UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Gºen Ingeniería en Informática

Trabajo final del GºIng.Informática:

Despertador Android con lectura de eventos

Presentado por Álvaro Pérez Delgado
en Julio de 2016
Tutor César Represa Pérez

Índice de contenido

Índice de ilustraciones

Resumen

El propósito de este trabajo ha sido el desarrollo de una aplicación para dispositivos móviles Android que pueda utilizarse como despertador o avisador de modo que a la hora determinada por el usuario la aplicación proceda a la lectura de ciertos eventos programados. Estos eventos se seleccionan de una lista definida en la aplicación y que incluye la lectura de un mensaje personalizado a tipo recordatorio que podrá configurar a su gusto, un archivo de audio o la lectura del buzón de correo, además de la información del tiempo en una ciudad seleccionada y de los eventos del calendario de tu móvil. También podremos disponer de toda esta información hablada con solo pulsar un botón.

La aplicación se ha llevado a cabo utilizando el entorno Android Studio y Java como lenguaje de programación.

Palabras clave:

Alarma, aplicación, Android, móvil, correo electrónico, calendario, información meteorológica.

Abstract

The target of this work has been the development of an application for Android mobile devices that can be used as an alarm clock or timer so that when the user determines the time, the application starts to read properly certain events scheduled. These events are selected from a defined in the application list that includes reading a custom reminder message type that you can configure to your liking, an audio file or reading your emails besides , you can see the weather information in the city that you have choosen and your mobile's timetable events. You also can have all of this information as a audio just pressing a button . The application has carried out using Android Studio environment and Java programming language .

I - *INTRODUCCIÓN*

Actualmente la mayoría de nuestros dispositivos móviles disponen de un asistente por voz, los más conocidos son Siri en Apple, Cortana en Microsoft y Google Now en Android. Esto no solo nos hace pensar que nuestro dispositivo está "vivo", sino que nos ayuda en momentos en los que no podemos leer, como en el coche, o incluso ayuda a personas con discapacidades a hacer su vida algo más fácil.

Los móviles ya no son lo que eran y los podemos usar para muchas más cosas, hoy en día ya no es necesario estar leyendo toda la información sino que podemos escucharla, y así no tener que estar mirando a nuestra pantalla constantemente. Y ya si lo primero que escuchamos al despertarnos, nos ayuda a tener el día un poco mejor planificado, en vez de levantarnos con un sonido infernal, pues ahorramos tiempo y salud.

Este proyecto se centra en la información meteorológica, los eventos de nuestro calendario y los Correos electrónicos, aunque también podemos escuchar un mensaje motivador o incluso una canción que se encuentre en nuestro dispositivo.

II - OBJETIVOS DEL PROYECTO

1. Objetivos del proyecto

Tras varias reuniones con el tutor y varias ideas encima de la mesa, se establece como objetivo principal el desarrollo de una aplicación Android consistente en una alarma programable que al activarse nos proporcione información sobre algunos acontecimientos importantes del día.

Esto implica que nuestra aplicación tendrá que obtener información de nuestro calendario, para planificar las cosas que tenemos que hacer desde que nos despertamos, también tendrá que obtener la información meteorológica para prepararnos a como hace en la calle, también tendrá que obtener los mensajes que tenemos en nuestro Email, la bandeja de entrada, para saber si tenemos que responder a algo urgente.

2. Objetivos técnicos generales

Para la consecución de este trabajo se han fijado los siguientes objetivos técnicos:

- Estudiar como se realiza una aplicación en Android y el funcionamiento de su entorno de desarrollo, en nuestro caso Android Studio.
- Investigar las posibilidades en cuanto a herramientas de prototipado.
- Estudiar el lenguaje, librerías y herramientas que se necesiten para el desarrollo de esta aplicación.
- Analizar las ventajas e inconvenientes que nos ofrece Android.
- Examinar las opciones que nos ofrece Android Studio para la internacionalización, para poder incorporar fácilmente textos en otro idioma.

- Valorar las versiones que tiene Android y ver cuales consideraremos para que sean compatibles.
- Aplicación de metodologías ágiles para el desarrollo del proyecto, concretamente Scrum.
- Control de versiones: Uso de GitHub como software de control de versiones y repositorio público del proyecto.
- Control de iteraciones y tareas: Uso de un gestor de incidencias para el registro de las diferentes tareas y cambios que componen el desarrollo ágil, en este caso Trello.

3. *Objetivos software de la aplicación*

Los objetivos específicos marcados por los requisitos software de la aplicación son:

- Se podrá poner una alarma para una hora deseada.
- Los usuarios deben poder escuchar la información que seleccionen cuando llegue la hora que hayan establecido con anterioridad.
- Los usuarios podrán escuchar, pulsando un un botón, en el instante que quieran la información antes mencionada.
- También podrán visualizar en la pantalla principal la información del tiempo, los Emails de la bandeja de entrada y los eventos de ese día.
- Habrá un espacio reservado dentro de la aplicación para enseñar al usuario a utilizarla.

III - Conceptos teóricos

El presente trabajo consiste en el desarrollo de una aplicación para dispositivos móviles con S.O. Android. Así, en este apartado se expondrán una serie de conceptos relacionados con el desarrollo y planificación de aplicaciones e interfaces.

1. *Android*

Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tablets o tabléfonos.

Android, al contrario que otros sistemas operativos para dispositivos móviles como iOS o Windows Phone, se desarrolla de forma abierta y se puede acceder tanto al código fuente como a la lista de incidencias donde se pueden ver problemas aún no resueltos y reportar problemas nuevos.[1]

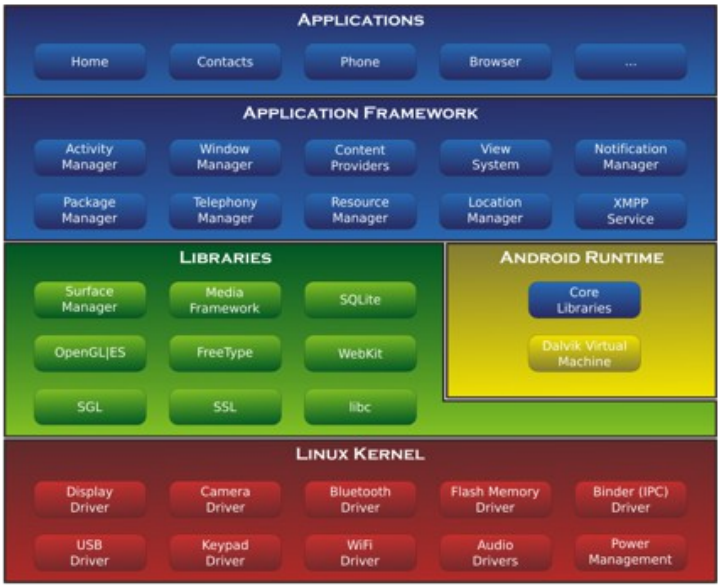

Ilustración 1: Arquitectura de Android

En cuanto a su arquitectura, podemos definir los siguientes niveles:

Aplicaciones: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java. [2] [3]

Marco de trabajo de aplicaciones: los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

Bibliotecas: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.

Runtime de Android: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente.

Núcleo Linux: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

Las características más importantes de Android son las siguientes:

Diseño de dispositivo	La plataforma es adaptable a pantallas de mayor resolución, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0 y diseño de teléfonos tradicionales.
Almacenamiento	SQLite, una base de datos liviana, que es usada para propósitos de almacenamiento de datos.
Conectividad	Android soporta gran numero de diferentes tecnologías de conectividad.
Mensajería	SMS y MMS son formas de mensajería, incluyendo mensajería de texto y ahora la Android Cloud to Device Messaging Framework (C2DM) es parte del servicio de Push Messaging de Android.
Navegador web	El navegador web incluido en Android está basado en el motor de renderizado de código abiertoWebKit, emparejado con el motor JavaScript V8 de Google Chrome.
Soporte de Java	Aunque la mayoría de las aplicaciones están escritas en Java, no hay una máquina virtual Java en la plataforma. El bytecode Java no es ejecutado, sino que primero se compila en un ejecutable Dalvik y corre en la Máquina Virtual Dalvik. Dalvik es una máquina virtual especializada, diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.
Soporte para hardware adicional	Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, sensores de luz, gamepad, termómetro, aceleración por GPU 2D y 3D.
Entorno de desarrollo	Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software. Inicialmente el entorno de desarrollo integrado (IDE) utilizado era Eclipsecon el plugin de Herramientas de Desarrollo de Android (ADT). Ahora se considera como entorno oficial Android Studio, descargable desde la página oficial de desarrolladores de Android.
Google Play	Google Play es un catálogo de aplicaciones gratuitas o de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.
Multitarea	Multitarea real de aplicaciones está disponible, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj.
Tethering	Android soporta tethering, que permite al teléfono ser usado como un punto de acceso alámbrico o inalámbrico.

Ilustración 2: Características principales de Android

1.1. Diseño mediante Activities

Podemos decir que todas las pantallas de una aplicación son una “activity”. Más adelante vamos a ver que existen algunas variaciones, pero por ahora digamos que todas lo son. Es decir, que si una aplicación tiene cinco pantallas, tiene 5 “Actividades” o *activities*.

Las activities están conformadas por dos partes: la parte lógica y la parte gráfica.

La parte lógica es una archivo .java que es la clase que se crea para poder manipular, interactuar y colocar el código de esa actividad.

La parte gráfica es un XML que tiene todos los elementos que estamos viendo de una pantalla declarados con etiquetas parecidas a las del HTML, es decir, que el diseño de una aplicación en Android se hace similar a una página web; XML es un primo de HTML.

Resumiendo, una actividad está conformada por la parte lógica (un archivo Java) y la parte gráfica (un archivo XML).

Adentrando más en el tema, ya sabemos que tenemos un archivo .java, esto quiere decir que tenemos una clase principal, al ser una actividad extiende de la clase Activity (por eso el nombre) que nos proporciona Android para crear actividades con sus métodos asignados.[4]

1.2. Diseño mediante fragments

Los *fragments* son componentes que funcionan dentro de una *activity*. Surgen como solución al problema que presentaba la incorporación de dispositivos de gran tamaño como las tablet, que dificultaban la adaptación de interfaces de usuario diseñadas para teléfonos a estos dispositivos.

Podría definirse como una porción de la interfaz de usuario que puede añadirse o eliminarse de forma independiente al resto de elementos que se encuentran en la *activity* y que puede ser reutilizado por otras *activities*. Esto permite dividir la interfaz en varias porciones de forma que se pueden diseñar distintas configuraciones según el tamaño de la pantalla. El ciclo de vida de un *fragment* esta ligado directamente al de la *activity* en la que esta contenido. Este diseño mediante fragments solo lo hemos utilizado para las pestañas principales del menú, para el resto de activities, como puede ser la inserción de datos se ha optado por el uso de activities normales.

El menú principal se ha hecho mediante unos fragments especiales llamados tabFragments que sirven para hacer de menú inferior que queda bastante bien.

2. Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como *WORA*, o "*write once, run anywhere*"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.\l
Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente. [6]

En el caso concreto de Android, y como ya se ha visto en la tabla de características de Android, el bytecode Java no es ejecutado, sino que primero se compila en un ejecutable Dalvik y corre en la Máquina Virtual Dalvik especializada, diseñada y optimizada específicamente para Android. [5]

Página web de la herramienta:

https://www.oracle.com/java/index.html

3. Interfaz gráfica de usuario

Es un tipo de interfaz que proporciona un entorno visual sencillo, permitiendo al usuario interactuar de forma amigable con el sistema operativo de dispositivos electrónicos a través de un lenguaje visual.

Surge como evolución de las interfaces de línea de comandos, las cuales provocan un lento aprendizaje del usuario ya que requieren el conocimiento de los comandos que deben ser introducidos por teclado.

Este proyecto necesita tener una interfaz gráfica amigable para el fácil manejo de la aplicación. [6]

4. Aplicación móvil

Una aplicación móvil o app (en inglés) es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. Por lo general se encuentran disponibles a través de plataformas de distribución, operadas por las compañías propietarias de los sistemas operativos móviles como Android, iOS, BlackBerry OS, Windows Phone, entre otros. Existen aplicaciones móviles gratuitas u otras de pago, donde en promedio el 20-30% del costo de la aplicación se destina al distribuidor y el resto es para el desarrollador.

El desarrollo de aplicaciones para dispositivos móviles requiere tener en cuenta las limitaciones de estos dispositivos. Los dispositivos móviles funcionan con batería, hay que considerar una gran variedad de tamaños de pantalla, datos específicos de software y hardware como también distintas configuraciones. El desarrollo de aplicaciones móviles requiere el uso de entorno de desarrollo integrados. [7]

Tipos de App:

– Nativas: Las aplicaciones móviles nativas son las que se desarrollan específicamente para cada sistema operativo, iOS, Android o Windows Phone, adaptando a cada uno el lenguaje con el que

se desarrolla: lenguaje Objective-C para iOS, Java para Android, y .Net para Windows Phone. [3]

– Web: Las aplicaciones móviles web se desarrollan con lenguaje Javascript, CSS o HTML. A diferencia de las aplicaciones nativas, la aplicación web es compatible, se adapta, a cualquier sistema operativo, por lo que no tiene que desarrollarse una app para cada uno como sucede con el caso anterior. Asimismo, se adapta al navegador móvil utilizado por el dispositivo. [3]

– Híbridas: Se llaman híbridas porque combinan aspectos de las aplicaciones nativas y de las aplicaciones web según más convenga. Por un lado, se desarrollan bajo lenguaje Javascript, CSS o HTML, al igual que las apps web, lo cual permite la adaptación a cualquier sistema operativo; y por otro lado, como sucede con las apps nativas, permiten el acceso a las funcionalidades del dispositivo. [3]

En nuestro caso, el objetivo del proyecto es el de desarrollar una aplicación móvil nativa para Android.

5. Entornos de desarrollo integrados

Un ambiente de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software. Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense). Algunos IDE contienen un compilador, un intérprete, o ambos, tales como NetBeans y Eclipse; otros no, tales como SharpDevelop y Lazarus. El límite entre un IDE y otras partes del entorno de desarrollo de software más amplio no está bien definido. Muchas veces, a los efectos de simplificar la construcción de la interfaz gráfica de usuario (GUI, por sus siglas en inglés) se integran un sistema controlador de versión y varias herramientas. Muchos IDE modernos también cuentan con un navegador de clases, un buscador de objetos y un diagrama de jerarquía de clases, para su uso con el desarrollo de software orientado a objetos. [7]

IV - Técnicas y herramientas

1. Técnicas metodológicas

Para este proyecto se han utilizado técnicas de desarrollo ágil, basada en reuniones con el tutor cada semana aproximadamente, en la cuál se entregaba un producto nuevo.

1.1. Prácticas del proceso de desarrollo ágil

A continuación se detallara un conjunto de buenas prácticas para el proceso de desarrollo ágil en un proyecto.\l

1.1.A. Refactorizaciones

Una aplicación bien refactorizada tiene mayor valor que una mal refactorizada. Mediante tareas de refactorización se buscará mejorar el código evitando defectos como métodos largos, código duplicado, nombres de variables poco descriptivos, etc.

A lo largo del desarrollo del proyecto se han realizado refactorizaciones como por ejemplo el renombrado de variables o clases para facilitar su entendimiento.

1.1.B. Control de versiones

Es importante hacer uso de un sistema de control de versiones donde almacenar todas las partes del proyecto como: código, documentación, scripts de construcción, pruebas, etc. Es muy útil en el caso de que queramos volver a una versión anterior porque nuestro código tenga problemas. No se debe guardar código en el control de versiones que no compile ni pase las pruebas.

Para el control de versiones del proyecto se ha usado Github. [8]

El repositorio es accesible desde la siguiente dirección:
https://github.com/apd0025/AlarmaHablada

1.1.C. Comunicación

Dado que el método de desarrollo elegido ha sido ágil la comunicación es fundamental para poder llevar a cabo las iteraciones periódicas.

Debe existir un canal de comunicación definido entre los desarrolladores y los clientes. Esta comunicación puede ser: reuniones en persona por demanda, reuniones en persona diarias o semanales, contacto telefónico, mensajería instantánea, por correo electrónico, etc.

La comunicación en el proyecto se ha realizado a través de reuniones periódicas cada semana de forma habitual, también ha existido comunicación a través de Mail y de la herramienta Trello.

1.1.D. Control de iteraciones y tareas

El proyecto debe contar con un sistema definido para controlar las diferentes iteraciones, tareas y fallos. El sistema debe permitir asignar las tareas a los diferentes desarrolladores.

Para gestionar las iteraciones y el trabajo que se debía realizar en cada una de ellas se ha ido asignando a través de la aplicación Trello y sus tablones y tarjetas. [9]

1.1.E. Buena documentación de código

Los comentarios de código deberían tener la misma calidad que el propio código. Se debe hacer todo lo posible para asegurarse que no se necesita documentación técnica complementaria para comprender el código.

1.1.F. Trabajo en curso

Cada vez que se completa una iteración o tarea se debe publicar una versión funcional del producto para que el cliente pueda probar los cambios y con ello comprobar si el resultado es el esperado. Realizando las tareas de esta forma conseguimos ahorrar tiempo y costes, ya que el cliente puede ver constantemente como va evolucionando el producto.

Cada una de las tareas del proyecto han sido cerradas con un commit que contiene una versión funcional de la aplicación.

1.1.G. Mecanismos de realimentación

Se necesitan mecanismos para que los miembros del proyecto y el cliente puedan exponer sus opiniones sobre como va el proyecto y el estado de satisfacción.

En nuestro caso lo hacíamos a través de reuniones cada vez que se finalizaba una iteración.

2. Herramientas de desarrollo

2.1. Android Studio

Este es el IDE que se ha utilizado para el proyecto, por recomendación del tutor, es muy similar a eclipse así que el cambio fue relativamente fácil. Al ser el entorno oficial de google pues tiene bastante documentación y la mayoría de desarrolladores utiliza este entorno.

Android Studio es un entorno de desarrollo integrado para la plataforma Android. Es publicado de forma gratuita a través de la Licencia Apache 2.0, y está disponible para Windows 2003, Vista, 7, 8 y GNU/Linux, tanto plataformas de 32 como de 64 bits, Linux con GNOME o KDE y 2 GB Ram mínimo y Mac OS X, desde 10.8.5 en adelante. [10]

Las opciones que nos ofrece así como su utilización se verán en el manual del programador (Anexo IV).

Características:

- Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Renderización en tiempo real.
- Soporte para construcción basada en Gradle.
- Refactorización especifica de Android y arreglos rápidos.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear.

Requisitos del sistema:

- 2 GB de RAM (4 GB recomendados)
- 400 MB de espacio en disco
- 1 GB para Android SDK
- Monitor de 1280x800
- Java Development Kit 7\J

Web de la herramienta:

http://developer.android.com/intl/es/sdk/index.html\J

2.2. AVD Manager

Al principio hacia las pruebas de interfaz con este simulador, pero si no tienes un ordenador muy potente tarda mucho tiempo en arrancar, así que con el tiempo me decidí por hacer la mayor parte de las pruebas en un terminal Android.

El Administrador de AVD proporciona una interfaz gráfica de usuario en el que se pueden crear y gestionar dispositivos virtuales Android (AVDS) en los que probar las aplicaciones. Estos emuladores pueden ser creados con perfiles predefinidos de dispositivos de fabricantes reales, o con características propias personalizadas por el programador. [11]

No precisa de descarga puesto que viene incluido junto con Android Studio.

En muchos casos se ha utilizado directamente un dispositivo móvil para las pruebas ya que es algo más rápido.

Para conectar mi móvil Android y poder probar la aplicación tuve que realizar los siguientes pasos en el móvil**:** opciones de desarrollador, y activar "modo depuración" y permitir "ubicaciones simuladas"

2.3. ButterKnife

Es una librería que nos facilitará la tarea de relacionar los elementos de las vistas con el código en nuestras aplicaciones Android El código queda mucho más limpio. Nos evitará tener que utilizar "*findViewById*" y simplificará el código.

Se tomo la decisión de implantar esta librería porque el código queda mucho más limpio y es extremadamente sencilla de utilizar.

Web de la librería: http://jakewharton.github.io/butterknife/

2.4. CalendarContract

El proyecto de código abierto Android (AOSP) ha tenido una aplicación de calendario desde sus primeros días. Esta aplicación fue diseñada originalmente para sincronizar con Google Calendar, luego se extendió a otras fuentes de sincronización, como Exchange de Microsoft. Sin embargo, esta aplicación no era parte del SDK de Android, así que no había manera de acceder a él desde la aplicación Android.[12]

Se tomó la decisión de utilizar este recurso por la cantidad de ejemplos que hay en la red y su sencillez de uso. Nos permite acceder a los eventos de los calendarios que tengamos sincronizados en nuestro dispositivo Android.

Web de soporte al API

https://developer.android.com/reference/android/provider/CalendarContract.html?hl=es

Alternativas estudiadas.

No se opto por esta solución porque es un API relativamente nuevo y no existe mucha documentación al respecto, además Calendar Contract funciona perfectamente y hace justo lo que necesitamos.

Google Calendar API

Se integra con aplicaciones móviles y web

Puede utilizar la API de Google Calendar para encontrar y ver eventos de la agenda pública. Si está autorizado para ello, también se puede acceder y modificar los calendarios y eventos privados en esos calendarios.

Utilice la API de Google Calendar para lograr una mayor integración con Google Calendar.

Las aplicaciones móviles, aplicaciones Web y otros sistemas pueden crear, mostrar, o sincronizar con los datos del calendario.

La API de Calendario le permite ver, crear y modificar eventos de calendario, así como el trabajo con muchos otros objetos relacionados con el calendario, como calendarios o los controles de acceso. En este documento se describe cómo utilizar las llamadas REST y bibliotecas de cliente para varios lenguajes de programación (Java, PHP, .NET, JavaScript, NodeJs, Ruby,

Python, Go, Android, IOS).[13]

Web de soporte al API:

https://developers.google.com/google-apps/calendar/

2.5. Javamail

Es un API desarrollado por Oracle

El API JavaMail es un paquete opcional (extensión estándar) para leer, componer, y enviar mensajes electrónicos. Usamos este paquete para crear programas del tipo MUA (Mail User Agent), similares a Eudora, Pine, y Microsoft Outlook. Su propósito principal no es transportar, enviar, o re-enviar mensajes como sendmail u otros programas del tipo MTA (Mail Transfer Agent). En otras palabras, los usuarios interactúan con los programas para leer y escribir e-mails. Los programas MUA tratan con los programas MTA para el envío real. El API JavaMail está diseñado para proporcionar acceso independiente del protocolo para enviar y recibir mensajes dividiéndose en dos partes: l La primera parte del API es el foco de este tutor. Básicamente, cómo enviar y recibir mensajes independientemente del proveedor/protocolo. l La segunda parte habla de lenguajes específicos del protocolo como SMTP, POP, IMAP, y NNTP. Con el API, JavaMail para poder comunicar con un servidor, necesitamos un proveedor para un protocolo. [14]

API descargado de esta web: https://code.google.com/archive/p/javamail-android/downloads

2.5.A. Alternativas estudiadas.

No optamos por desarrollarlo con esta API, porque el tiempo era justo y se necesitaba más tiempo para implementar la funcionalidad con este api.

2.5.B. Gmail API

La API de Gmail es una API REST que se puede utilizar para tener acceso a los buzones de correo de Gmail y enviar correo. Para la mayoría de aplicaciones web (incluyendo aplicaciones móviles), la API de Gmail es la mejor opción para el acceso autorizado a los datos de un usuario de Gmail. [15]

La API de Gmail te da acceso flexible, REST bandeja de entrada del usuario, con una interfaz natural de Temas, imágenes, etiquetas,borradores , y la historia. Desde el lenguaje moderno de su elección, su aplicación puede utilizar la API para añadir funciones de Gmail, como:

- Leer los mensajes de Gmail.
- Enviar mensajes de correo electrónico.
- Modificar las etiquetas aplicadas a los mensajes e hilos.
- Buscar mensajes e hilos específicos.

Todo lo que necesita para utilizar la API de Gmail es la biblioteca del cliente de su elección de idioma y una aplicación que puede autenticarse como usuario de Gmail.

La web oficial de este servicio es la siguiente: https://developers.google.com/gmail/api/

2.6. Yahoo! weather API

API que nos permite, mediante unas llamadas, obtener una información precisa del tiempo de la ciudad que deseemos. También nos permite definir si queremos los datos del tiempo en grados Celsius o Fahrenheit. El tratamiento de estos datos se hace mediante objetos JSON. [16]

La web oficial de este servicio es la siguiente: https://developer.yahoo.com/weather/

2.7. Adobe Photoshop

He utilizado este programa para el diseño de los botones y del aspecto visual de la aplicación para que todos los botones y colores sigan un patrón.

Adobe Photoshop es un editor de gráficos rasterizados desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos, su nombre en español significa literalmente "taller de fotos". Es líder mundial del mercado de las aplicaciones de edición de imágenes y domina este sector de tal manera que su nombre es ampliamente empleado como sinónimo para la edición de imágenes en general. [17]

2.8. Adobe Illustrator

Adobe Illustrator (AI) es un editor de gráficos vectoriales en forma de taller de arte que trabaja sobre un tablero de dibujo, conocido como «mesa de trabajo» y está destinado a la creación artística de dibujo y pintura para ilustración (ilustración como rama del arte digital aplicado a la ilustración técnica o el diseño gráfico, entre otros). Es desarrollado y comercializado por Adobe Systems y constituye su primer programa oficial de su tipo en ser lanzado por ésta compañía definiendo en cierta manera el lenguaje gráfico contemporáneo mediante el dibujo vectorial. Adobe Illustrator contiene opciones creativas, un acceso más sencillo a las herramientas y una gran versatilidad para producir rápidamente gráficos flexibles cuyos usos se dan en (maquetación-publicación) impresión, vídeo, publicación en la Web y dispositivos móviles. Las impresionantes ilustraciones que se crean con este programa le han dado una fama de talla mundial a esta aplicación de manejo vectorial entre artistas gráficos digitales de todo el planeta, sin embargo, el hecho de que hubiese sido lanzado en un principio para ejecutarse solo con el sistema operativo Macintosh y que su manejo no resultara muy intuitivo para las personas con muy poco trasfondo en manejo de herramientas tan avanzadas afectó la aceptación de éste programa entre el público general de algunos países. Actualmente forma parte de la familia .[18]

Alternativas para la realización de la interfaz

Elegí el Photoshop y el Illustrator, además del propio gestor de Android Studio porque me pareció suficiente y además el Photosop da más opciones visuales.

Justinmind Prototyper

Aplicación de escritorio que nos permite crear prototipos para aplicaciones móvil y tablet. Dispone de una versión gratuita, también tiene versión de pago, con funcionalidades suficientes para realizar prototipos completos de una forma sencilla.

2.9. TextToSpeech

La síntesis de habla es la producción artificial del habla. El sistema computarizado que es usado con este propósito es llamado computadora de habla o sintetizador de voz y puede ser implementado en productos software o hardware. Un sistema Text-To-Speech (TTS) convierte el lenguaje de texto normal en habla; otros sistemas recrean la representación simbólica lingüística como transcripciones fonéticas en habla. El habla sintetizada puede ser creada a través de la concatenación de fragmentos de habla grabados que son almacenados en una base de datos. La versión 1.6 de Android agregó soporte para los sintetizadores de voz (TTS). [19]

3. Herramientas de planificación

3.1. Git

Git (pronunciado "guit") es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones

cuando éstas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o front end como Cogito o StGIT. Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux. El mantenimiento del software Git está actualmente (2009) supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores.[8]

3.2. Github desktop para Mac

Es una aplicación de escritorio que sirve para gestionar nuestro repositorio GitHub de una forma muy cómoda, sencilla y muy intuitiva. GitHub escritorio está diseñado para simplificar los pasos esenciales en el flujo de trabajo y reemplazar la versión web de GitHub, o a la integración de GitHub con tu IDE. También está disponible para Windows.

La web oficial de la aplicación es: https://desktop.github.com/

Alternativas estudiadas

La mayor diferencia frente a GitHub es que permite de manera gratuita tener repositorios privados, pero escogí GitHub por que tenia la experiencia de haber trabajado anteriormente con él.

BitBucket

Bitbucket es un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de revisiones Mercurial y Git. Bitbucket ofrece planes comerciales y gratuitos. Se ofrece cuentas gratuitas con un número ilimitado de repositorios privados (que puede tener hasta cinco usuarios en el caso de cuentas gratuitas) desde septiembre de 2010, los repositorios privados no se muestran en las páginas de perfil - si un usuario sólo tiene depósitos privados, el sitio web dará el mensaje "Este usuario no tiene repositorios". El servicio está escrito en Python. Es similar a GitHub, que utiliza Git. En una entrada de blog del 2008, Bruce Eckel hace una comparación favorablemente de Bitbucket frente a Launchpad, que utiliza Bazaar.[20]

3.3. Trello

Es una aplicación que sirve para la gestión de tareas dentro de un proyecto, esta aplicación fue desarrollada por Fog Creek Software en 2011.

Tiene un modelo freemium de negocio, que oferta de forma gratuita el servicio para particulares, pero también tiene una versión para negocios que es de pago.[9]

Herramientas para documentación

3.4. Zotero

Zotero es un gestor de referencias bibliográficas, libre, abierto y gratuito desarrollado por el Center for History and New Media de la George Maison University que funciona también como servicio. Como aplicación es posible instalarlo como Extensión de Firefox o como programa independiente (Zotero Standalone), y en ambos casos es multiplataforma, estando disponible para los sistemas operativos Windows, Mac y GNU/Linux. [21]

El funcionamiento de Zotero se basa en los siguientes principios:

- *Recopilar* fuentes bibliográficas, ya sea de forma manual, automática a partir del identificador del recurso o semiautomática a partir de los metadatos de las páginas web. \l
- *Organizar* las referencias a partir de carpetas, etiquetas y carpetas inteligentes. \l
- *Citar* las referencias en documentos, para lo cual ofrece varios formatos (citas, bibliografías o informes), estilos de citación e integración con procesadores de texto. \l

- *Sincronizar* la biblioteca de referencias, notas y adjuntos (para ello es necesaria la creación de una cuenta de usuario) entre distintos equipos o a través de la web oficial. \l
- *Colaborar* con una persona, un grupo de autores o públicamente compartiendo referencias, notas y adjuntos. \l

Web de la herramienta: https://www.zotero.org/

3.5. Open Office

Apache OpenOffice es una suite ofimática libre, de código abierto y de distribución gratuita que incluye herramientas como procesador de textos, hoja de cálculo, presentaciones, herramientas para el dibujo vectorial y base de datos. Soporta numerosos formatos de archivo, incluyendo como predeterminado el formato estándar ISO/IEC OpenDocument (ODF), entre otros formatos comunes, y se enfoca en mantener compatibilidad con el estándar OpenOffice XML, el formato de Microsoft, así como también soporta más de 110 idiomas, desde febrero del año 2010. Apache OpenOffice es uno de los sucesores del proyecto OpenOffice.org e integra características de otras suites ofimáticas como IBM Lotus Symphony. La suite ofimática está disponible para varias plataformas, tales como Microsoft Windows, GNU/Linux, BSD, Solaris y Mac OS X., ademas de diversos ports realizados a otros sistemas operativos. El software es distribuido bajo la licencia Apache.[5] La primera version lanzada fue la 3.4.0, el 8 de Mayo de 2012. Desde esa primera versión, se han realizado diversas bifurcaciones como LibreOffice, StarOffice, o el proyecto Go-OO. Apache OpenOffice desciende de OpenOffice.org, un proyecto que tiene como base inicial a StarOffice, una suite ofimática desarrollada por StarDivision y adquirida por Sun Microsystems en agosto de 1999. El desarrollo de la suite estaba liderado por Sun Microsystems y con posterioridad abandonado por Oracle Corporation. El código fuente de la aplicación está disponible bajo la Licencia pública general limitada de GNU (LGPL) versión 3 hasta la versión 3.4.0 Beta 1.

V - Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto. Se han categorizado en dos apartados: Ciclo de vida y decisiones técnicas relevantes.

1. Ciclo de vida del proyecto

El proyecto comenzó el día 26 de Enero de 2016, día que después de varias reuniones con el tutor llegamos a sentar las bases de la app, establecimos unos objetivos y unos requisitos para empezar desde ahí. Este proyecto nace de una motivación personal, ya que me hubiera gustado disponer de una app así durante mi vida de estudiante. El desarrollo de esta aplicación es propuesto por el alumno, ya que considera que puede ser útil para futuros alumnos de la universidad o incluso para cualquier persona.

Este proyecto se ha desarrollado mediante reuniones periódicas con el tutor, cada semana o cada dos semanas, utilizando así el método Scrum.

Cada reunión costaba de dos partes, la primera, en la que revisábamos los requisitos pedidos y los comparábamos con las funcionalidades obtenidas y la segunda en la que hablábamos sobre cuales podían ser los próximos pasos a dar y establecíamos unos nuevos requisitos para el producto que se debería entregar en la siguiente reunión.

La evolución cronológica del proyecto ha sido la siguiente:

Lo primero, una vez decidimos que podíamos realizar este proyecto personal fue decidir el IDE de desarrollo, que fue Android Studio y familiarizarme con el entorno.

También elegir el programa para llevar el control de versiones, Github fue el elegido ya que ya lo había utilizado anteriormente y contaba con una aplicación de escritorio muy intuitiva. Otra decisión que tomamos fue elegir la herramienta que íbamos a utilizar para gestionar las tareas a realizar en cada iteración, que fue Trello, elegida por su versatilidad y sencillez.

Las tareas que realizamos durante este periodo, en cuanto desarrollo mantuvieron el siguiente orden:

1. Desarrollo de la interfaz de la aplicación, en función a los requisitos que nos habíamos planteado.
2. Diseño de las clases modelo que íbamos a utilizar en nuestra aplicación.
3. Establecer la alarma en una hora seleccionada y poder pararla.
4. Añadir la funcionalidad de que la aplicación pudiera hablar, esto fue posible gracias a TextToSpeech. Para ello reproducimos un mensaje personalizado.
5. Acceder a los archivos mp3 de nuestra tarjeta SD para reproducir una canción cuando suene la alarma.
6. Obtención de los datos del tiempo a través de Yahoo Weather, en función de una ciudad que introduzcamos.
7. Obtención de los datos del Mail, en este caso solo lo hemos hecho para obtenerlos de Gmail, con el API JavaMaill.
8. Obtener las cadenas de caracteres necesarias para que nuestra aplicación lea lo que queremos.
9. Obtención de los datos del calendario interno de nuestro móvil, con el API CalendarContract.
10. Hacer la aplicación persistente, es decir que nuestros datos permanezcan constantes aunque cerremos la aplicación, en este caso lo hemos realizado con SharedPreferences.
11. Cargar los datos al iniciar la aplicación.
12. Mejora de la interfaz con diseño de iconos y botones en Adobe Photoshop y Adobe Illustrator.
13. Generación de la documentación técnica del proyecto.

El proyecto es el resultado final de todo este proceso de este conjunto de iteraciones, conseguidas gracias a las reuniones periódicas y entregas de un nuevo producto cada vez que nos reuníamos.

En el siguiente apartado se explica el porque de algunas decisiones tomadas, en cuanto a los aspectos técnicos del proyecto.

2. Decisiones técnicas

2.1. Versión Android

Una de las primeras dudas que surgió fue la de la versión de Android para la que desarrollar la aplicación. A partir de la siguiente gráfica proporcionada por Android podemos comprobar casi el 90% de los dispositivos actualmente disponen de versiones 4.0 Gingerbread en adelante, por lo que la decisión fue la de hacer una aplicación para esta versión y por lo tanto, compatible con las siguientes versiones.

En los casos que se ha podido se han utilizado los widgets más nuevos en función de la versión, esto se logra poniendo un if, y si la versión del móvil es inferior a lo que se pregunta instanciaremos el último widget que permita esa versión.

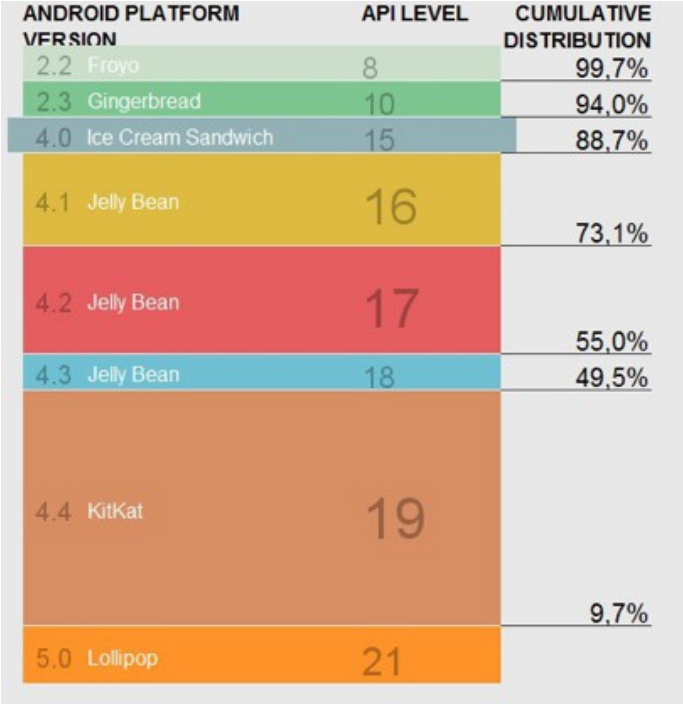

Ilustración 3: Porcentaje de usuarios de las versiones compatibles de Android

2.2. SharedPreferences

Ante el problema de guardar los datos que se introducen en la pantalla de perfil de usuario, como son ciudad, correo, nombre, etc. se sopesaron varias opciones, como la de guardar los datos en un archivo de texto, o el de utilizar la base de datos SQLite. Al final la decisión tomada fue la de utilizar el objeto de la clase *SharedPreferences* que nos aporta Android. [23]

Podemos obtener una referencia a este objeto a través del método *getSharedPreferen*ces().

```
public void loadConfiguration() {
    SharedPreferences sharedPreferences = getSharedPreferences("MyPreferences", Context.MODE_PRIVATE);
    //Cargar los datos del usuario
    user.setName(sharedPreferences.getString("UserName", "User"));
    user.setMailUser(sharedPreferences.getString("UserMailUser", "apd0025tfg"));
    user.setMailPass(sharedPreferences.getString("UserMailPass", "444044ap"));
    user.setCity(sharedPreferences.getString("UserCity", "Burgos"));
    user.setTitle(sharedPreferences.getString("UserTitle", ""));

    //Cargar los datos de alarma
    alarm.setHour(sharedPreferences.getInt("AlarmHour", 10));
    alarm.setMin(sharedPreferences.getInt("AlarmMin", 0));
    alarm.setIsActive(sharedPreferences.getBoolean("AlarmIsActive", false));
    alarm.setSelectCalendar(sharedPreferences.getBoolean("selectCalendar", true));
    alarm.setSelectWeather(sharedPreferences.getBoolean("selectWeather", true));
    alarm.setSelectCustom(sharedPreferences.getBoolean("selectCustom", true));
    alarm.setSelectMail(sharedPreferences.getBoolean("selectMail", true));


    //Cargar el mensaje customizado
    container.setCustomMessage(sharedPreferences.getString("ContainerCustomMessage", "Este es un mensaje person
    container.setSongName(sharedPreferences.getString("ContainerSongName", "Por defecto"));
    //Log nos indica si hemos cargado las preferencias compartidas
    Log.d("LoadPreferences", "SharedPreferences Loaded");
}
```

Ilustración 4: Cargar datos con SharedPreferences

Además para poder editar las preferencias necesitaremos un objeto Editor:

Nos permite tanto leer como escribir datos en este archivo de preferencias. Para ello utilizaremos los métodos *get* y *set* que nos ofrece. En el caso de escribir, precisaremos de realizar un *commit* en el editor para terminar, sino los cambios no quedarán almacenados:

```
public void savePreferences() {
    SharedPreferences sharedPreferences = getSharedPreferences("MyPreferences", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putBoolean("AlarmIsActive",true);
    editor.putBoolean("selectCalendar",swCalendar.isChecked());
    editor.putBoolean("selectWeather",swWeather.isChecked());
    editor.putBoolean("selectCustom",swCustomMessage.isChecked());
    editor.putBoolean("selectMail",swMail.isChecked());

    editor.commit();
}
```

Ilustración 5: Guardar los datos con SharedPreferences

2.3. Carga de datos al iniciar

Se tomó la decisión de que cada vez que se abra la aplicación después de ser cerrada se cargaran los datos del tiempo, Mail y calendario, esto también se hará cuando suene la alarma y pulsemos en la notificación que nos llevará a una pantalla que se pondrá a hablar lo que hayamos configurado que haga. Durante un tiempo tuvimos la funcionalidad de cargar los datos en un botón pero nos pareció innecesario mantenerla.

2.4. ButterKnife

Con el uso de esta librería hemos conseguido que la inyección de los widgets quede mucho más limpia y ordenada, además que ocupa menos líneas de código. Considero que es una práctica indispensable para el desarrollo de una app Android.

En la siguiente imagen podemos observar como era la inyección de widgets al principio.

```
//inicializar los componentes de la interfaz
private TextView tvDaysClick;
private TextView tvSongClick;
private TextView tvCustomMessageClick;
private TextView tvSetAlarmClick;
private TextView tvTimeShow;
private TextView tvCustomMessageAddShow;
private TextView tvSongAddShow;
private TextView tvDaysAddShow;

//Inicializar botones
private Button btAccept;
private Button btCancel;

//Objeto alarm manager que gestiona nuestra alarma
private AlarmManager alarmManager;

//Creamos un PendingIntent que retrasara nuestro intent
//Hasta lo especificado en el calendario
PendingIntent pendingIntent;


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_alarm);
    tvDaysClick = (TextView) findViewById(R.id.tvDaysAddClick);
    tvSongClick = (TextView) findViewById(R.id.tvSongAddClick);
    tvCustomMessageClick = (TextView) findViewById(R.id.tvCustomMessageAddClick);
    tvSetAlarmClick=(TextView)findViewById(R.id.tvTimeAddClick);
    btAccept=(Button)findViewById(R.id.btAcceptAddAlarm);
    btCancel=(Button)findViewById(R.id.btCancelAddAlarm);
    tvTimeShow=(TextView)findViewById(R.id.tvTimeShow);
    tvCustomMessageAddShow =(TextView)findViewById(R.id.tvCustomMessageAddShow);
    tvSongAddShow =(TextView)findViewById(R.id.tvSongAddShow);
    tvDaysAddShow=(TextView)findViewById(R.id.tvDaysAddShow);
```

Ilustración 6: Inyección de widgets sin ButterKnife

La siguiente imagen muestra como queda el código después de utilizar la librería ButterKnife, se observa más limpieza de código.

```
//inicializar los componentes de la interfaz
@Bind(R.id.tvDaysAddClick)
TextView tvDaysClick;
@Bind(R.id.tvSongAddClick)
TextView tvSongClick;
@Bind(R.id.tvCustomMessageAddClick)
TextView tvCustomMessageClick;
@Bind(R.id.tvTimeAddClick)
TextView tvSetAlarmClick;
@Bind(R.id.tvTimeShow)
TextView tvTimeShow;
@Bind(R.id.tvCustomMessageAddShow)
TextView tvCustomMessageAddShow;
@Bind(R.id.tvSongAddShow)
TextView tvSongAddShow;

//Inicializar botones
@Bind(R.id.btAcceptAddAlarm)
Button btAccept;
@Bind(R.id.btCancelAddAlarm)
Button btCancel;
```

Ilustración 7: Instanciación de widgets con ButterKnife

Después de de instanciarlos, de una manera mucho mas limpia que la anterior, lo único que tenemos que hacer para inyectarlos (Crearlos) es lo siguiente y hay dos maneras de hacerlo, una para una Activity normal y otra para un Fragment.

Para una Activity haremos lo siguiente, dentro del método onCreate de nuestra Activity y antes de utilizar ningún widget.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_help_mail);
    ButterKnife.bind(this);
```

Ilustración 8: Inyección de widgets con ButterKnife en una Activity

Para un Fragment haremos lo siguiente, dentro del metodo onCreate de nuestro fragment, pero necesitaremos la View para inyectarlo.

```
//desde aqui llamamos al layout que queremos que muestre
View v = inflater.inflate(R.layout.tab_fragment_4talk, container, false);

//Inyectamos los widgets con butterknife
ButterKnife.bind(this, v);
```

Ilustración 9: Inyección de widgets con ButterKnife en un Fragment

2.5. Calendar contract

Se ha utilizado este API para la obtención de los datos del calendario y así poder leerlos después. Primero se necesita que esté sincronizado en nuestro dispositivo el calendario que deseamos leer.

2.6. Javamail

Se ha utilizado esta librería para la obtención de los datos procedentes del correo electrónico. Sirve para obtener los mensajes provenientes de nuestro correo electrónico, en este caso solo hemos utilizado los mensajes de la bandeja principal. Se obtienen unos objetos Message en los que accedemos al contenido de los correos electrónicos.

2.7. Yahoo! Weather API

Este API ha servido para la obtención de la información meteorológica, es un API con la que trabajan muchos desarrolladores y nos ofrecía justamente lo que necesitábamos, lo único que la condición del estado del tiempo venia en inglés y se tuvo que hacer una traducción mediante un switch case para mostrar la información el idioma correcto.

2.8. Adobe Photoshop y Adobe Illustrator

Con estos programas de diseño gráfico, se han desarrollado las partes visuales de la aplicación, como pueden ser los botones del menú principal, el botón de Stop, las imágenes de carga, el diseño de las imágenes de la notificación y el diseño del logo de la aplicación.

2.9. Vclouds weatherIcons

Son unos iconos que representan el tiempo y que se pueden utilizar en cualquier sitio siempre y cuando no sea para uso comercial. Seleccione estos porque eran los que tenían un aspecto visual más agradable para el usuario.

Han sido descargados de esta página: http://vclouds.deviantart.com/art/VClouds-Weather-

Icons-179152045

2.10. Otras herramientas

2.10.A. JSON

Este tipo de archivos se ha utilizado exclusivamente para obtener la información meteorológica, ya que el API de Yahoo! Weather nos ofrece la posibilidad de obtener así los datos de manera muy sencilla.

JSON, acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función eval(), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web. En la práctica, los argumentos a favor de la facilidad de desarrollo de analizadores o del rendimiento de los mismos son poco relevantes, debido a las cuestiones de seguridad que plantea el uso de eval() y el auge del procesamiento nativo de XML incorporado en los navegadores modernos. Por esa razón, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo, Google, etc., que atienden a millones de usuarios) cuando la fuente de datos es explícitamente de fiar y donde no es importante el no disponer de procesamiento XSLT para manipular los datos en el cliente. Si bien es frecuente ver JSON posicionado contra XML, también es frecuente el uso de JSON y XML en la misma aplicación. Por ejemplo, una aplicación de cliente que integra datos de Google Maps con datos meteorológicos en SOAP hacen necesario soportar ambos formatos. En diciembre de 2005 Yahoo! comenzó a dar soporte opcional de JSON en algunos de sus servicios web. El término JSON está altamente difundido en los medios de programación, sin embargo, es un término mal descrito ya que en realidad es solo una parte de la definición del estándar ECMA-262 en que está basado Javascript. [24]

VI - TRABAJOS RELACIONADOS

Una vez decidimos cuáles iban a ser los requisitos y objetivos de nuestra aplicación, buscamos aplicaciones similares para sacar ideas y encontramos varias, de entre las cuáles destacamos estas dos, una de Android y otra de iOS, porque tenían funcionalidades similares a las de nuestra app y por su interfaz que era sencilla e intuitiva.

1. SmartAlarm de Samsung o Morning!

Es una aplicación Android desarrollada por Samsung, nos ofrece toda la funcionalidad de un gestor de alarmas, unidas a la información del tiempo y a la información en tiempo real de las incidencias de tráfico entre nuestro lugar de residencia y nuestro lugar de trabajo.

2. Alarm Clock

Es una aplicación para iOS. Reloj despertador libre convierte tu iPhone o iPod touch en un hermoso reloj y alarma de un reloj digital de forma gratuita! Incluso muestra viven, las condiciones climáticas locales y de la temperatura que hace que usted sabe sobre el tiempo inmediatamente cuando se despierta.

Sitio de descarga oficial: https://itunes.apple.com/us/app/alarm-clock-free/id332064280?mt=8

VII - Conclusiones y líneas de trabajo futuras

1. Conclusiones

Durante el desarrollo del proyecto he sacado varias conclusiones, que me parecen importantes, desde el significado que tienen personalmente para mi, como del resultado final obtenido.

En cuanto al resultado final, se ha conseguido desarrollar una aplicación completamente funcional, desde cero. Con más tiempo hubiera podido desarrollar más funcionalidades que hubieran dado un extra a la aplicación, pero estoy conforme habiendo conseguido los objetivos planteados al inicio. Respecto a más funciones adicionales que se hayan podido plantear, se explicaran en el apartado de líneas de trabajo futuro para que sean implementadas en siguientes versiones de la aplicación.

Sobre el desarrollo en Android, debo decir que es bastante asequible de aprender, sobretodo teniendo buena base con Java, tiene mucha documentación en la web y da muchas posibilidades al desarrollador. A buen seguro que no será mi ultima aplicación en Android y espero seguir aprendiendo mucho al respecto.

Respecto a la conclusión personal que saco, debo decir que sirve para acercarnos un poco a lo que va a ser nuestra vida laboral, tener reuniones, respetar horarios, entregas, aprender constantemente cosas nuevas y debo decir que me gustaría seguir relacionado con trabajos de este tipo. En lo que a conocimientos se refiere, me ha obligado a estudiar nuevas herramientas y sopesar los pros y contras cuando se dispone de diferentes opciones para realizar una misma tarea, lo cual creo que también es de gran importancia para mi trabajo como desarrollador o Gestor de proyectos.

2. Líneas de trabajo futuras

Con la terminación del proyecto ponemos en marcha una aplicación que puede ayudar desde estudiantes, hasta altos ejecutivos, pero solo es la primera versión funcional. A partir de aquí podemos ir añadiendo infinidad de nuevas funciones, sobretodo respecto a la información que obtenemos para que sea hablada.

1. Añadir información de twitter, en este caso había pensado en añadir las tendencias que son tópicos en un país. Para ello deberíamos familiarizarnos con la API de twitter y con su autenticación para obtener estos datos que también los obtendríamos a través de un archivo JSON. Información disponible en la web oficial: https://dev.twitter.com/rest/reference/get/trends/place
2. Añadir información del tráfico, o lo que es lo mismo tiempo que tardamos en ir al trabajo.
3. Añadir información de las portadas de ciertos periódicos, que pudiéramos elegir.
4. Obtener los datos de la ciudad a través de GPS.
5. Añadir más idiomas, en este caso esta preparado, solo deberíamos añadir los strings necesarios a nuestro .xml, y cambiar algunas de las cadenas y ponerlas en función el .xml también. Text to Speech está diseñado para hablar en multitud de idiomas, uno de ellos

obviamente es el inglés que sería el primero con el que empezaríamos la internacionalización.

6. Respecto al mail, quizás Gmail API es un poco más robusta y nos permite acceder a los mails sin tener que permitir aplicaciones no seguras. Habría que trabajar con el API y con su autenticación.

VIII - REFERENCIAS

1. *Bibliografía*

[1] Android. (2016, 15 de Junio). Wikipedia, la enciclopedia libre. Fecha de consulta: 12:00, Junio 16, 2016 desde https://es.wikipedia.org/w/index.php?title=Android&oldid=91713158

[2] Aplicación móvil. (2016, 15 de junio). Wikipedia, la enciclopedia libre. Fecha de consulta: 14:30, junio 16, 2016 desde https://es.wikipedia.org/w/index.php?title=Aplicaci%C3%B3n_m%C3%B3vil&oldid=91706420

[3] Tipos de aplicaciones móviles: nativas, webs, híbridas. (2016, 12 de Mayo). Blog de Solbyte. Fecha de consulta: 18:30, mayo 16, 2016 desde http://blog.solbyte.com/2014/07/21/tipos-de-aplicaciones-moviles-nativas-webs-hibridas/

[4]Activity (Android). (2016, 15 de Mayo). Desarrolloweb. Fecha de consulta: 20:20, Mayo 15, 2016 desde http://www.desarrolloweb.com/articulos/android-que-es-una-activity-o-actividad.html

[5] Java (lenguaje de programación). (2016, 10 de Junio). *Wikipedia, la enciclopedia libre*. Fecha de consulta: 19:30, Junio 12, 2016 desde https://es.wikipedia.org/w/index.php?title=Java_(lenguaje_de_programaci%C3%B3n)&oldid=91608334

[6] Interfaz gráfica de usuario. (2016, 30 de mayo). Wikipedia, la enciclopedia libre. Fecha de consulta: 09:00, junio 1, 2016 desde https://es.wikipedia.org/w/index.php?title=Interfaz_gr%C3%A1fica_de_usuario&oldid=91377408

[7] Entorno de desarrollo integrado. (2016, 6 de Junio). *Wikipedia, la enciclopedia libre*. Fecha de consulta: 12:30, Junio 10, 2016 desde https://es.wikipedia.org/w/index.php?title=Entorno_de_desarrollo_integrado&oldid=91536202

[8] Git. (2016, 24 de Febrero). Wikipedia, la enciclopedia libre. Fecha de consulta: 17:00, Mayo 22, 2016 desde https://es.wikipedia.org/w/index.php?title=Git&oldid=89371728

[9] Trello. . (2016, 14 de Junio). Wikipedia, la enciclopedia libre. Fecha de consulta: 10:30, Junio 16, 2016 desde https://en.wikipedia.org/w/index.php?title=Trello&oldid=725243831

[10] Android Studio. (2016, 19 de Mayo). *Wikipedia, la enciclopedia libre*. Fecha de consulta: 09:30, Junio 1, 2016 desde https://es.wikipedia.org/w/index.php?title=Android_Studio&oldid=91167304

[11] Android AVD manager. (2016, 1 de junio). Android developers page. Fecha de consulta: 09:00, junio 5, 2016 desde https://developer.android.com/studio/run/managing-avds.html

[12]Calendar contract. (2016, 1 Junio). CommonsWare. Fecha de consulta: 17:00, Junio 1, 2016 desde

https://commonsware.com/Android/previews/the-calendarcontract-provider

[13]Google Calendar API. (2016, 1 Junio). Google developers page. Fecha de consulta 17:30, Junio 1, 2016 desde https://developers.google.com/google-apps/calendar/

[14] Javamail API. (2016, 1 Junio). Cabildodelanzaronte. Fecha de consulta 18:00, Junio 1, 2016 desde http://www.cabildodelanzarote.com/informacion/internet/tutoriales/javamail.pdf

[15] Gmail API (2016, 1 Junio). Google developer page. Fecha de consulta 18:30, Junio 1, 2016 desde https://developers.google.com/gmail/API/

[16] Yahoo Weather API(2016, 1 Junio). Yahoo. Fecha de consulta 18:30, Junio 1, 2016 desde

https://developer.yahoo.com/weather/

[17] Adobe Photoshop. (2016, 22 de Mayo). Wikipedia, la enciclopedia libre. Fecha de consulta: 10:00, Junio 2, 2016 desde https://es.wikipedia.org/w/index.php?title=Adobe_Photoshop&oldid=91380566

[18] Adobe Illustrator. (2016, 13 de Junio). Wikipedia, la enciclopedia libre. Fecha de consulta: 20:00, Junio 2, 2016 desde https://es.wikipedia.org/w/index.php?title=Adobe_Illustrator&oldid=91669414

[19] TextToSpeech (Síntesis del habla). (2016, 1 de Mayo). Wikipedia, la enciclopedia libre. Fecha de consulta: 20:20, Mayo 20, 2016 desde https://es.wikipedia.org/w/index.php?title=S%C3%ADntesis_de_habla&oldid=90805987

[19] Android.SharedPreferences. (2016, 1 Junio). Android developers. Fecha de consulta 13:30, Junio 1, 2016 desde

http://developer.android.com/intl/es/reference/android/content/SharedPreferences.html

[20]BitBucket. (2016, 11 de febrero). Wikipedia, la enciclopedia libre. Fecha de consulta: 20:20, Mayo 22, 2016 desde https://es.wikipedia.org/w/index.php?title=Bitbucket&oldid=89086675

[21] Zotero. (2016, 8 de Abril). Wikipedia, la enciclopedia libre. Fecha de consulta: 10:00, Junio 2, 2016 desde https://es.wikipedia.org/w/index.php?title=Zotero&oldid=90338701

[22]OpenOffice(2016,23 de Junio). Wikipedia, la enciclopedia libre. Fecha de consulta: 16:32, Junio 26, 2016 desde https://es.wikipedia.org/w/index.php?title=Apache_OpenOffice&oldid=91876135

[23] JSON. (2016, 22 de Mayo) Wikipedia, la enciclopedia libre. Fecha de consulta: 10:00, Junio 2, 2016 desde https://es.wikipedia.org/w/index.php?title=JSON&oldid=91218559

Impreso en Burgos el jueves, 7 de julio de 2016