando el teclado con un contador de caracteres; y por último, la UIBlockerView, una interfaz que sirve como pantalla de

carga entre operaciones e interfaces, para que el usuario reciba una respuesta interactiva mientras que el sistema realiza consultas y operaciones. Si continuamos observando, vemos que todas estas interfaces están relacionadas entre ellas, pero, sobre todo, están todas relacionadas con la librería LVLib. Todas las interfaces estarán realizadas mediante controles de esta librería, ya que todos los controles tienen que ser accesibles. La librería esta relacionada con el paquete TTS para tener características de Text-To-Speech y con la clase SoundEffect para que los controles proporcionen efectos sonoros cuando se interactúa con ellos.

Por otro lado, nos queda por describir dos clases mas, LVAppDelegate y TwitterEngine. LVAppDelegate es una clase autogenerada por el IDE XCode cuando se comienza un proyecto de desarrollo de una aplicación para dispositivos iOS. Esta clase es la encargada de iniciar la aplicación y llamar a la interfaz inicial, TwitterViewController, avisándole de que es la primera vez y tiene que cargar esta interfaz con los datos del usuario del dispositivo. TwitterEngine es la clase qué se encarga de hacer de intermediario entre las interfaces del sistema y la red social Twitter. Recibe datos de que desea hacer el usuario, los procesa, envía las pertinentes peticiones, firmadas con los datos del usuario, a la red social, recibe las respuesta, las procesa y se las ofrece a las interfaces para que las muestren de la forma que crean conveniente. Las peticiones se realizan mediante llamadas a la API de servicios REST que proporciona la plataforma, algunas de las cuales explicaremos en la siguiente sección.

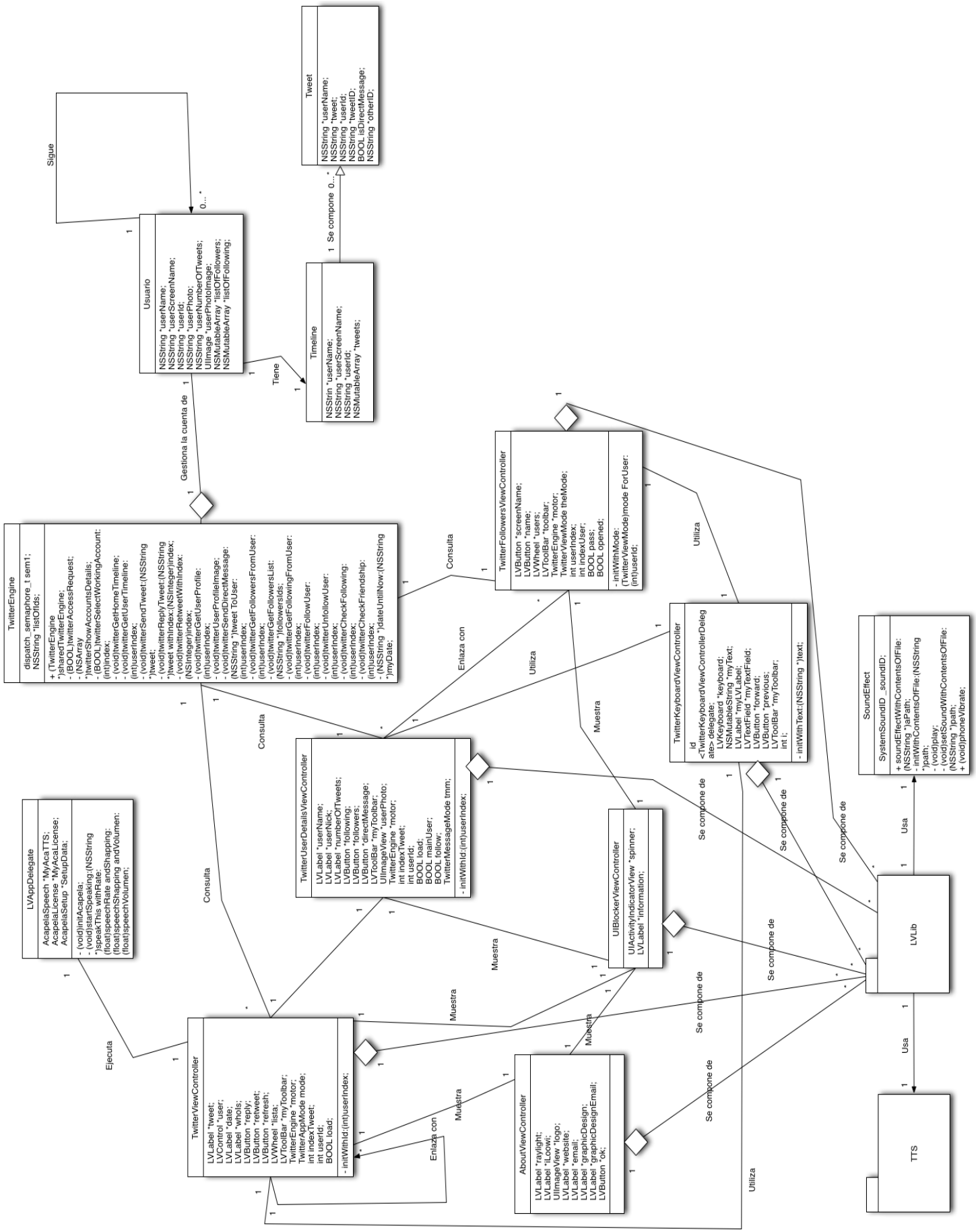


Figura 18: Diagrama de clases de la aplicación.

5.4 Servicios REST

En el capítulo tercero definíamos qué es la API de servicios REST, cómo funciona y que podemos esperar de este tipo de servicios. En esta sección se pretende analizar más profundamente cuáles son las llamadas que van a ser necesarias en la elaboración de este proyecto y qué información proporcionan. No se va a detallar cómo se hacen las llamadas ni cómo se procesan las peticiones y las respuestas, pero sí que se mostrarán las URLs que se van a utilizar y se describirán los parámetros que se incluyen en estas URLs para obtener las respuestas deseadas.

En primer lugar, la primera petición que se tiene que realizar para poder hacer uso de la API de servicios REST de Twitter es la de acceso firmado al sistema. Esta petición viene proporcionada con la integración de Twitter de iOS 5.0, con lo que no supone mayor problema en cuanto al análisis de la misma.

Una vez aprobado el acceso al sistema, la primera llamada que debemos realizar es la que nos proporcione la información del timeline personal:

```
GET http://api.twitter.com/1/statuses/home_timeline.json?count=200
```

Como vemos en la petición, el parámetro `count` recibe un valor de 200. El objetivo de esta petición es recibir un documento JSON ordenado cronológicamente hasta la fecha actual, con el máximo número de tweets posible, limitado a 200 tweets por la plataforma, y que se proporcionen los detalles de los mismos.

De la misma manera que accedemos a nuestro timeline, podemos acceder al timeline de otros usuarios:

```
GET
http://api.twitter.com/1/statuses/user_timeline.json?count=200&user_id
=x
```

Las diferencias entre esta petición y la anterior es que pedimos el `user_timeline` en vez del `home_timeline` y le proporcionamos un `user_id`, que en este caso le hemos dado un valor `x` a modo de ejemplo.

Para el envío de tweets, la petición ha realizar sería:

```
POST http://api.twitter.com/1/statuses/update.json?status=mensaje
```

El parámetro status es el que contiene el tweet a enviar, en este caso `mensaje`. Si `mensaje` superase los 140 caracteres, se enviaría igualmente el tweet, pero se cortaría en el carácter número 140. Por otro lado, si quisiésemos enviar un tweet como respuesta a otro tweet previo, se debería introducir el parámetro `in_reply_to_status_id` con el número que identifica al tweet a contestar como valor.

Para retwittear un tweet es necesario conocer el número que identifica a dicho tweet. Una vez conocido el identificador, la petición sería:

```
POST http://api.twitter.com/1/statuses/retweet/x.json
```

Esta petición no necesita parámetros, pero en la petición se observa que el final de la misma es `x.json`. Esta `x` tiene que ser sustituida por el número identificador del tweet a retwittear.

Para obtener los detalles de un usuario, la petición que se deberá realizar será:

```
GET http://api.twitter.com/1/users/lookup.json?user_id=x
```

Como parámetro se debe de proporcionar el número identificador del usuario, en este caso representado con `x`.

La forma más eficiente de conseguir tanto la lista de *followers* como la lista de *following* según los datos que proporcionan como respuestas las distintas peticiones a la API de servicios REST de Twitter es obtener los números de identificación de los usuarios que forman estas listas y después obtener los detalles de los mismos. Para ello, las peticiones que se tienen que realizar en primer lugar serían:

```
GET http://api.twitter.com/1/followers/ids.json?user_id=x
```

```
GET http://api.twitter.com/1/friends/ids.json?user_id=x
```

Estas peticiones reciben como parámetro el número de identificación del usuario sobre el que se quieren consultar estas listas de *followers* y *following* respectivamente. A modo de demostración, el valor del parámetro `user_id` en este caso es `x`.

En segundo lugar, hay que realizar una petición que proporcione los detalles asociados a los identificadores obtenidos. Ya hemos hablado antes de esta petición, pero en este caso deberá tener unos parámetros distintos, con lo que la petición con los nuevos parámetros quedaría:

```
GET http://api.twitter.com/1/friendships/lookup.json?user_id={a,b,c…}
```

Como podemos observar, el parámetro `user_id` ya no es un único valor, si no que es una lista de valores que puede contener hasta 100 valores por petición, realizando así un número menor de llamadas a la API de servicios REST de Twitter.

Para crear una relación entre dos usuarios (seguir a otro usuario), la petición sería:

```
POST http://api.twitter.com/1/friendships/create.json?user_id=x
```

El parámetro `user_id`, qué aquí toma el valor de `x`, es el número identificador del usuario al que se va a seguir.

El proceso contrario al anterior sería el de destruir una relación con otro usuario (dejar de seguir a un usuario). Para ello, la petición quedaría de la siguiente manera:

```
POST http://api.twitter.com/1/friendships/destroy.json?user_id=x
```

De nuevo, el valor `x` de el parámetro `user_id` tiene que ser el identificador numérico del usuario sobre el que realizar la acción.

Para poder seguir o dejar de seguir a un usuario, primero tenemos que comprobar que relación existe entre dos usuarios, o si no existe ninguna. Además, esto también servirá si queremos saber si un usuario es nuestro seguidor para poder enviarle un mensaje directo. Para conocer este dato, la petición es la siguiente:

```
GET
http://api.twitter.com/1/friendships/exists.json?user_id_a=x&user_id_b
=y
```

Los parámetros, que en este caso toman valores `x` e `y`, determinan hacia que dirección estamos comprobando la relación, es decir, si queremos saber si un usuario es follower de otro o si un usuario está *following* a otro.

La última petición que se realizará es la del envío de mensajes directos. Para tener la seguridad de que esta petición se puede realizar, debemos comprobar con anterioridad que relación existe entre dos usuarios como se comentaba anteriormente. Después la petición que se debe de realizar quedaría de la siguiente forma:

```
POST
https://api.twitter.com/1/direct_messages/new.json?user_id=x&text=mens
aje
```

Los parámetros definen hacia que usuario se envía el mensaje mediante el `user_id`, en este caso `x`, y que `mensaje` se le envía gracias al parámetro `text`.

Toda la información mostrada por la aplicación se obtendrá como resultado de las peticiones descritas, aunque en algunas ocasiones se deberán de combinar para obtener el resultado deseado.

5.5 Prototipos de las interfaces gráficas de usuario

Para finalizar este capítulo de análisis vamos a describir diversos prototipos de las interfaces gráficas de usuario que garanticen una experiencia de usuario de alto nivel. El objetivo es marcar las guías que determinen como se tienen que construir las interfaces y dar algunas explicaciones acerca del porqué las interfaces están con construidas de una u otra manera.

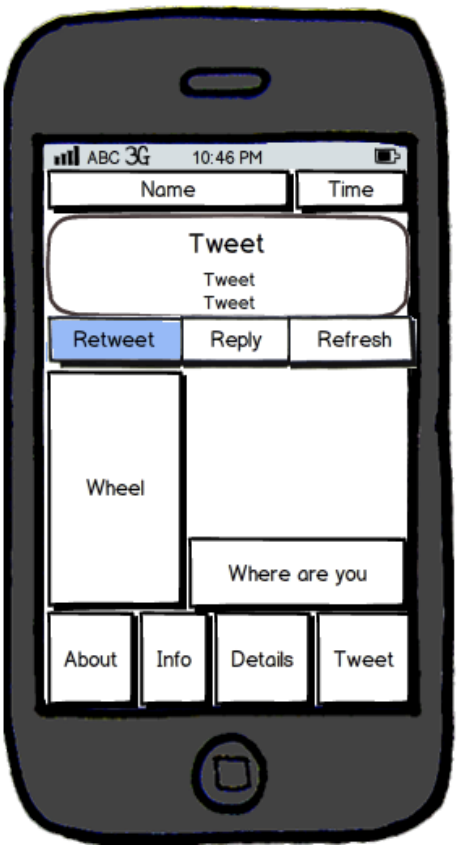


Figura 19: Prototipo de la interfaz gráfica TwitterViewController.

La primera interfaz es la que se muestra en la figura 19, correspondiente a la vista definida en la clase TwitterViewController, y que define la vista que mostrará los timelines de los usuarios. Debido a la cantidad de información que tiene que mostrar esta interfaz, hay componentes que tienen un tamaño menor que otros componentes que se consideran más prioritarios. En la parte superior encontramos dos controles, "Name" y "Time". "Name" se comportará como un botón si la interfaz se encuentran mostrando el timeline personal del usuario y como una etiqueta si se está mostrando el timeline de otro usuario. "Time" siempre será una etiqueta. Bajo estos controles se sitúa uno de los controles importantes de esta interfaz, la etiqueta "Tweet". Esta etiqueta irá mostrando los distintos tweets que componen el timeline. Casi en la zona central de la pantalla podemos encontrar tres botones, "Retweet", "Reply" y "Refresh". Estos botones representan distintas acciones disponibles en la plataforma, cómo hacer retweets, contestar tweets de otras personas o refrescar el timeline para ver los últimos tweets. En la parte central de la pantalla, pegado al margen izquierdo, se encuentra el otro control importante, el control rueda. Se detallará más adelante el porqué de este nombre, pero será el encargado de poder navegar por la lista de tweets y hacer que la etiqueta "Tweet" vaya mostrando los distintos tweets del timeline. A la derecha de este control, casi en la parte inferior de la pantalla se encuentra la etiqueta "Where are you" que informará al usuario en que timeline se encuentra. Para finalizar, en la parte inferior de la pantalla se encuentra la barra de herramientas, qué en este caso dispondrá de un botón "About" que enlazará con la información acerca de la empresa, el botón "Info", donde se informará al usuario de todas las opciones y controles disponibles en la interfaz.

La segunda interfaz de la que vamos a hablar es la interfaz que se corresponde con la clase AboutViewController. Como vemos en la figura 20, esta interfaz nos va a mostrar información acerca de la empresa, cómo contactar la misma, quien ha sido el diseñador gráfico, el logo de la compañía y el botón para volver a la aplicación en sí. En la parte superior, centrado, se encontrará el logo de la compañía. Debajo del logo y ocupando la mayor parte de la pantalla se encontrarán varias etiquetas con la información de la compañía, el contacto y el estudio de diseño gráfico que ha colaborado en la iconografía. En la parte inferior de la pantalla, ocupando lo que en otras interfaces es la barra de herramientas, encontramos el botón "Back" que nos permitirá volver al timeline de usuario.

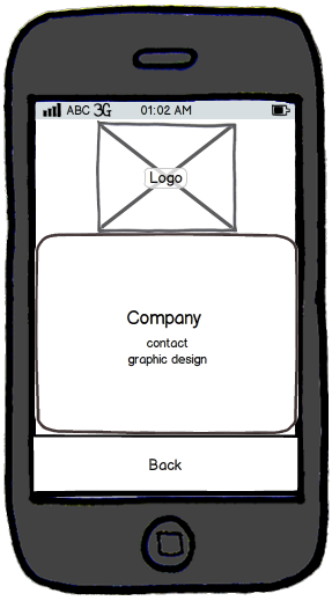


Figura 20: Prototipo de la interfaz AboutViewController.

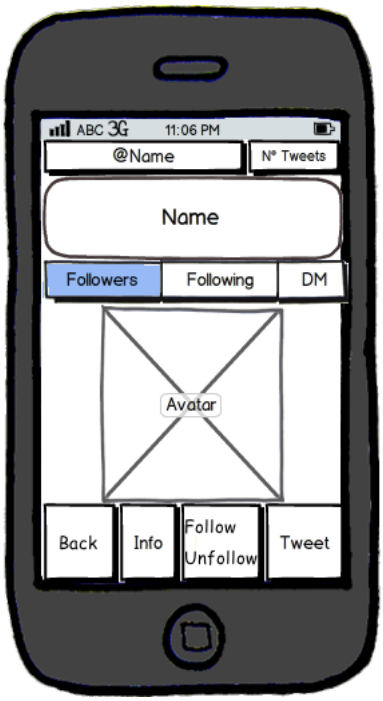


Figura 21: Prototipo de la interfaz TwitterUserDetailsViewController.

Otra de las interfaces que encontraremos es la interfaz que se representa en la clase TwitterUserDetailsViewController y que se encargará de mostrarnos los detalles de los distintos usuarios que visitemos (figura 21). La idea de construcción de esta interfaz es que esta cambie lo mínimo posible respecto a las anteriores, ya que al ser una aplicación orientada a usuarios con baja visión, es conveniente que las interfaces sean lo menos dinámicas posibles para que el usuario memorice una disposición de componentes y cuando cambie de interfaz no se sienta perdido. Con esta idea, en la parte superior encontramos la etiqueta "@Name" que corresponde al nombre del usuario dentro de la red social Twitter. A su izquierda esta la etiqueta "Nº Tweets" que muestra el número de tweets publicados por el usuario. Debajo de estas dos encontramos otra etiqueta, "Name", que muestra el nombre del usuario. Los tres botones que encontramos debajo son el botón de "Followers", que enlaza con la lista de *followers* del usuario, el botón "Following", que enlaza con la lista de usuarios a los que se sigue y el botón "DM" para enviar mensajes directos si se cumplen las condiciones. En la zona central de la interfaz encontramos el "Avatar" o imagen del usuario en la red social Twitter. Para finalizar, en la parte inferior de la pantalla encontraremos de nuevo la barra de herramientas, pero esta vez el primer botón empezando por la izquierda es el botón "Back", para volver a la pantalla anterior, y el tercer botón empezando también por la izquierda es el botón "Follow/Unfollow" para seguir o dejar de seguir al usuario.

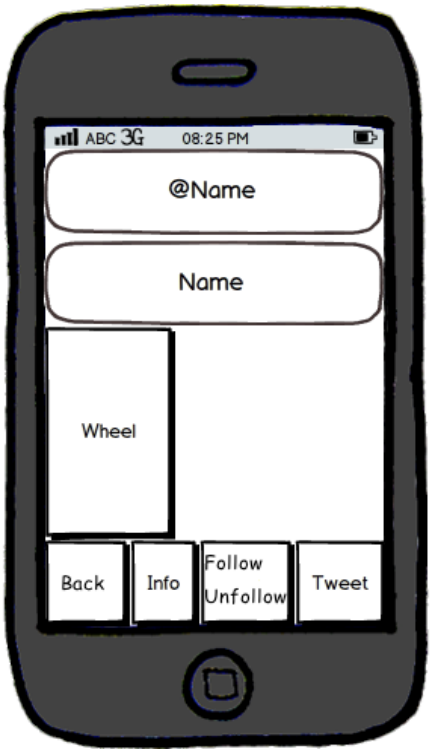


Figura 22: Prototipo de la interfaz TwitterFollowersViewController.

El cuarto tipo de interfaz que nos podemos encontrar es la asociada a la clase TwitterFollowersViewController. Esta interfaz mostrará las listas de *followers* y *following* correspondientes a un usuario. La construcción de esta interfaz sigue la regla de mantener una estructura lineal con el resto de interfaces. De esta manera, y tal y como vemos en la figura 22, en la parte superior encontramos dos etiquetas, una llamada "@Name", que muestra el nick del usuario en la red social Twitter y otra llamada "Name", con el nombre del usuario. En la parte central izquierda tenemos un control "Wheel", que será el encargado de la navegación entre los distintos usuarios que conforman la lista, haciendo que las etiquetas superiores muestren los nicks y nombres de estos usuarios. La barra de herramientas mostrará un botón "Back" para volver a la pantalla anterior, el botón de "Info" para que el usuario tenga información acerca de la interfaz, el botón para seguir o dejar de seguir los usuarios que la lista le muestra y un botón "Tweet" para escribir un tweet a los usuarios.

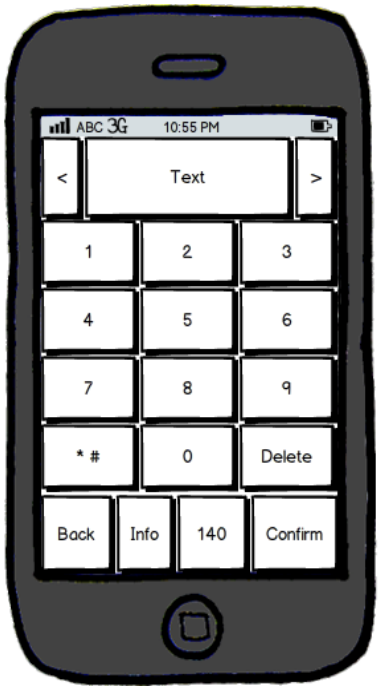


Figura 23: Prototipo de la interfaz TwitterKeyboardViewController.

Todas las interfaces que tengan la posibilidad de escribir un tweet o contestar a un tweet enlazarán con la interfaz representada en la clase TwitterKeyboardViewController. Como podemos observar en la figura 23, en la parte superior podemos encontrar en la parte central un cuadro de texto y a sus laterales, dos botones que permitan navegar por el texto. Debajo encontraremos los botones que conforman un teclado , dispuestos como los teclados que se encuentran en los teléfonos móviles no táctiles. Esta decisión se tomo después del estudio previo con los usuarios ya que encontraban muy molesto el teclado QWERTY que ofrecen los dispositivos táctiles. En la barra de herramientas encontraremos un botón "Back" para volver a la pantalla anterior sin enviar el mensaje, un botón "Info" para que el usuario tenga información acerca de la interfaz en la que se encuentra, un contador de caracteres, ya que Twitter sólo permite 140 caracteres en sus tweets y un botón para confirmar el envío del mensaje a la red social Twitter.

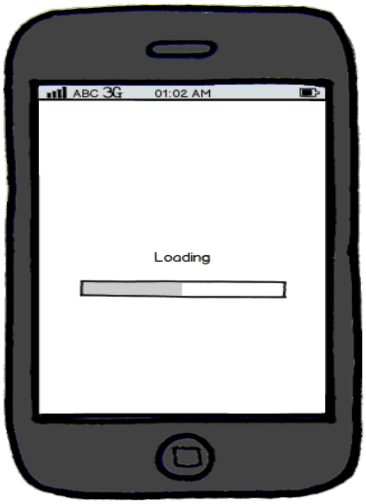


Figura 24: Prototipo de la interfaz UIBlockerViewController.

La última interfaz que podremos encontrar en la aplicación estará relacionada con todas las anteriores, ya que es la que se corresponde con la clase UIBlockerView y su objetivo es ofrecer una interfaz que bloquee la aplicación mientras esta realiza operaciones y carga las interfaces. Como vemos en la figura 24, esta interfaz es muy sencilla, mostrará una etiqueta accesible que cubra toda la pantalla y que al pulsarla se informe al usuario de la situación, y algún icono de carga, en el mockup es una barra de carga, pero en la aplicación será un Spinner, que indica de forma gráfica que el sistema esta realizando operaciones y no está parado.

6. Desarrollo

Después de conocer los requisitos de la aplicación y haber realizado un análisis previo, el siguiente paso a seguir es el desarrollar la aplicación tal y como anteriormente se ha descrito. Para ello, este proceso se dividirá en 6 fases, las cuales dividirán el capítulo en secciones. Estas fases son el estudio preliminar del problema, seguido del desarrollo de los componentes a realizar, la realización de pruebas de los componentes desarrollados, el desarrollo de la aplicación en sí, las consiguientes pruebas de la aplicación y, por último, la distribución de la misma por el AppStore, esta última a modo teórico ya que por motivos de marketing aún no ha sido acometida.

6.1 Estudio preliminar

La estrategia planteada desde la compañía Raylight para conseguir una experiencia de usuario de alto nivel es la de comprobar cómo el usuario se siente más cómodo a la hora de interactuar con los dispositivos móviles. Para ello se desarrollaron unos controles previos, que se montaron en pequeñas aplicaciones, cada una con objetivos claros y definidos. Con ellas se realizaron una serie de pruebas con usuarios discapacitados visuales para ver la reacción de estos, cuáles eran útiles, cuáles no, y, en definitiva, recibir un feedback que sirviese para realizar la interacción más adecuada para el sector de la baja visión. Estas pruebas se documentaron gracias a la realización por parte de los usuarios de un test en el que se recogieron los datos obtenidos, además de que todas las pruebas fueron registradas en video. No se van a detallar los resultados del test, aunque sí que se comentarán las conclusiones para poder entender las decisiones tomadas.

6.1.1 Aplicaciones de test

En esta sección vamos a hablar y mostrar las aplicaciones que fueron desarrolladas para realizar los test con los usuarios. Estas aplicaciones tienen diseños muy sencillos y su único objetivo es obtener una serie de datos que sirvan para un mejor diseño y desarrollo de los futuros controles que formarán parte de la librería de controles accesibles.

La primera aplicación fue ButtonMenus, una aplicación que muestra 2 botones, tal y como vemos en la figura 25. Cada uno de estos botones tienen un tipo de interacción especial. El primero de los botones necesita un dedo para la localización, y una vez localizado y pulsado, con un segundo dedo se confirma la acción. El segundo botón necesita un dedo para la localización, y una vez encontrado, al levantar el dedo de la pantalla se confirma la acción. El objetivo inicial era comprobar si el usuario detectaba con facilidad cuál era el tipo de interacción. Además, también se recogieron datos sobre la detección de los sonidos que la aplicación emitía o con que dedo hacía la exploración.

Figura 25: Aplicación de test ButtonMenus.

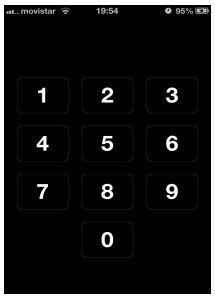


Figura 26: Interfaz gráfica de las aplicaciones de test ButtonPhone1 y ButtonPhone2.

También se desarrollaron las aplicaciones de prueba denominadas como ButtonPhone1 y ButtonPhone2. Como vemos en la figura 26, estas aplicaciones muestran 10 botones posicionados en la pantalla como un teclado numérico. En estas aplicaciones el objetivo era que el usuario averiguase que tipo de interacción usa la aplicación cada una y qué es lo que se representaba en la aplicación. En la primera aplicación, la interacción era la localización de los botones con un dedo, y, con el mismo, levantar el dedo del botón localizado para confirmar la acción. En la segunda aplicación, la localización se realiza de la misma manera, pero para confirmar la acción se necesita un segundo dedo que pulse en cualquier lugar de la pantalla. Al igual que en la aplicación anterior, otros de los detalles a tener en cuenta en esta prueba son el tiempo que tarda el usuario en hacer las averiguaciones, con que dedo usa la aplicación y que sonidos detecta.

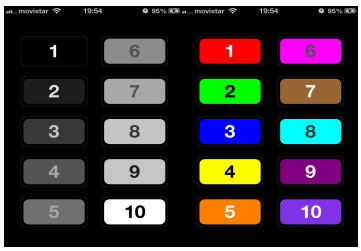


Figura 27: Aplicaciones de test Colors1 y Colors2.

Otro par de aplicaciones que fueron desarrolladas son las aplicaciones Colors1 y Colors2. En ambas se disponen 10 botones situados en 5 filas y 2 columnas, numeradas de 1 a 10, empezando desde la parte superior izquierda a la parte inferior derecha de la pantalla, y de forma incremental por columna. En la primera de ellas los botones comienzan con una representación gráfica con una combinación de colores de blanco sobre negro, y según aumenta la cifra del botón se va haciendo una combinación sobre la escala de grises entre la etiqueta del botón y el fondo hasta llegar en la última posición a la combinación negro sobre blanco. La segunda aplicación tiene el mismo orden numérico, pero cada uno de los botones tiene una combinación de colores basada en el código RGB. Estas aplicaciones, las cuales podemos observar en la figura 27, sólo tienen un tipo de interacción, la de localizar con un dedo y levantar el dedo de la pantalla, ya que en estas aplicaciones el objetivo no está basado en la interacción, si no que el principal objetivo de estas aplicaciones

es conocer que combinación es la ideal para que los usuarios con baja visión puedan distinguir en mayor grado los controles representados en la pantalla. Además, con la segunda aplicación también se intenta conocer si existe alguna combinación de colores que puedan dar una imagen más colorida a las aplicaciones. Como siempre, el tiempo en el que el usuario toma una decisión, con que dedo explora la aplicación y los sonidos que detecta son detalles recogidos en el estudio.

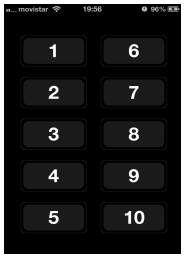


Figura 28: Aplicación de test Voices

Al igual que se desarrollaron aplicaciones orientadas a recoger datos acerca de los aspectos gráficos de los controles, también se desarrolló una aplicación para obtener datos acerca de los aspectos sonoros de la aplicación. Esta aplicación, cuya interfaz vemos en la figura 28 y que sigue la distribución de controles que tenían las anteriores aplicaciones, adquirió el nombre de Voices. En ella, cada uno de los controles recita un texto a diferentes velocidades. En este caso, la interacción no es lo más importante, si no que lo que interesa de esta aplicación es conocer si el usuario detecta todos los sonidos que la aplicación emite y ver a que velocidad del texto el usuario se siente más cómodo, ya que una velocidad lenta a la hora de recitar un texto puede resultar muy pesado para el usuario y una velocidad rápida puede llegar a ser ininteligible para el usuario. Otro objetivo de la aplicación es comprobar qué voz de las que recitan los textos es más agradable para el usuario, ya que va a ser la voz a la que los usuarios tienen que acostumbrarse al interactuar con las aplicaciones que se desarrollen con los controles que se construyan.

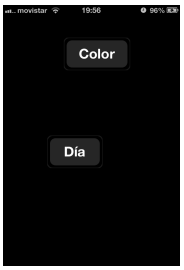


Figura 29: Aplicación de test SegmCtrlDays

Para conocer otro tipo de controles, también se desarrolló la aplicación SegmCtrlDays. Esta aplicación, mostrada en la figura 29, muestra dos controles que, en aspecto, pueden parecer dos botones, pero que en realidad son una representación adaptada de los controles segmentados disponibles en aplicaciones móviles. Cada uno de ellos agrupa una serie de datos por los cuales se puede navegar para elegir una de esas opciones. El primero de esos controles agrupa un conjunto de colores y el segundo control contiene los días de la semana. Para elegir una opción de las disponibles en cualquiera de los dos controles, el usuario tiene que localizar el control deseado (Color o Día) y una vez localizado, con un segundo dedo se puede cambiar la opción del control cada vez que se pulse en la pantalla. Una vez localizada la opción deseada, basta con levantar el primer dedo de la pantalla para seleccionar la opción. El objetivo de esta aplicación es comprobar si el usuario es capaz de averiguar cómo se maneja este nuevo tipo de control, cuanto tiempo tarda en aprender ese manejo y que impresiones le deja al usuario este nuevo tipo de control.

Figura 30: Aplicación de test SCPhone.

La siguiente aplicación que se desarrolló la denominamos SCPhone. En ella se representa un teclado numérico y una etiqueta (figura 30). Este teclado tiene la misma funcionalidad que los teclados numéricos clásicos de los teléfonos anteriores a los dispositivos táctiles. Al igual que la aplicación anterior, cada control es de tipo segmentado, por lo que cada uno de ellos tiene varias opciones, en este caso caracteres que se encuentran distribuidos y agrupados de idéntica manera que lo estaban en los teléfonos que disponían de este tipo de teclado (por ejemplo, la tecla 2 contiene a los caracteres a, b y c). Para elegir una de ellas, el usuario tiene que localizar el control que contenga el carácter deseado, pulsar con un segundo dedo en la pantalla hasta que se encuentre la letra buscada y, por último, levantar el dedo con el que localizó el control para que el carácter se muestre en la etiqueta. El objetivo de esta aplicación es comprobar si el usuario entiende el funcionamiento de los controles, conocer el tiempo que tarda en acostumbrarse a la interacción, conocer el tiempo que tarda en escribir un texto, y, como el resto de aplicaciones, ver con que dedo realiza las exploraciones y saber si distingue los distintos sonidos de la aplicación.

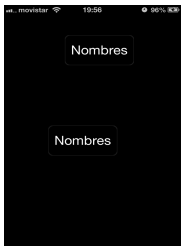


Figura 31: Aplicación de test WheelNames.

Por último, y para probar otro tipo de control, se desarrolló una aplicación llamada WheelNames. Esta aplicación contiene dos controles, de nuevo con apariencia de botones, pero que en realidad representan listas de opciones (figura 31). A pesar de que suene parecido a los controles presentados en anteriormente, existe una gran diferencia: los controles segmentados son útiles en conjuntos cortos de opciones, mientras que los controles "rueda" (llamados así porque la filosofía de interacción del control es la misma qué la rueda del ratón) permiten navegar por listas con gran cantidad de datos y opciones de forma veloz y ágil. Hay que tener en cuenta que las personas que no padecemos ningún tipo de discapacidad visual recorremos las listas basándonos completamente en lo que observamos (por ejemplo, si queremos buscar un contacto llamado Raúl, navegaremos en primer lugar hasta la posición de la lista donde visualicemos la letra R y después buscaremos en ese grupo el contacto con el nombre Raúl de forma visual). Cada control disponible en esta aplicación representa una lista de nombres, como la que se puede encontrar en la lista de contactos. Para interaccionar con los controles, el usuario tiene que localizar el área del control, y una vez localizada, deslizará en dirección horizontal o vertical un segundo dedo (se representan las dos opciones en la aplicación, una por cada control) para ir navegando por todos los componentes de la lista. El objetivo de esta aplicación es comprobar si el usuario entiende el nuevo tipo de interacción, cuanto tiempo tarda en comprenderlo y ver con que opción de navegación se siente más cómodo, si con la navegación horizontal o la vertical.

6.1.2 Conclusiones y resultados del test preliminar

Gracias al test realizado se obtuvieron una serie de datos que marcaron la dirección hacia dónde se debía de dirigir el diseño de los controles a realizar. Aunque se recogieron muchos datos cuantificables y útiles a la hora de tomar decisiones, los datos más importantes que recogimos fueron todos aquellos más enfocados a conceptos e ideas.

Antes de hablar de estos conceptos e ideas, la primera conclusión que extrajimos del estudio es qué los usuarios que ya habían tenido algún tipo de contacto con dispositivos táctiles aprendían la filosofía de los controles presentados a un ritmo superior que aquellos usuarios que nunca se habían enfrentado a una pantalla táctil. Otra conclusión fue qué pudimos observar que la forma en que los usuarios interaccionan con estos dispositivos es muy similar a la hora de explorar las aplicaciones, pero difiere un poco entre usuarios a la hora de, sobre todo, usar un segundo dedo para la confirmación de acciones, navegación en listas o selección de opciones.

Los conceptos e ideas extraídos del estudio hicieron que, en algunos casos, tuviésemos que redefinir algunos de los controles presentados en las aplicaciones de test. Además estos conceptos e ideas también fomentaron el diseño y el desarrollo de nuevos controles. Por ejemplo, una de las ideas que obtuvimos del estudio es qué el control segmentado podría tener una versión más eficiente si, en vez de necesitar dos dedos para buscar el control, seleccionar la opción mediante pulsaciones y, finalmente, levantar el dedo para fijar la selección, todo este proceso se resumiese en localizar el control y seleccionar la acción. De este concepto nació el control segmentado automático, que recorre la lista de opciones con sólo mantener el dedo dentro del control y que basta con levantar el dedo cuando se recite la opción deseada para realizar la selección. De esta nueva idea y de algunas de las sugerencias recibidas por parte de los usuarios de mejorar los teclados disponibles en los dispositivos táctiles, surgió la idea de crear un componente teclado. Este teclado contará con una distribución como la de los teclados existentes en dispositivos móviles no táctiles, y, gracias a los controles segmentados automáticos permitirá escribir texto de una manera más ágil y sencilla. Otra de las ideas que recogimos con especial interés era el hecho de que las interfaces de las aplicaciones debían de ser lo más estáticas posibles, para facilitar el aprendizaje de las mismas por parte de los usuarios. De esta idea surgirá el control barra de herramientas, que contendrá una serie de controles que diferirán poco o nada durante todo el ciclo de vida de la aplicación que lo utilice. Por último, los usuarios nos recomendaron aumentar la respuesta sonora y el uso de vibraciones en ciertos casos.

6.2 Diseño y desarrollo de la librería de controles accesibles

Una vez realizado el estudio preliminar y extraídas las pertinentes conclusiones, el siguiente paso a seguir es el diseño y desarrollo de la librería de controles accesibles que sirvan para construir aplicaciones adaptadas a personas con baja visión.

6.2.1 Diseño de la librería de controles accesibles

A continuación se detallan los controles que se diseñaron y se han desarrollado y que controles comunes sustituyen de una forma accesible:

Figura 32: LVLabel.

- LVLabel: cuadro con texto que permite leer el texto que contiene en su interior (figura 32), introduciendo un dedo en su área de la pantalla, ya sea de forma directa o mediante la localización arrastrando por la pantalla. También permite leer el texto palabra por palabra utilizando un segundo dedo pulsando en cualquier parte de la pantalla. Cada pulsación provocará qué se lea la siguiente palabra. Incluye efectos sonoros tanto en la entrada como en la salida del área del control. Sustituye a las clásicas etiquetas disponibles en la librería de controles para interfaces gráficas para la programación de aplicaciones en dispositivos con el sistema iOS.

Figura 33: LVTextField.

- LVTextField: campo de texto editable mediante teclado (figura 33). En su interior encontraremos el texto que se haya introducido y un guión parpadeando, qué indica que se puede introducir texto en ese punto. Proporciona métodos para retrasar la edición hasta el comienzo del texto o avanzar dicha edición hasta el final del texto. El usuario escuchara el texto y la posición del guión al entrar en el área del control con un dedo. Al igual que el control LVLabel, con un segundo dedo se puede leer el texto palabra a palabra mediante pulsaciones en la pantalla. El objetivo es sustituir de manera accesible a los campos de texto tradicionales a la hora de programar interfaces de usuario.


Figura 34: LVButton con el logotipo de Raylight.

- LVButton: botón para la realización de acciones y la confirmación o cancelación de eventos o sucesos (figura 34). Primero, el usuario localiza el botón con un dedo, ya sea de forma directa o mediante la localización arrastrando sobre la pantalla. El botón proporciona una respuesta sonora tanto de la entrada o salida del área del control como de su contenido. Una vez localizado, existen dos versiones de confirmar la acción disponible en el control, confirmación de selección mediante un dedo, que consiste en simplemente levantar el dedo dentro del área del control, o confirmación de selección mediante dos dedos, que consiste en pulsar sobre la pantalla con un segundo dedo, sin levantar el primero del área del

control. Cuando se confirma una selección, se le comunica al usuario con un mensaje sonoro especial. Este control es la representación accesible de los botones y también pueden ser usados para representar controles de tipo interruptor.

- LVSegmentedControl: control con múltiples opciones, sólo una de ellas seleccionable cada vez, en el cual, una vez localizado el control, ya sea de forma directa o arrastrando el dedo sobre la pantalla, podremos cambiar la opción seleccionada mediante pulsaciones con un segundo dedo en cualquier parte de la pantalla (figura 35). Para confirmar la selección, tan sólo tendremos que levantar el primer dedo. Incluye efectos sonoros tanto en la entrada como en la salida del área del control, además de la información correspondiente a cada opción. Este control trata de representar los controles de opción múltiple y aquellos controles que representen listas cortas de opciones.

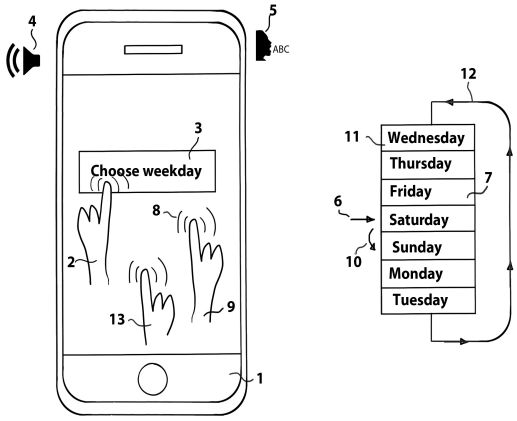


Figura 35: Gráfico de la interacción de los controles de tipo LVSegmentedControl.

- LVSegmCtrlTime: el concepto de este control es el mismo que el de el control LVSegmentedControl (figura 36). La principal diferencia entre estos controles es qué para recorrer las opciones tan sólo basta con mantener el dedo que se ha utilizado para localizar el control dentro del área del mismo. Para seleccionar una opción, tan sólo tendremos que levantar el dedo en el momento que escuchemos la opción que deseamos. Incluye efectos sonoros tanto en la entrada como en la salida del área del control, además de la información de cada una de las opciones. Este control tiene los mismos objetivos de representación que los controles del tipo LVSegmentedControl, pero de una forma mas ágil y cómoda.

Figura 36: LVSegmentedControl y/o LVSegmCtrlTime.

- LVKeyboard: teclado de tipo keypad (tal y como vemos en la figura 37, con una distribución similar a la de los teclados disponibles en teléfonos no táctiles) formado por 12 controles de tipo LVSegmCtrlTime. El usuario localiza con un dedo la tecla que desea pulsar, espera a que mediante el control temporal se recite la opción que desea teclear y, una cuando esta sea leída por el dispositivo, bastará con levantar el dedo de la pantalla para teclear la opción. Tiene 4 modos de inserción de caracteres, numérico, simbólico, mayúsculas y minúsculas. Tiene también la capacidad de mostrarse o esconderse según se necesite o no. Tiene los mismos recursos sonoros que sus

Figura 37: LVKeyboard.

componentes e incluye un efecto de vibración en la tecla representada por el dígito número 5, el cual se asocia al centro del teclado. Esta pensado como una solución eficiente para introducir datos en el dispositivo en lugar del uso de los tradicionales teclados QWERTY que aparecen en los dispositivos táctiles.

- LVWheel: control con múltiples opciones, sólo una seleccionable cada vez, en el cual, una vez localizado el control, ya sea de forma directa o arrastrando el dedo sobre la pantalla, vamos a poder recorrer estas opciones, de tal manera que, con un segundo dedo, deslizando de forma vertical o de forma horizontal, según este configurado, se nos irá comunicando la opción sobre la que nos situamos (figura 38). Si llegamos a la primera o a la última opción, se produce un efecto sonoro que marca el tope de opciones. Para seleccionar, tan sólo tendremos que levantar el primer dedo. Incluye efectos sonoros tanto en la entrada como en la salida del área del control, además de la información correspondiente a cada opción. Este control trata de representar controles de tipo deslizadera en su versión horizontal y controles que representen listas con gran cantidad de opciones en su opción vertical.

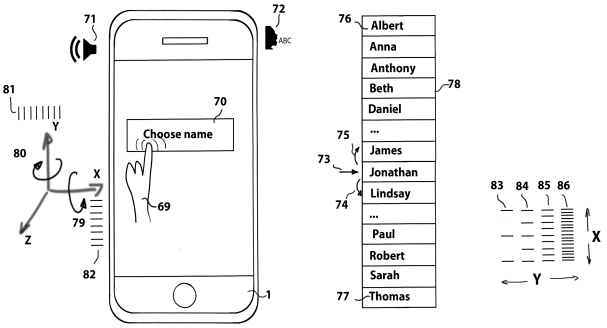


Figura 38: Gráfico de la interacción de los controles tipo LVWheel.

- LVToolBar: barra de herramientas (figura 39) compuesta de un par de controles con una labor fija (primer control comenzando por la izquierda, LVButton para volver a la pantalla anterior, segundo control, a la derecha del primero, LVLabel con la información disponible en toda la interfaz) y un par de controles configurables entre LVLabel, LVSegmCtrlTime, LVSegmentedControl o LVButton. Todos los controles cuentan con su interacción habitual. Este control trata de agrupar una serie de controles que deberían de ser comunes durante todo el ciclo de vida de una aplicación para asegurar una buena experiencia de usuario gracias a mantener una serie de opciones situadas siempre de forma estática en la interfaz.

Figura 39: LVToolBar con las opciones Home (LVButton), Info (LVLabel), espacio vacío y Tweet (LVButton).

Además de estos controles, se diseñó un control más que representaba la barra de estado disponible en los dispositivos móviles con el sistema operativo iOS, pero por motivos de infracción de las normas de diseño impuestas por Apple [25], ese tipo de controles se consideran obsoletos.

6.2.2 Desarrollo de la librería de controles accesible

Para agrupar todos estos controles, se creo una clase llamada LVControl como clase padre de todos ellos. Esta clase desciende de la clase de la librería UIKit de Apple UIControl y añade las capacidades de efectos sonoros y Text-To-Speech a los controles, gracias al uso de la librería de la compañía Acapela.

Otra clase interesante es la clase LVUtils. Todos los controles tienen definido su comportamiento según reciban eventos de tipo UITouch (eventos de la pantalla táctil), pero no saben distinguir si un determinado evento de este tipo es para un control u otro. LVUtils es la encargada de analizar y comprobar hacia quien se dirige un determinado evento de tipo UITouch, y hacer de tubería hacia el control idóneo.

Por tanto, y después de la descripción dada, la librería de controles accesibles quedaría encuadrada en la jerarquía de clases de la librería UIKit de Apple tal y como se muestra en la figura 40:

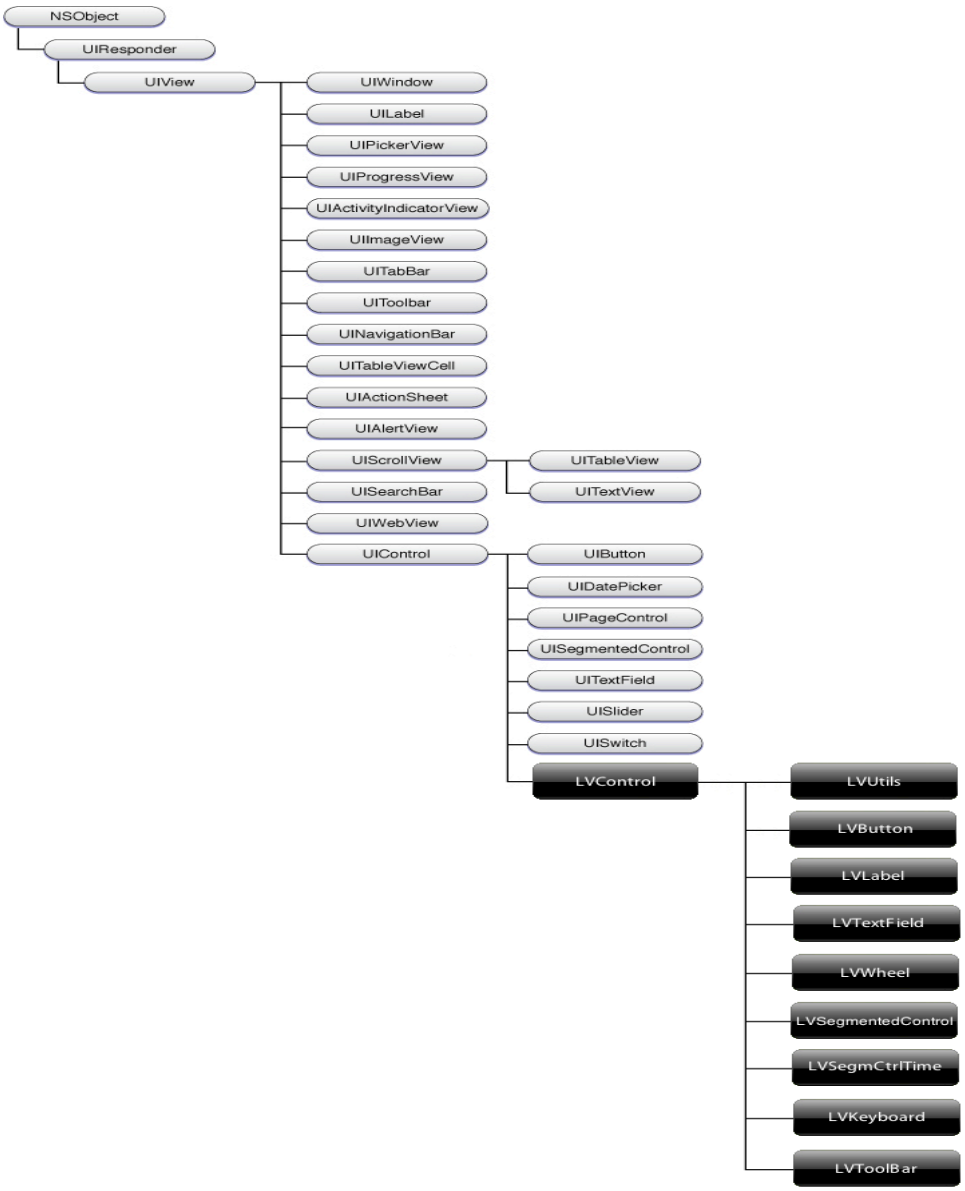


Figura 40: Jerarquía de clases después de la implementación de la librería de controles accesibles.

Para crear una clase en Objective – C es necesario el uso de dos tipos de archivos [23][26]. Al igual que en C++, las clases están divididas en dos archivos: el primero contiene el nombre de las propiedades, los atributos, las cabeceras de los métodos y otros tipos de datos, como los correspondientes a la herencia o a los mecanismos de delegación y su extensión es .h; y un segundo archivo con la implementación de la clase, cuya extensión es .m.

Debido a que esta librería es un producto comercial, no se va a mostrar código de la implementación de las clases y de los componentes creados, sin embargo, para una mejor comprensión de los archivos .h se muestra a continuación el código de la cabecera de la clase LVToolBar:

```
#import "LVControl.h"
#import "LVButton.h"
#import "LVSegmCtrlTime.h"
#import "LVSegmentedControl.h"
#import "LVWheel.h"
#import "LVLabel.h"
#import "LVUtils.h"
#import "LVspeech.h"

typedef enum {
    LVButtonConfiguration = 0,
    LVSegmentedControlConfiguration = 1,
    LVSegmentedControlTimeConfiguration = 2,
    LVWheelConfiguration = 3,
    LVLabelConfiguration = 4
} LVConfigurableControl;

typedef enum {
    LVBackControl = 0,
    LVInfoControl = 1,
    LVFirstConfigurableControl = 2,
    LVSecondConfigurableControl = 3
} LVControlInBar;

@protocol LVToolbarDelegate;

@interface LVToolBar : LVControl<LVButtonDelegate, LVSegmCtrlTimeDelegate,
LVSegmentedControlDelegate, LVWheelDelegate, LVLabelDelegate>{
    id <LVToolbarDelegate> delegate;
    LVButton *backButton;
    UIImageView *backImage;
    LVLabel *infoControl;
    UIImageView *infoImage;
    LVControl *configurableControl1;
    LVControl *configurableControl2;
    LVConfigurableControl configurableControlMode1;
    LVConfigurableControl configurableControlMode2;
    CGPoint offset;
    CGSize size;
}

@property (nonatomic, retain) LVControl *configurableControl1;
@property (nonatomic, retain) LVControl *configurableControl2;
@property (nonatomic, assign) LVConfigurableControl configurableControlMode1;
@property (nonatomic, assign) LVConfigurableControl configurableControlMode2;
@property (nonatomic, retain) UIImageView *backImage;
@property (nonatomic, assign) id <LVToolbarDelegate> delegate;

-(void)setFirstConfigurableControl:(LVConfigurableControl)configurableControlType;
-(void)setSecondConfigurableControl:(LVConfigurableControl)configurableControlType;
-(void) setSpeechSpeed:(float) speed;
-(void)setBackEnabled:(BOOL)enabled;
-(void)setInfoEnabled:(BOOL)enabled;
-(void)setBackControlInformation:(NSString*)message;
-(void)setInfoControlInformation:(NSString*)message;
-(void)setBackControlMessageEnd:(NSString*)message;
```

```
@end

@protocol LVToolbarDelegate
- (void)LVToolbar:(LVToolBar*)LVToolbar didLoad:(LVControlInBar)control
withControlType:(LVConfigurableControl)type;
- (void)LVToolbar:(LVToolBar*)LVToolbar didChooseOption:(LVControlInBar)control
withControlType:(LVConfigurableControl)type;
- (void)LVToolbar:(LVToolBar*)LVToolbar didCancel:(LVControlInBar)control
withControlType:(LVConfigurableControl)type;
@end
```

Como podemos observar, en la parte superior se importan las clases y las librerías necesarias para la implementación de la nueva clase. A continuación, se definen un par de tipos de datos de tipo enumerado. Más adelante, después de la palabra reservada `@protocol` se establece qué la clase disparará la llamada a una serie de delegados que se definirán al final de la declaración de todos los componentes de la clase. Tras la palabra reservada `@interface`, va el nombre de la clase, seguido de dos puntos y el nombre de la clase que hereda, si la hubiese. Los atributos privados van entre paréntesis después del nombre de la clase. Las propiedades se definen después de este paréntesis y con la clausula `@property`, qué creará automáticamente los métodos get y set de cada una de estas propiedades. Lo siguiente a definir serán las cabeceras de los métodos precedidos de + si son métodos de clase, es decir, no necesitan de una instancia de la clase para ser invocados, o – si son métodos de instancia, los cuales sí que necesitan una instancia de la clase para poder ser invocados. La palabra reservada `@end` indica el fin de la declaración de atributos, propiedades y métodos. Si no se implementase ningún método de delegación, el archivo de cabecera estaría completo, pero en este caso sí que se indico que iba a implementarse delegados, cuyas cabeceras podemos encontrarlas entre las clausulas `@protocol`, seguido de el nombre del delegado, y `@end`.

6.3 Pruebas de la librería de controles accesibles

Una vez desarrollados los componentes que conforman la librería de controles accesibles se desarrollaron dos aplicaciones muy sencillas que usasen estos controles. Las aplicaciones desarrolladas fueron una aplicación que envíe mensajes SMS a cualquiera de los contactos que tiene el usuario almacenados en su dispositivo y otra aplicación para navegar, consultar, editar, añadir y eliminar contactos del dispositivo. Estas aplicaciones conforman la versión Beta de la plataforma de aplicaciones accesibles *iLoowi* (figura 41). Esta versión Beta fue publicada a finales de febrero de 2012, coincidiendo con la feria del sector de dispositivos móviles MobileWorldCongress, celebrada en Barcelona, y donde se presentó la empresa Raylight y sus productos accesibles. La plataforma puede descargarse por 3'99 € desde el AppStore en sus versiones en castellano, catalán, inglés británico e inglés americano. En la fecha de redacción de este documento (junio de 2012) la plataforma continua en la versión Beta y se reciben respuestas y sugerencias de los usuarios que descargan la plataforma a través de la tienda de aplicaciones de Apple AppStore.

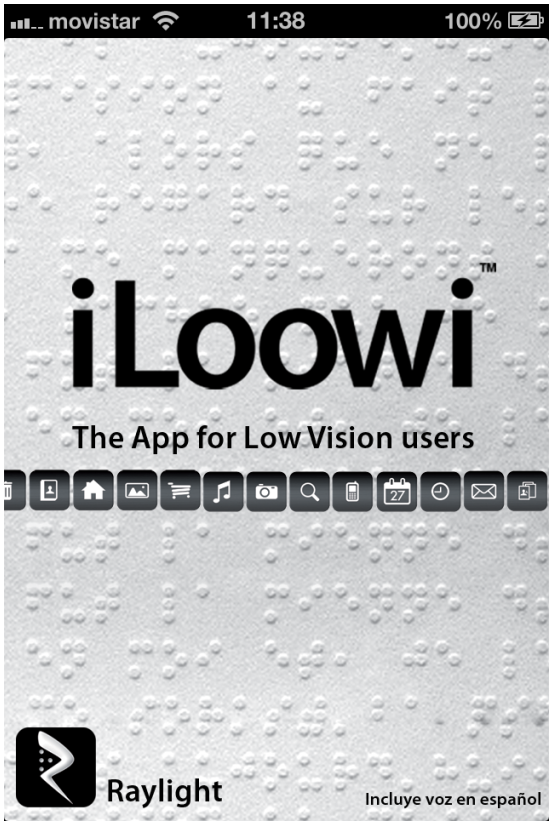


Figura 41: Plataforma iLoowi de la compañía Raylight.

Por otro lado, tanto Joaquín Selva Roca de Togores, fundador de la empresa Raylight y con un alto grado de discapacidad visual, como algunas asociaciones de discapacitados visuales cercanas a la empresa Raylight recibieron una copia gratuita de la plataforma para que realizasen pruebas sobre la misma y proporcionasen a la compañía una respuesta que ayude a mejorar tanto dicha plataforma como los controles que la conforman.

6.4 Desarrollo de un cliente de Twitter accesible

Tal y como se comenta al comienzo de este documento, el objetivo de la compañía no es sólo desarrollar aplicaciones básicas, que de mayor o menor manera ya se encuentran adaptadas a personas con baja visión, si no que el objetivo es proporcionar adaptaciones de aplicaciones más complejas y que cuenten con una gran cantidad de usuarios o con una gran utilidad para el sector de la discapacidad visual. Es por ello qué se propuso la realización de este cliente de la plataforma Twitter.

De nuevo, debido a que es un producto comercial, prácticamente no se mostrará código, aunque sí que se describirá el trabajo realizado. El patrón de arquitectura de software seguido en el desarrollo de aplicaciones para dispositivos iOS es el de Modelo-Vista-Controlador. Debido a que es los controles a utilizar pertenecen a una librería propia, la vista y los controladores de cada interfaz se combinarán en una sola clase. El modelo será el encargado de encapsular todos los componentes necesarios para realizar una comunicación constante con la red social Twitter.

Figura 42: Interfaces gráficas de usuario.

La mayoría del trabajo de desarrollo de esta aplicación se basa en la construcción de las interfaces gráficas de usuario. Haciendo uso de los controles accesibles y utilizando de guía las interfaces diseñadas en la fase de análisis, se desarrollaron las 6 interfaces que forman parte de la aplicación. Además, y para hacer más atractiva la aplicación, se encargó una iconografía al estudio Nesmanpro que siguiese la línea de colores necesaria en las aplicaciones adaptadas a personas con discapacidad visual al estudio. Esta iconografía esta diseñada en color blanco y con formato vectorial, con lo que es bastante sencillo adaptarla a los controles que lo necesiten sin que se pierda calidad en la imagen. En la figura 42 se puede observar el resultado obtenido en el desarrollo de las interfaces, empezando, de izquierda a derecha y de arriba a abajo, por la interfaz correspondiente a la clase AboutViewController, seguido de la interfaz de la clase TwitterViewController, la interfaz de la clase TwitterUserDetailsViewController, la

interfaz correspondiente a la clase TwitterFollowersViewController, la interfaz de la clase TwitterKeyboardViewController y, por último, la interfaz de la ventana de bloqueo UIBlockerView.

En cuanto al modelo, se desarrolló una clase que se encargase de obtener los datos necesarios del dispositivo acerca del usuario e hiciese de intermediario en la comunicación entre la aplicación y la red social Twitter. Esta clase se llama TwitterEngine y, debido a que en ciclo de vida de la aplicación sólo se necesita un acceso a los datos del usuario del dispositivo y un único permiso que firme las peticiones a la API de servicios REST de Twitter, se le aplicó un patrón Singleton [27] a la clase, de tal manera que en toda la aplicación sólo exista una instancia de la clase. A continuación podemos ver el código de dicho patrón:

```
+(TwitterEngine*)sharedTwitterEngine{
    @synchronized([TwitterEngine class])
    {
        if (!_sharedTwitterEngine){
            //Si la instancia no existe, la creamos
            _sharedTwitterEngine = [[self alloc] init];
        }
        return _sharedTwitterEngine;
    }
    return nil;
}
+(id)alloc
{
    @synchronized([TwitterEngine class])
    {
        //Comprueba si la instancia ya existe
        NSAssert(_sharedTwitterEngine == nil, @"Attempted to allocate a second instance
of a singleton.");
        //Crea la instancia y la almacena en una variable estática
        _sharedTwitterEngine = [super alloc];
        return _sharedTwitterEngine;
    }
    return nil;
}
-(id)copy {
    //Evita la copia de la instancia
    return self;
}
-(id)init{

    if(self = [super init]){
      //Se realiza la inicialización de las variables necesarias
    }
    return self;
}
```

Como podemos observar en el código, el primer método que encontramos es el método de clase `+(TwitterEngine*)sharedTwitterEngine{}`. Este método comprueba si hay una instancia del objeto, y si no la hay la crea. Para crearla, llama en primer lugar al otro método de clase que encontramos, el método `+(id)alloc{}`. Este método es el encargado de reservar la memoria necesaria para la creación de una instancia de la clase a la que se le aplica el patrón, en este caso TwitterEngine. En segundo lugar, el método `-(id)init{}` se encarga de inicializar las variables necesarias en la implementación de la clase. Cuando se han realizado estos dos procesos, `+(TwitterEngine*)sharedTwitterEngine{}` devuelve un objeto del tipo `TwitterEngine` o avisa de que ya hay una instancia de esa clase en la aplicación. Por otro lado, el método `-(id)copy{}` se sobrecarga para que no haga la copia del objeto, si no que devuelva el mismo objeto.

A partir de ese momento, la aplicación ya es capaz de preguntar al sistema operativo si tiene cuentas de Twitter asociadas al dispositivo. Para ello, la aplicación tiene que hacer uso de las clases contenidas en el framework Accounts y en el framework Twitter [26], disponibles en el kit de desarrollo de software para aplicaciones iOS. Si no hay ninguna cuenta asociada al dispositivo, la aplicación no permitirá el acceso a la misma. Si, al contrario, el usuario tiene asociada una cuenta de Twitter al dispositivo, la aplicación hará uso de sus datos de la cuenta (después de que el usuario acepte qué la aplicación haga uso de los mismos) para realizar consultas a la API de servicios REST de la red social Twitter. Gracias a la integración de la plataforma Twitter en los dispositivos iOS en su versión 5.0, las peticiones se firman y se realizan de forma sencilla gracias a la clase TWRequest. Sólo habrá que indicar la URL de la petición, las etiquetas y los valores de los parámetros que se quieran establecer contenidos en un diccionario y, por último, el tipo de consulta, si de tipo GET (consulta) o POST (petición). Se definieron métodos para las distintas llamadas a la API que se realizan en la aplicación, consiguiendo de esta manera encapsular todo el proceso de la llamada y tratamiento de la respuesta en la invocación a un solo método por acción. Esta clase también tiene un método para formatear las fechas de la misma manera que podemos observar en la versión web de Twitter, ya que los datos de fechas que proporcionan las respuestas a las llamadas a la API de servicios REST de Twitter están en formatos extendidos.

Ya hemos visto como se comunica el motor de Twitter con la plataforma, pero, ¿cómo informa el motor a las interfaces de que tiene los datos que ellas necesitan? Para realizar esta comunicación la clase implementa un conjunto de métodos de delegación. Estos métodos de delegación son adoptados por cada una de las clases que hacen uso del motor, y, cuando TwitterEngine finaliza alguna de sus tareas, avisa a las clases controladoras de las interfaces de el resultado de sus tareas. Si ha logrado una respuesta positiva por parte de la red social Twitter, la controladora recogerá los datos encapsulados en el motor y los distribuirá por los distintos controles de la interfaz según crea conveniente. Por ejemplo, al finalizar las distintas tareas necesarias para realizar el envío de un tweet, el motor, antes de finalizar el método que se encarga de esta acción contiene el siguiente código:

```
if(response){
  [delegate TwitterEngine:self didSendTweet:true];
}else{
  [delegate TwitterEngine:self didSendTweet:false];
}
```

Gracias a este código, las clases que implementan los métodos delegados de la clase Twitter Engine pueden realizar las tareas que consideren convenientes con el siguiente código:

```
- (void)TwitterEngine:(TwitterEngine*)engine didSendTweet:(BOOL)index{
    if(index){
    }else{
    }
}
```

Si se ha enviado el tweet, el delegado mandará un mensaje con la variable `index` a `true`, con lo que el programador sabrá que tiene los datos acerca de la acción actualizados dentro de las variables de la clase TwitterEngine. Si la variable `index` tiene el valor `false`, el programador también recibirá esa información y deberá tratarla como crea conveniente.

Gracias a estos dos mecanismos, el de comunicación con la API de servicios REST de la red social Twitter y los métodos de delegación, la clase TwitterEngine proporciona los mecanismos necesarios para lograr que las interfaces muestren de una forma eficiente y sencilla la información que se requiera en cualquier momento.

Por último, también comentar que todos los textos han sido localizados, de modo que para traducir una aplicación completa a otro idioma, basta con traducir un archivo que contiene las cadenas de texto de la aplicación. Un ejemplo de cómo se carga una `string` localizada podría ser:

```
NSLocalizedStringFromTable(@"succesfullyTweet", @"Twitter", @"")
```

6.5 Pruebas de la aplicación

Para comprobar el correcto funcionamiento de la aplicación, todo el equipo de Raylight ha ido probando la misma en un proceso paralelo al de desarrollo. De esta manera, cuando se detectaba algún error se corregía de manera casi inmediata y consolidaba la base de la aplicación sobre la cual seguir construyendo los siguientes componentes de la misma. También se han realizado pruebas por parte de Joaquín Selva Roca de Togores, propietario de Raylight y discapacitado visual, para conocer si el trabajo realizado era el idóneo o si se debían de realizar cambios o mejoras, tanto en las funcionalidades de la aplicación como en sus interfaces. Esta previsto el lanzamiento de una versión Beta de la cual se espera recibir más información acerca de posibles ampliaciones y mejoras antes de su lanzamiento oficial.

6.6 Distribución de la aplicación en el AppStore

Aunque por motivos de marketing la aplicación aún no ha sido lanzada al mercado, es importante para la redacción de este documento el describir como es la tarea de publicar una aplicación en la tienda de aplicaciones para dispositivos móviles de Apple AppStore.

Para poder publicar aplicaciones en el AppStore es necesario tener una cuenta de desarrollador válida en el momento de la carga de la aplicación. Si se dispone de dicha cuenta, los pasos a seguir son los siguientes:

1. Acceder a la plataforma iTunes Connect con tu cuenta de usuario.
2. Crear una nueva aplicación pulsando el botón “Add new app”.
3. Rellenar los datos de la aplicación.
4. Aplicar un precio a la aplicación estableciéndola en uno de los niveles ofrecidos.
5. Rellenar los metadatos de la aplicación, subir las capturas de pantalla que se desean publicar de la aplicación y el icono extendido que se presentará en el AppStore.
6. Subir la aplicación.

Una vez realizado todos estos pasos y aceptados los términos y condiciones de uso, la aplicación pasará al estado “Waiting for review”. En aproximadamente 15 días Apple revisará la aplicación, comprobará que no transgrede ninguna de sus normas de desarrollo de aplicaciones para dispositivos móviles y aprobará o no la aplicación para su distribución. Si la aplicación no es aceptada enviarán un comunicado detallando el porqué del rechazo. Este comunicado puede ser rebatido, pero una vez recibido se establece un periodo máximo para volver a cargar la aplicación de 5 días laborables o se deberá de repetir el proceso desde el inicio. Si la aplicación se vuelve a subir con las modificaciones comunicadas dentro del plazo, la aplicación será revisada en aproximadamente 3 días desde la recepción de la misma. En cuanto se superen todas estas fases, la aplicación aparecerá como “Ready for sale” y estará disponible en el AppStore.

6.7 Ampliaciones

Aunque se ha intentado cubrir la mayor cantidad de funcionalidades disponibles en la red social Twitter, en esta primera versión no se han implementado todas las opciones que la plataforma proporciona. En siguientes ampliaciones sería interesante poder proporcionar información sobre los "Trending Topics" (temas de tendencia) o poder consultar los mensajes directos.

Por otro lado, la aplicación ha sido desarrollada en 4 idiomas, con lo que también se debe de considerar la opción de traducir los textos que utiliza a una cantidad mayor de idiomas, apuntando prioritariamente a los mercados con mayor posibilidad de adaptación del sistema.

Por último, pero no menos importante, Raylight escucha a sus usuarios, ya que los productos están enfocados a un público muy especifico, con lo que cualquier modificación, ampliación, sugerencia o crítica recibida acerca de los controles diseñados y las interfaces desarrolladas son tenidas en cuenta en futuros desarrollos y en posibles mejoras, ya que ayudan a la compañía a ofrecer mejores productos a sus clientes.

7. Conclusiones

Las nuevas tecnologías proporcionan una serie de herramientas destinadas a facilitar la vida a los usuarios de las mismas. Sin embargo, estas tecnologías están en muchas ocasiones pensadas únicamente en las personas que no sufren ninguna discapacidad, dejando a las personas que sufren algún tipo de discapacidad aisladas.

Las redes sociales son una herramienta de software muy potente gracias a la cantidad de conexiones que permiten y la instantaneidad de las acciones que en ellas se realizan, pero no están pensadas para personas que sufren algún tipo de discapacidad. El acceso a las mismas resulta misión imposible para los usuarios que tienen problemas de baja visión, con lo que, directamente, no intentan involucrarse en ese tipo de tecnología.

Por otro lado, los dispositivos móviles han supuesto una revolución a la hora de comunicarnos con otras personas, ya que ahora no sólo se llama y se envían mensajes, si no que también se envían emails, se ponen comentarios en foros o realizan video-llamadas sin necesidad de tener un ordenador cerca. Algunos fabricantes de estos dispositivos sí que hacen intentos por adaptar esta tecnología a personas discapacitadas, pero por motivos comerciales, estas adaptaciones no resultan suficientes ya que no proporcionan herramientas específicas si no que más bien son pequeñas transformaciones de lo inicialmente desarrollado.

En este proyecto se establecieron una serie de requisitos de diseño y funcionalidad que se han alcanzado, produciendo una herramienta que se espera que tenga una gran aceptación en el mercado, ya que intenta superar la barrera del acceso a redes sociales y el uso de dispositivos móviles por personas con cualquier tipo de discapacidad visual. Además, el proyecto ha cubierto todas las fases del desarrollo de la aplicación, desde las fases mas tempranas de análisis, pasando por el diseño de las interfaces, el desarrollo de la aplicación e incluyendo las guías de cómo publicar la aplicación en el mercado de aplicaciones AppStore.

La realización de este proyecto ha supuesto un largo e intenso aprendizaje, tanto a nivel técnico y comercial como a nivel personal. Técnicamente, lo primero que he aprendido es a desarrollar aplicaciones para dispositivos iOS, cosa que desconocía al empezar esta aventura. También he aprendido la importancia de la experiencia de usuario a la hora de desarrollar aplicaciones que tienen que competir en un mercado de más de 500000 aplicaciones. Otra de los aspectos que puedo destacar es que he mejorado mis destrezas como programador y como ingeniero del software. A nivel comercial, he aprendido cómo funciona el mundo de las “startups”, cómo ser creativo a la hora de desarrollar estrategias de marketing efectivas, y, en definitiva, cómo actuar frente a posibles clientes para intentar convencerlos de que compren tu

producto. Considero este aprendizaje muy importante debido a que, hoy en día, y en el contexto de crisis en el que nos encontramos, no sólo es necesario tener un buen producto software, si no que también hay que saber venderlo para que este tenga éxito y sea rentable. A nivel personal puedo decir que he aprendido mucho acerca de la discapacidad, y más en concreto, de la discapacidad visual. He comprobado el espíritu de lucha que tienen las personas que sufren algún problema de visión y que tratan de superar su discapacidad orientándola como un cambio que puede suponer una oportunidad, en vez de pensar en ella como un problema.

Como reflexión final me gustaría decir que la accesibilidad en productos software es un factor el cual todos los que nos dedicamos a desarrollar este tipo de herramientas deberíamos de tener más en cuenta por varios motivos: el primero y más importante, por el hecho de ayudar y facilitar la utilización de nuestros productos a personas que de otra manera no podrían tener acceso a las tecnologías que desarrollemos; en segundo lugar, por motivos económicos, ya que son mercados con gran cantidad de usuarios, apoyados en muchas ocasiones por grandes asociaciones, y en los cuales no existe una gran competencia; y, por último, porque la gran mayoría de personas con discapacidad no sufren sus problemas de nacimiento, si no que son el resultado de algún tipo de enfermedad, con lo que las bases que sentemos hoy pueden ser necesarias para nosotros mismos mañana.

Anexo I. Manual de usuario

En este apartado se va a detallar como se usa la aplicación desarrollada. Para poder utilizar la misma, una de las condiciones que el usuario debe cumplir es que esté registrado en la red social Twitter. A continuación se procede, en primer lugar, a explicar como registrar al usuario en la red social Twitter si este no tiene cuenta, y, en segundo lugar, como hacer uso de la aplicación una vez el usuario tenga asociada su cuenta de Twitter a su dispositivo.

Si no se tiene cuenta de usuario de Twitter

1. Entramos en los ajustes del dispositivo y seleccionamos la opción de Twitter.

2. Una vez accedida a la ventana de los ajustes de Twitter, en la parte inferior de la pantalla encontraremos un botón con la etiqueta "Crear nueva cuenta".

3. Al pulsarlo se nos solicitará una serie de datos que deberemos de rellenar.

4. Una vez completados todos los datos, tendremos que pulsar el botón de registrar.

Si se dispone de cuenta de Twitter

La filosofía de interacción de la aplicación es distinta a la tradicional, pero bastante sencilla de comprender. La idea es utilizar dos dedos para interactuar con la aplicación, excepto cuando usemos el teclado. El primer dedo servirá para explorar la interfaz. Esta ofrecerá una respuesta sonora en todo momento, sea con efectos sonoros, sea con la lectura de información. Si queremos confirmar la acción que hemos escuchado en un botón, deberemos pulsar en cualquier parte de la pantalla con un segundo dedo. Si queremos recorrer la lista de opciones de un control rueda, deslizaremos arriba y abajo un segundo dedo por la pantalla. Para escribir con el teclado, sólo necesitamos un dedo, el cual posicionaremos sobre la tecla que contenga la opción que deseamos introducir, esperaremos a que esta se recite, y, cuando esto suceda, levantaremos el dedo de la pantalla para confirmar la selección del carácter.

A continuación se muestran todas las opciones de la aplicación mediante capturas de pantalla con los convenientes comentarios para que sean entendidas más fácilmente:

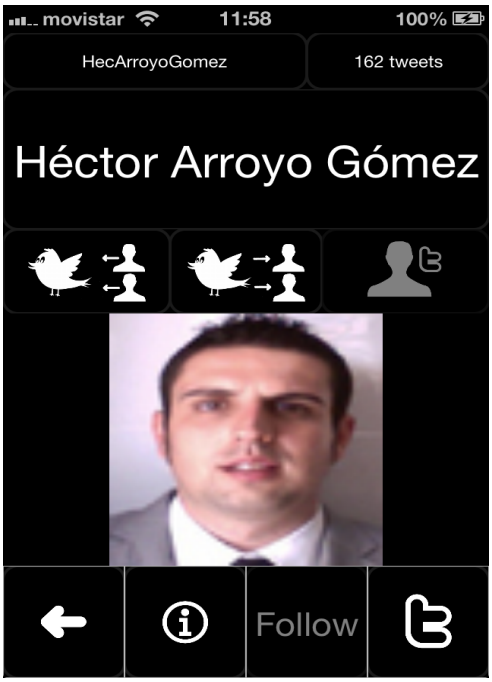
1. Una vez tenemos cuenta de Twitter, comprobamos que esté asociada al dispositivo.

2. Antes de entrar en la aplicación, debemos de comprobar que esta activado el sonido y que tenemos red.

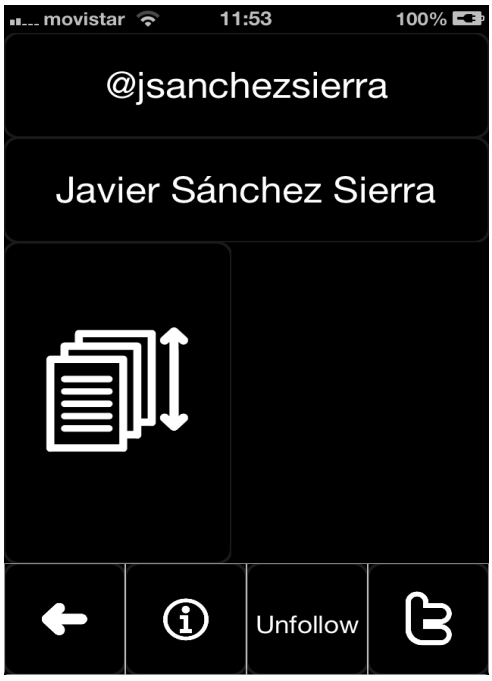
3. Accedemos a la aplicación. La primera interfaz que encontraremos es la correspondiente al timeline personal. En la parte superior de la pantalla se encuentran tres etiquetas, una con el nombre del autor del tweet, otra con el tiempo que hace que el tweet ha sido enviado, y debajo de estas dos el texto del tweet. Debajo del tweet hay tres botones, de izquierda a derecha: el primer botón es el de "Retweet", el cual introduce el tweet seleccionado en nuestro timeline. El segundo botón es el de "Responder". El último botón es el botón "Refrescar". Más abajo, al lado izquierdo, encontrarás el control rueda. Si pulsas con un dedo y deslizas arriba y abajo con un segundo dedo se mostrarán los tweets cargados en el timeline. A la derecha del control rueda encontraremos una etiqueta que indica en que timeline te encuentras. Por último, en la barra de herramientas encontrarás el botón "Acerca de", la etiqueta de información, el botón "Detalles" y el botón "Tweet".

4. Si pulsamos en el nombre de usuario asociado a un tweet de nuestro timeline, accederemos al timeline de ese usuario. Las opciones serán las mismas que las asociadas al timeline personal, pero afectando al usuario a cuyo timeline se ha accedido. El botón "Acerca de" será sustituido por el botón "Atrás" y se actualizará la etiqueta que informa de cual es el timeline en el que el usuario se encuentra.

5. Si pulsamos el botón "Acerca de", accederemos a la interfaz que muestra información acerca de la empresa y del equipo de diseño gráfico, además de los datos de contacto. En la parte inferior hay un botón para volver a la pantalla anterior.

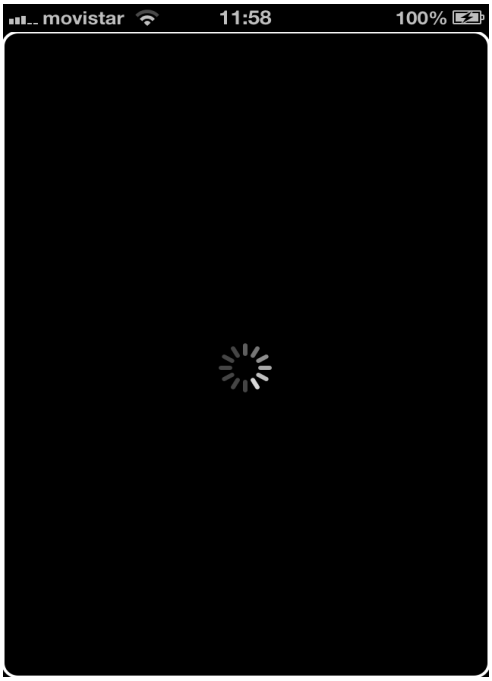


6. Si pulsamos el botón de "Detalles" se abrirá la ventana de detalles de usuario. En la parte superior de la pantalla se encuentran dos etiquetas. En la primera se muestra el nombre de usuario de twitter y en la segunda el número de tweets que el usuario ha escrito desde que pertenece a la red social. Justo debajo de estas etiquetas encontraremos una tercera etiqueta con el nombre extendido del usuario. En la zona central se encuentran tres botones. De izquierda a derecha, el primero nos enlaza con la lista de los folowers del usuario, el segundo botón enlaza con la lista de usuarios que están siguiendo al usuario (*following*) y el tercer botón es el de "Mensaje Directo"(solo es posible si este usuario te sigue). Debajo de estos botones encontraremos la foto del usuario. Bajo del todo encontraremos la barra de herramientas con el botón "Atrás", la etiqueta de información de la interfaz, el botón follow/unfollow para seguir o dejar de seguir al usuario y el botón "Tweet".



7. Si desde la ventana de detalles de un usuario accedemos a la lista de seguidores de un usuario o a la lista de usuarios que siguen al usuario, aparecerá la interfaz que se muestra. En la parte superior de la pantalla encontrarás dos etiquetas, una con el nombre de un usuario y otra con su nick dentro de la red social, con la @ antes del nombre. Debajo de ellas encontrarás el control rueda para navegar por la lista de usuarios, sea de seguidores o de usuarios que te siguen. Esta lista estará ordenada temporalmente por la fecha en la que se estableció la relación. En la parte inferior de la pantalla encontrarás la barra de herramientas con el botón "Atrás", la etiqueta de información, el botón de follow/unfollow al usuario v el botón "Tweet".

8. Si en algún punto de la aplicación hemos pulsado el botón "Tweet", "Reply" o "Mensaje Directo" se abrirá el teclado. En la parte superior se encuentra el campo de texto que editable.. Mediante los botones que tienes a ambos lados del texto, se puede desplazar el cursor de edición palabra a palabra por todo el texto. El resto de la pantalla está compuesto por un teclado numérico que consta de 3 filas y 4 columnas. Cada número tiene asociadas 3 o 4 letras. Alguno de los botones del teclado no contiene letras, sino comandos que te permitirán borrar una letra, o bien cambiar el modo de teclado a numérico, o simbólico. En la barra de herramientas, situada en la parte inferior de la pantalla, encontraremos, de izquierda a derecha, el botón "Cancelar", la etiqueta de información de la interfaz, la etiqueta con el número de caracteres restantes y el botón "Aceptar". Como referencia, mientras el usuario desliza su dedo por el teclado, podrá notar que existe una vibración en el número 5, marcándole que está en el centro de la pantalla.



9. Todas estas interfaces podrán encontrar intercalada la interfaz de bloqueo. Esta interfaz aparecerá cada vez que la aplicación ejecuta tareas de comunicación con la red social Twitter o de carga de datos en la interfaz para evitar conflictos por la ausencia de datos.

Anexo II. Referencias

[1] http://www.mediabistro.com/alltwitter/social-media-users_b22556

[2] http://www.computing.es/soluciones/tendencias/1060096002801/gartner-mercado-mundial-dispositivos-moviles.1.html

[3] http://www.who.int/mediacentre/factsheets/fs282/es/

[4] http://support.google.com/ics/nexus/bin/answer.py?hl=en&answer=2492341

[5] http://www.apple.com/accessibility/iphone/vision.html

[6] http://www.apple.com/iphone/features/siri.html

[7] http://en.wikipedia.org/wiki/Nuance_Communications#Partnership_with_Siri_and_Apple_Inc.

[8] http://www.apple.com/

[9] http://developer.apple.com/xcode/

[10] http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

[11] http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C

[12] http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html
http://es.wikipedia.org/wiki/Objective-C

[13] http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/DesignPatterns/StreamlineYourAppswithDesignPatterns/StreamlineYourApps/StreamlineYourApps.html
http://developer.apple.com/library/ios/#documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html#//apple_ref/doc/uid/TP40010810-CH14-SW1

[14] http://developer.apple.com/library/ios/#releasenotes/General/RN-iOSSDK-5_0/_index.html

[15] http://developer.apple.com/library/ios/#documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/00_Introduction.html#//apple_ref/doc/uid/TP40010188

[16] http://es.wikipedia.org/wiki/Representational_State_Transfer
http://dev.twitter.com/docs/api

[17] http://www.json.org/json-es.html

[18] http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

[19] http://developer.apple.com/programs/

[20] http://www.acapela-for-iphone.com

[21] http://dev.twitter.com/

[22] http://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

[23] Curso de desarrollo de aplicaciones para iPhone y iPad. Paul Hegarty, Stanford University, Fall 2011

[24] Apuntes de Ingeniería del Software de Gestión. María Carmen Penades Gramage, Universidad Politécnica de Valencia, 2010-2011

[25] iOS Human Interface Guidelines. Apple Inc. 2012

[26] Apuntes del curso de especialista en desarrollo de aplicaciones para dispositivos móviles. Miguel Ángel Lozano, Javier Aznar, Universidad de Alicante, 2011-2012

[27] Foro de programadores StackOverflow. Múltiples autores, http://stackoverflow.com