Grado en Ingeniería de Sistemas Audiovisuales

TRABAJO DE FIN DE GRADO

Desarrollo de una App para un servicio de agenda multimodal avanzada en dispositivos móviles Android

Autora: Alba Fernández Lozano

Tutor/Director: Dr. David Griol Barres

Leganés, septiembre de 2015

Resumen

El objetivo principal del presente Trabajo de Fin de Grado es el desarrollo de una aplicación multimodal para dispositivos móviles Android que sirva al usuario como agenda personal avanzada.

A través de ella, se puede almacenar distintos tipos de información relacionada con las actividades diarias, a modo de diario o recordatorio, apoyándose en los servicios y posibilidades que ofrecen los dispositivos móviles Android. Esta información está divida en categorías dependiendo de la naturaleza de cada ítem: una imagen, una nota, un lugar visitado (geolocalización) o un futuro evento. De esta forma se identifica con mayor precisión la información guardada y facilita su posterior búsqueda. El usuario puede acceder a los elementos guardados cuando lo requiera de forma rápida e intuitiva mediante distintos métodos de búsqueda y filtrado de la información.

Además, la aplicación permite el acceso multimodal, es decir, puede ser manejada con el uso de los canales habituales táctiles (pantalla y teclado) o a través de la voz. Asimismo, las respuestas generadas por la aplicación proveen tanto de forma visual como oral, ésta última a través de frases del lenguaje natural de manera que se emula un diálogo entre el usuario y el dispositivo. Esto permite una interacción con la aplicación más natural e intuitiva, lo que facilita el uso de la misma a usuarios con problemas de visión o discapacidades motoras, o en contextos en los que resulta difícil o poco aconsejable el uso de interfaces tradicionales (por ejemplo, durante la conducción). Asimismo, con el propósito de aumentar el número de posibles usuarios, la aplicación ha sido implementada en dos idiomas: inglés y español.

El presente proyecto también incluye un estudio completo de los sistemas de diálogo existentes con el fin de conocer en detalle las características que ofrecen este tipo de tecnologías e incorporar las de mayor interés en el desarrollo de la mencionada aplicación para dispositivos móviles Android.

El desarrollo de la aplicación se apoya en tecnologías adicionales cuyo estudio se incluye también en este proyecto, como son la utilización de una base de datos SQLite, la implementación de servicios basados en localización *(location based service)* y mapas (GoogleMaps) para Android o la integración del manejo de sensores disponibles en los dispositivos móviles Android.

Además, el presente proyecto supone un exhaustivo estudio de la plataforma Android y el entorno de desarrollo de sus aplicaciones, lo cual es de gran interés en un contexto en el que el sector de las aplicaciones móviles se encuentra en pleno auge.

Palabras clave: Sistemas multimodales, App, Dispositivos móviles, Asistentes Personales, Interacción oral, Sistemas de diálogo, Android.

Índice general

Introduction

In this first chapter, the background of this project is presented in order to justify the motivation to develop a multimodal application for Android mobile devices. Subsequently the project objectives as well as the phases of development and the resources used for the implementation are described. Finally, the set of chapters included in this document are summarized.

1. Background

The quick progress of technology and the emergence of smart devices have brought new ways to generate, store and access our information. In addition, the boom of handheld devices such as mobile phones or tablets allows the access to that information wherever the user is located, turning this accessibility an indispensable necessity in our daily life. The information that interests us no longer occupies shelf space so the variety of stored data is increasing.

In this context, the *vital digital storage* concept emerges, which refers to the act of collecting, digitizing and storing in an orderly manner all the information that is generated around the life of a person [1]. This trend, fueled by the social networks, encourages users to digitally record those acts of daily life, with greater or lesser importance, as a diary and usually with the aim of sharing with people around. An image, a phrase we want to remember, a special place we have visited... There is so much information that we accumulate in our digital memory that the organization and search methods to use are very important to allow access to it when required. It is interesting to develop applications systems to manage this information in an automatic, secure and personalized way.

In general, technology is developing in such a way that it's increasingly more present in our daily lives. We are facing a growing number of actions we make through our technological devices, which can be used in many different environments and contexts. If the intention is to continue advancing towards a full integration of the technological world into society, to achieve a more and more natural and intuitive management of our devices is of great importance.

Following this trend, new devices and applications are being developed due to the introduction of multimodal systems. The multimodal interaction uses several parallel input interfaces such as a keyboard, a mouse, a microphone, a camera, a touch screen, etc. Also, these systems can use multiple output channels for providing information to the user such as voice, text, images or graphics so that several user senses are stimulated simultaneously. With this type of human-machine interaction, a more

complete communication is achieved, reducing significantly the typical limitation of systems that use a single communication channel. Furthermore, these systems usually provide to users the possibility to choose the best communication channel depending on the conditions in which they are located.

In particular, the most natural and efficient way of communication used by humans is speech. That is why it is particularly interesting to integrate dialogue systems into computer systems through the development of techniques of speech. Equipping our electronic devices with the ability of understand and generate spoken messages decrease the barriers when interacting with them. This is especially useful for users with impaired vision or motor disabilities, or in environments where it is undesirable or impossible to use traditional interfaces (e.g. while driving).

2. Goals

The main goal of this project is the development of a multimodal application for handheld Android devices which serves as an advanced personal diary.

This application allows to store all kinds of information about their daily activities in an organized way and a quick and easy access to it by diverse research methods. Moreover, the application takes advantage of services and possibilities offered by the Android platform as well as smart phones sensors in order to enhance the way to generate content. Thus, the user can store simple notes as well as pictures, voice records or locations.

Furthermore, an easy and intuitive interaction with the application is intended. This is possible with a multimodal interface which incorporates both touch interface and oral interface that uses natural language sentences for emulating a dialog with the user. This kind of interface that includes diverse access channels provides a more comprehensive user-machine communication which stimulates several sense simultaneously. This is particularly beneficial for environments where it is undesirable or impossible to use traditional interfaces, or for users with impaired vision or motor disabilities.

Intending to achieve the main goal, the following sub-goals have been defined:

- To conduct a comprehensive study of dialog systems in regard to its concept and modular architecture. Also, to analyze some examples of real applications in order to integrate de most interesting features into the Android application developed for this project.
- To conduct a detailed study about the Android platform and its application development environment.

- To study the possibilities offered by the Android platform for developing applications which integrate automatic speech recognition and Text-to-Speeh Synthesis for the communication with the user.
- To analyze necessary technologies for the development of the Android application of this project. This analysis includes the creation and management of a SQLite database, the implementation of location based services and maps based services (*GoogleMaps)* for Android, the integration of sensors enabled on the devices and the development of a client-server architecture for extracting information from web sites.

3. Stages of development

Stage 1: Planning

- **Study of systems dialog:** to define the concept and its modular structure, the requirements to be met by an ideal dialog system and its limitations and to present available real applications of them.
- **Study of the Android platform:** detailed study about the Android platform and its application development environment.
- **Study about systems dialog in Android applications:** to study the possibilities offered by the Android platform for developing applications which integrate automatic speech recognition and Text-to-Speech Synthesis for the communication with the user.
- **To define the functionality of the developed application for Android devices:** description of the functional requirements and basic needs to be met by the application.
- **Study of the necessary technologies for the development of the application:** that includes the study of management systems of SQLite databases, the study of location based services and maps based activities for Android and the development of a client-server architecture for extracting information from web sites.

Stage 2: Development

- **Detailed design:** detailed design and division of the different functionalities of each module and sub-module.
- **Application programming:** application functionality programming by using the Eclipse development environment.
- **Integration and testing:** functional testing for each module separately and for the whole system in order to reach a completely stable version.

- **Evaluation of the application:** analysis of the evaluation systems for oral interfaces, preparation of survey question, design of the online survey and collection and analysis of results.

Stage 3: Documentation

- **Report drafting for the Bachelor's Degree Project.**
- **Preparing the presentation.**

4. Resources employed

Hardware resources:

- Laptop
- Smartphone Sony Ericsson Xperia V
- USB cable

Software resources:

- Integrated development environment Eclipse.
- SDK: Software Development Kit for Android.
- JDK: Java Development Kit.
- Plug-in ADT (*Android Development Tools*) for Eclipse: allows developing Android applications in Eclipse.
- Voice synthesizer PICO TTS.
- Voice search application by Google.
- Google Maps Android v2 API.
- Android support libraries (v4 y v7)*:* Android support libraries that provides compatibility among recent Android APIs and applications running on earlier versions.
- Free source code editor Notepad++.
- *OpenProj:* open source software for project management

5. Structure of the report

This chapter includes a brief summary of each chapter of this document in order to facilitate the reading:

Chapter 1: Introduction: Includes the motivation and the goals of the project, development phases, the resources used for its implementation and the memoir structure.

Chapter 2: State of the art: A study of spoken dialog systems and the possibilities offered by the Android platform to integrate the Automatic Speech Recognition and Text-To-Speech Synthesis is performed. The chapter concludes with a descriptions of examples and relevant applications for the project.

Chapter 3: General description of the system: This chapter begins with an overview of the development system including a description of each module in terms of functionality and architecture. Finally, an analysis of the technologies used in the application development as well as an overview of the implementation of its functions is performed.

Chapter 4: Detailed description of the systems modules: This chapter provides a detailed description of each module of the Android mobile application developed in this project. For each module, the functionality, architecture, data flow and examples of usage scenarios are detailed.

Chapter 5: System evaluation: This chapter describes the methodology used for evaluating the multimodal application developed in this project. Later, the results of the evaluation are discussed to draw conclusions.

Chapter 6: Conclusions and future work: This chapter includes a general assessment of the work done in this project, making an assessment of goals met. Finally, it presents possible future work that could be developed from this project in order to improve the developed application.

Chapter 7: Project management: This chapter presents the timing of the tasks in which the project is divided, as well as a breakdown of the project costs.

Bibliography: It contains a list of references that have been consulted to perform the whole project.

Conclusions and future work

In this chapter, a general assessment of the work done in this project is carried out. First, the final conclusions drawn are discussed making an assessment of the accomplishment of the goals against those initially set. Finally, possible future work that could be developed from this project to improve the developed application is presented.

1. Conclusions

In this final project, a multimodal application for Android mobile devices has been developed, being this the main objective of the project. It is the application called DayByDay which has as main function to serve as an advanced diary for the user. This application allows the user to store all kinds of information about their daily activities in an organized way and a quick and easy access to it by diverse research methods. Moreover, the application takes advantage of services and possibilities offered by the Android platform as well as smart phones sensors in order to enhance the way to generate content. Thus, the user can store simple notes as well as pictures, voice records or locations.

Furthermore, it is intended that the application provides the user with an easy and intuitive interaction. This is possible thanks to a multimodal interface which incorporates both touch interface and oral interface that uses natural language sentences for emulating a dialog with the user. After a review of the functionality and operation of the application developed, one can conclude that the main goal of this project has been reached.

With the aim to achieve the main goal, a set of sub-goals have been initially defined. The achievement of these sub-goals could be assessed as follows:

- **Study of spoken dialog systems.**
  First, a comprehensive study about dialog systems in regard to its concept and modular architecture has been conducted. An analysis of the requirement to be met by an ideal dialog system has also been carried out, as well as examples of applications that use this system. This study has been necessary to add the most relevant characteristics of dialog systems to the application developed, in order to take advantage of all their potential.

- **Study of the Android platform.**
  A full study of the Android platform in terms of architecture, components and functionality has been performed. Also, the Android development environment and its tools and resources has been analyzed in depth. This study is essential

to implement any application for Android devices and has been a great source of knowledge and experience in these systems.

- **Analysis of the possibilities that Android provides for developing applications that allow oral interaction with the user.**
  Before beginning to develop the multimodal application for Android devices, it has been necessary to conduct a comprehensive study of the possibilities that Android provides for integrating Automatic Speech Recognition and Text-To-Speech Synthesis in its applications.

  As it has been seen in this study, from the beginning Android developers, along with Google, have worked to integrate these systems on Android mobile devices in order to make the use of these devices more comfortable. That is why Android provides a multitude of possibilities for developing application that allow oral interaction with the user.

  In regard to voice recognition, it has been concluded that the simplest options is to send an object of the `android.speech.RecognizerIntent` class to the voice search application by Google, so the main features of the `android.speech` package of Android API have been analyzed.

  As for the Text-To-Speech Synthesis, it has been seen that from Android version 1.6 a synthesis engine called Pico TTS is incorporated in most devices, although alternative engines that the user can install and configure in the device have been studied. This synthesis engine allows to integrate the Text-To-Speech Syntehsis trough the `TextToSpeech` class of the `android.speech.tts` package.

  In addition, in order to help to refine the application design, examples of virtual assistants and other relevant applications that integrate the Automatic Speech Recognition and Text-To-Speech Synthesis have been consulted.

- **Study of other necessary technologies for the development of the application.**
  Aside from the conducted study on the Android platform and the methodology for integrating voice recognition systems and text-to-speech synthesis on an Android application, it is necessary to study the integration of other required technologies in the development of this application:

  - The creation and management of a SQLite database for Android, which is necessary to store and administrate the information introduced for the user. SQLite databases, due to their small size, are highly suitable for use in low-memory devices like smartphones or, in general, any Android mobile device. In addition, SQLite provides functionality through a library and not as a separate process so this reduces external

dependencies, simplifies operations and increases portability. Therefore, based on the characteristics studied, it was considered that SQLite reaches the storage requirements of the application developed.

- The study of location providers and location manager which are provided by the Android platform to create location based services, as well as the Google Maps Android API v2 for implementing maps based activities. Specifically, to obtain the location of the device the methods and constants of the `android.location.LocationManager` class, which are available in the Android API, have been used. Also the `android.location.Geocoder` class has been implemented for the translation between locations (expressed in latitude and longitude) and street address. The study of these technologies has been necessary for the ***Places*** and ***Events*** modules.

- The implementation of a client/server based web service for extracting web page content in the ***Notes*** module. Specifically, a RESTful web service has been built due to its high scalability and system performance. For this purpose, the `http-request` and Gson libraries have been used to make the request and to parse the server response. Furthermore, the implementation of this service has involved the execution of background tasks due to the fact that long and complex operations can block the main thread. To prevent this effect, the auxiliary `AyncTask` class has been implemented.

Therefore, the project balance is very positive since all the initial partial objectives have been achieved and thus so has been the main target. Also, the development of this project has enabled the acquisition of extensive experience in developing Android applications as well as new high technology skills.

2. Future work

Coming up next, the futures lines of work which are proposed as improved performance of the developed application are described:

- **Improvements in Automatic Speech Recognition (ASR)**

A potential improvement is the implementation of an “Always listening” mechanism, so that the application may just be waiting for the user to say a keyword to start processing the requests. Thus, a complete management of the application without using the touch system would be achieved, increasing functionality in environments where the use of traditional interfaces is difficult or unwise, or for users with visual

impairments or motor disabilities. While these mechanisms are beginning to be used in some applications and devices, there is no available API for its implementation or documentation that refers to this functionality yet.

On the other hand, as it has been seen, the *Events* and *Notes* items creation does not allow to fill the date and time inputs by voice. Although these fields can be filled later in the editing of the items when the user can use the touch interfaces, to allow the user the full creation of these items by voice would be interesting, implementing a mechanism for processing the received speech to recognize each form field.

Another line of work regarding the Automatic Speech Recognition function is to enable the offline mode, allowing the user to use the voice recognition without Internet access. At the moment, there is not available API to implements this functionality, which can be activated only in a few devices through the settings menu, so this function has not been incorporated in the application developed.

- **Learning from the user's preferences**

An important improvement is the implementation of a mechanism of "learning" by the application based on the information provided by the user. Thus, it is suggested working on the possibility of enabling the application to make recommendations and provide personalized information without requiring the user to make any request. For example, the application would be able to suggest shopping centers from the notes with the *Shopping* category or report on upcoming concerts from the notes with the *Music* category. This function would be achieved through a collection of the user preferences (musical genres, authors of books, frequently visited cities, etc.) storing them in the internal SQLite database and carrying a web service through them which offers relevant information to the user.

- **Languages**

As seen, the developed application is implemented in two languages: Spanish and English. This implies that the vocabulary, the locutions generated by the systems and the methodology for voice recognition operate according the selected language on the device. It would be beneficial for increasing the number of potential users to extend the use of the application into other languages.

Capítulo 1

Introducción

En este primer capítulo se presentan los antecedentes del presente Proyecto de Fin de Grado, con el fin de justificar la motivación de desarrollar una aplicación multimodal para dispositivos móviles Android. Posteriormente, se describen los objetivos del proyecto así como las fases del desarrollo y los recursos empleados para la ejecución del mismo. Por último, se resume el conjunto de capítulos que estructuran el presente documento.

1.1 Antecedentes

El avance tecnológico y la aparición de los dispositivos inteligentes han traído consigo nuevas formas de generar, guardar y acceder a nuestra información. Además, el auge de los dispositivos portátiles como los teléfonos móviles o *tablets* permite el alcance a esa información donde quiera que se encuentra el usuario, volviéndose esta accesibilidad en una necesidad indispensable en nuestra vida diaria. La información que nos interesa ya no tiene porqué ocupar espacio en las estanterías por lo que cada vez es más amplia la variedad de datos almacenados.

En este contexto surge el concepto *almacenamiento vital digital* que hace referencia al acto de recopilar, digitalizar y almacenar de forma ordenada toda la información que se genera alrededor de la vida de una persona [1]. Esta tendencia, estimulada por las redes sociales, lleva a los usuarios a grabar digitalmente aquellos actos de su vida cotidiana, de mayor o menor importancia, a modo de diario y, habitualmente, con el objetivo de compartirlo con su entorno. Una imagen, una frase que queramos recordar, un lugar especial que hemos visitado… es tanta la información que vamos acumulando en nuestra memoria digital que la organización de la misma y los métodos de búsqueda a utilizar son muy importantes para posibilitar el acceso a ella cuando se requiera. Es interesante desarrollar aplicaciones y sistemas que gestionen esta información de una forma automática, segura y personalizada.

En general, la tecnología se está desarrollando de tal manera que cada vez se encuentra más presente en nuestra vida cotidiana. Nos encontramos ante una creciente cantidad de acciones que realizamos a través de nuestros dispositivos tecnológicos y son muchos los entornos y contextos en los que éstos pueden ser utilizados. Si se pretende seguir avanzando en una plena integración del mundo tecnológico en la sociedad, es de gran importancia que el manejo de nuestros

dispositivos se vuelva cada vez más natural e intuitivo y que éstos interactúen con nosotros de la forma menos intrusiva posible.

Siguiendo esta tendencia se están desarrollando los nuevos dispositivos y aplicaciones informáticas gracias a la introducción de los sistemas multimodales. La interacción multimodal utiliza varios interfaces de entrada paralelamente como por ejemplo un teclado, un ratón, un micrófono, una cámara, una pantalla sensible al tacto, etc. Así mismo, este tipo de sistemas puede utilizar varios canales de salida para proporcionar información al usuario como puede ser por voz, texto, imágenes o gráficos de forma que se estimulen varios sentidos del usuario de forma simultánea. Con este tipo de interacción persona-máquina, se consigue una comunicación más completa, disminuyendo considerablemente las limitaciones propias de los sistemas de un único canal de comunicación. Además, estos sistemas suelen ofrecer la posibilidad de que el usuario elija el canal de comunicación que mejor se adapte a las condiciones en las que se encuentra.

En concreto, la forma de comunicación más natural y eficiente que poseemos los seres humanos es el habla. Es por eso que es especialmente interesante integrar los sistemas de diálogo en los sistemas informáticos a través del desarrollo de técnicas del habla. Dotar a nuestros dispositivos electrónicos de la capacidad de comprender y reproducir mensajes orales disminuye considerablemente las barreras a la hora de interactuar con ellos, lo que es especialmente útil para usuarios con problemas de visión o discapacidades motoras, o en entornos en los que no es aconsejable o imposible el uso de interfaces tradicionales (por ejemplo, durante la conducción).

1.2 Objetivos

El objetivo principal del presente Trabajo de Fin de Grado es el desarrollo de una aplicación multimodal para dispositivos móviles Android que sirva como agenda personal avanzada para el usuario.

Dicha aplicación permite al usuario almacenar todo tipo de información referente a sus actividades diarias, a modo de recordatorio o diario, de forma organizada y permitiendo una acceso sencillo y rápido a la misma. Además, aprovecha los servicios y posibilidades que ofrece la plataforma Android así como los sensores disponibles en los teléfonos móviles inteligentes para enriquecer la forma de generar los contenidos. De esta forma, el usuario puede guardar en ella desde sencillas notas escritas hasta imágenes, grabaciones de voz o localizaciones.

Asimismo, se pretende una interactuación con la aplicación los más fácil e intuitiva posible. Esto se consigue a través de una interfaz multimodal que comprende las interfaces táctiles tradicionales y el uso de interfaces orales que, a través de frases del lenguaje natural, emulan un diálogo con el usuario. El uso de este tipo de interfaces que mezcla diferentes canales de acceso proporciona una comunicación usuario-

máquina más completa, estimulando varios sentidos del usuario de forma simultánea. Este hecho es especialmente beneficioso en contextos en los que resulta difícil o poco aconsejable el uso de interfaces tradicionales, o para usuarios con problemas de visión o discapacidades motoras.

Con la intención de alcanzar el objetivo principal, se han definido los siguientes objetivos parciales:

- Llevar a cabo un estudio completo de los sistemas de diálogo multimodal en cuanto a concepto y arquitectura modular, así como un análisis de las aplicaciones y ejemplos existentes en la actualidad con el fin de integrar las características de mayor interés en el desarrollo de la aplicación para dispositivos móviles Android del presente proyecto.

- Realizar un estudio detallado de la plataforma Android y el entorno de desarrollo de sus aplicaciones.

- Realizar un estudio completo sobre las posibilidades que ofrece Android para el desarrollo de aplicaciones que permitan la interacción oral con el usuario, es decir, que integren el reconocimiento automático del habla para la comunicación de entrada y la síntesis de texto a voz para la comunicación de salida.

- Realizar un estudio sobre las diferentes tecnologías en las que se apoya la aplicación desarrollada en el presente trabajo que incluya: la creación y gestión de una base de datos SQLite, la implementación de servicios basados en localización (*location based service*) y mapas (GoogleMaps) para Android, la integración del manejo de sensores disponibles en los dispositivos móviles Android y el desarrollo de una arquitectura cliente-servidor para la extracción de información de páginas web.

1.3 Fases de desarrollo

Fase 1: Planificación

- **Estudio de los sistemas de diálogo:** definición del concepto y su estructura modular, estudio de los requisitos que debe cumplir un sistema de diálogo ideal y las limitaciones a los que están sujetos y presentación de aplicaciones reales existentes hasta la actualidad.

- **Estudio de la plataforma Android:** estudio exhaustivo de la plataforma Android y del entorno de desarrollo de sus aplicaciones.

- **Estudio sobre los sistemas de diálogo en aplicaciones Android:** estudio de las posibilidades que ofrece la plataforma Android para el desarrollo de aplicaciones que integren el reconocimiento automático del habla y la síntesis de texto a voz como interfaz con el usuario.

- **Definición de la funcionalidad de la aplicación para dispositivos móviles Android:** descripción de los requisitos funcionales y necesidades básicas que debe cumplir la aplicación.

- **Estudio de las tecnologías necesarias para el desarrollo de la aplicación:** incluye el estudio de los sistemas de gestión de base de datos SQLite y su lenguaje de acceso, el estudio de la implementación de servicios basados en localización (*location based services)* y mapas (GoogleMaps) para Android y el desarrollo de una estructura cliente-servidor para la extracción de información de páginas web.

Fase 2: Desarrollo

- **Diseño detallado:** diseño detallado y división de las distintas funcionalidades de cada módulo y sub-módulo del sistema.
- **Programación de la aplicación:** programación de la funcionalidad de la aplicación utilizando el entorno de desarrollo Eclipse.
- **Integración y pruebas:** realización de pruebas funcionales para cada módulo por separado y del sistema completo hasta alcanzar una versión completamente estable.
- **Evaluación de la aplicación:** estudio de los sistemas de evaluación para interfaces orales, elaboración de las preguntas de la encuesta, diseño de la encuesta online y recogida y análisis de los resultados.

Fase 3: Documentación

- **Redacción de la memoria del Trabajo de Fin de Grado**
- **Preparación de la presentación**

1.4 Recursos empleados

Los recursos empleados en el presente Trabajo de Fin de Grado son los siguientes:

Recursos Hardware:

- Ordenador portátil.
- Smartphone Sony Ericsson Xperia V.
- Cable USB.

Recursos Software:

- Entorno de desarrollo de código abierto multiplataforma Eclipse.
- SDK *(Software Development Kit)* de Android: kit de desarrollo de software para Android.
- JDK *(Java Debelopment Kit)*: kit de desarrollo de Java.
- Plug-in ADT (Android Development Tools) para Eclipse: permite desarrollar aplicaciones Android en Eclipse.
- Sintetizador de voz PICO TTS.
- Aplicación de búsqueda por voz de Google.
- API versión 2 de Google Maps para Android.
- Librerías *Android-support* (v4 y v7)*:* librerías de soporte de Android que proporciona compatibilidad entres APIs de Android recientes y aplicaciones ejecutadas en versiones anteriores.
- Editor de programación Notepad++.
- Herramienta *OpenProj* de administración de proyectos.

1.5 Estructura de la memoria

A continuación se incluye un breve resumen del contenido de cada capítulo del presente documento con el fin de facilitar la lectura del mismo:

Capítulo 1: Introducción: Incluye la motivación y objetivos del proyecto, las fases de desarrollo, los recursos empleados para su implementación y la estructura de la memoria.

Capítulo 2: Estado del arte: Se realiza un estudio de los sistemas de diálogo hablado y de las posibilidades que ofrece la plataforma Android para integrar el reconocimiento automático del habla y la síntesis de texto a voz en aplicaciones. El capítulo concluye con una descripción de ejemplos y aplicaciones reales de interés para el presente trabajo.

Capítulo 3: Descripción general del sistema: Este capítulo comienza con una presentación general del sistema desarrollado. Con este objetivo, se realiza una descripción de cada uno de los módulos que la componen en cuanto a funcionalidad, arquitectura y tecnologías utilizadas. Finalmente, se realiza un análisis de las tecnologías empleadas en el desarrollo de la aplicación así como una descripción general de la implementación de sus funcionalidades.

Capítulo 4: Descripción detallada de los módulos del sistema: En este capítulo se realiza una descripción detallada de cada uno de los módulos que componen la aplicación para dispositivos móviles Android desarrollada en el presente trabajo. Para cada módulo se detalla la funcionalidad, arquitectura, flujo de datos y ejemplos de posibles escenarios usos.

Capítulo 5: Evaluación del sistema: Se describe la metodología utilizada en la evaluación de la aplicación multimodal para dispositivos móviles desarrollada en el presente Trabajo de Fin de Grado. Posteriormente, se analizan los resultados obtenidos en la evaluación para obtener conclusiones.

Capítulo 6: Conclusiones y trabajo futuro: Incluye un balance general del trabajo realizado en el presente Trabajo de Fin de Grado. Se exponen las conclusiones finales extraídas realizando una valoración de los objetivos cumplidos frente a los marcados inicialmente. Finalmente, se presentan los posibles trabajos futuros que se podrían desarrollar a partir de este proyecto con el fin de mejorar la aplicación desarrollada.

Capítulo 7: Gestión del proyecto: En este capítulo se presenta la planificación temporal de las tareas en las que se divide el proyecto, así como un desglose de los costes del proyecto.

Bibliografía: Contiene una lista de la referencias bibliográficas que se han consultado para la realización completa del presente trabajo.

Capítulo 2

Estado del arte

En este capítulo se describe el contexto en el que se enmarca el presente Trabajo de Fin de Grado. En primer lugar se realiza una introducción a los sistemas de diálogo hablado, su arquitectura modular y aplicaciones. A continuación se estudia las posibilidades que ofrece la plataforma Android para integrar estos sistemas en el desarrollo de aplicaciones. Para ello, se comienza con un estudio sobre el reconocimiento automático del habla en aplicaciones Android y, posteriormente, de la síntesis de texto a voz. Se concluye este capítulo con una descripción de ejemplos y aplicaciones reales de interés para el presente proyecto.

2.1 Los sistemas de diálogo hablado

2.1.1 Introducción a los sistemas de diálogo hablado

Los sistemas de diálogo hablado (*spoken dialogue systems)* son sistemas informáticos que reciben como entrada frases del lenguaje natural expresadas de forma oral y generan como salida frases del lenguaje natural expresadas de forma oral [2].

El objetivo de estos sistemas es emular el comportamiento inteligente de un ser humano durante el diálogo con otra persona, con el objetivo de que el sistema realice alguna tarea, resultado de esta interacción con el usuario.

A través de la conversación, la comunicación persona-máquina se vuelve más natural e intuitiva. Además, disponiendo de un interfaz hablado, se libera al usuario de utilizar otros canales tradicionales como el teclado, el ratón o la pantalla. Este tipo de interacción con el dispositivo facilita la accesibilidad a sus aplicaciones, especialmente en el caso de personas con discapacidades o entornos en los que no es aconsejable o resulta imposible el uso de interfaces táctiles o visuales (por ejemplo, durante la conducción) [3]. Por este motivo, es de gran interés que se introduzcan sistemas de diálogo hablado en aplicaciones reales a través del desarrollo de las tecnologías del habla, ámbito interdisciplinar que proporciona los conceptos, técnicas y herramientas necesarias para que un sistema informático pueda reconocer e interpretar el habla humana, así como reproducirla con un alto nivel de naturalidad.

En este ámbito, lo que se persigue especialmente es el desarrollo de sistemas que combinen la interacción oral con otro tipo de interfaces (por ejemplo, interfaces

tradicionales táctiles) de forma que se estimulen varios sentidos del usuario simultáneamente. En esto se fundamentan los sistemas multimodales, los cuales implican una comunicación usuario-máquina más completa al mezclar diferentes canales de acceso.

A pesar del gran avance en el estudio de la materia, construir sistemas de diálogos ideales supone aún un reto. Los sistemas de diálogos de los que disponemos actualmente están sujetos a ciertas limitaciones relacionadas con el reconocimiento del habla. No disponemos de reconocedores que puedan resolver con total fiabilidad la variación propia de la naturaleza espontánea del habla, la multiplicidad de realizaciones fonéticas causa de las diferentes procedencias geográficas y sociales de los locutores y las interferencias producidas por el entorno en el momento en que se está utilizando el sistema.

No obstante, los constantes avances de la investigación en Tecnologías del Habla permiten que sean factibles las interfaces orales cada vez más flexibles y completas, siendo integradas en un creciente número de servicios y aplicaciones.

2.1.2 Metodología y arquitectura de los sistemas de diálogo hablado

Este tipo de tecnologías utilizan usualmente varios turnos de diálogo de forma que se alcance gradualmente el objetivo del usuario. Este hecho precisa que el sistema requiera de cierta complejidad para procesar la información recibida anteriormente, tomar decisiones para reconducir el diálogo en el marco definido, solicitar aclaraciones cuando la información aportada por el usuario no sea clara, etc. [3]

La Figura 2.1 [3] resume las acciones básicas que realiza un sistema de diálogo. Tal y como se observa en la imagen, el sistema genera un mensaje inicial que sirve habitualmente para dar la bienvenida o informar al usuario sobre las características del sistema. Tras cada intervención del usuario, el sistema realiza un conjunto de acciones básicas que se repiten cíclicamente:

- Reconocer la secuencia de palabras mencionadas por el usuario.
- Comprender el significado de dichas palabras, es decir, extraer la información útil en el dominio del sistema.
- Realizar operaciones de acceso a bases de datos u otros recursos del sistema, en los que se almacena la información u operaciones solicitadas por el usuario.
- Decidir la respuesta que debe suministrar el sistema, es decir, qué acción o acciones deben realizarse a continuación de cada solicitud del usuario.
- Reproducir un mensaje hablado que indique al usuario qué acción ha seleccionado el sistema.

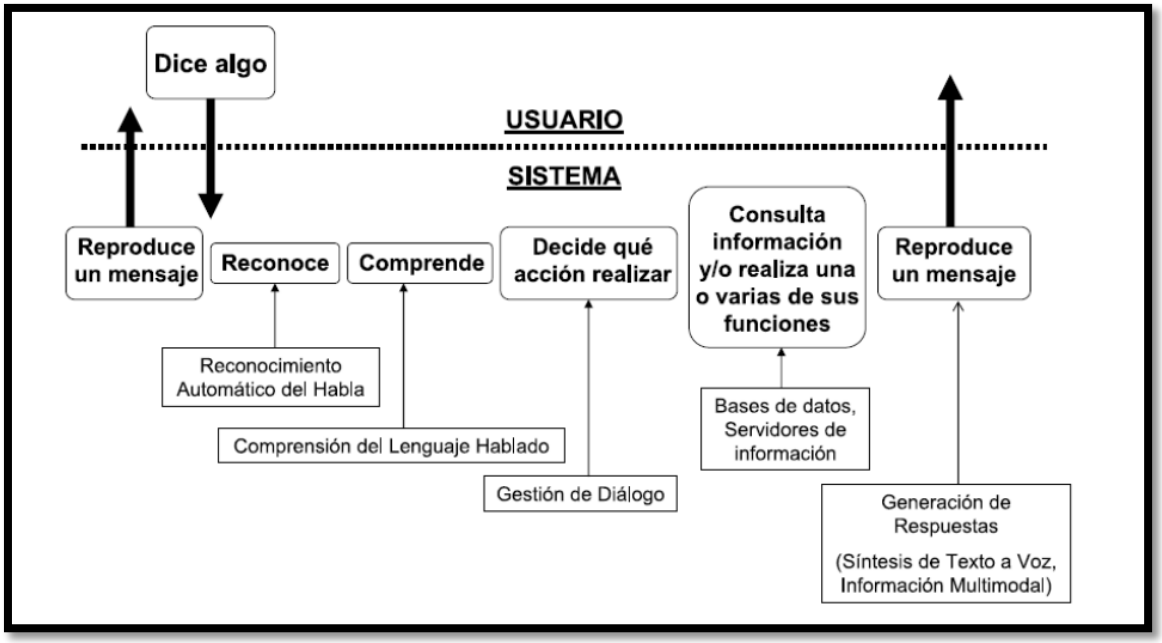


Figura 2.1: Diagrama de acciones de un sistema de diálogo

Debido el gran número de operaciones que conforma este proceso, la arquitectura de todo sistema de diálogo hablado se basa en una estructura modular que permite dividir el desarrollo en diferentes bloques y disminuir la complejidad del conjunto [3].

El esquema que observamos en la Figura 2.2 [3] muestra la arquitectura modular típica de un sistema de diálogo hablado. A continuación, se describen cada una de las unidades que la componen.

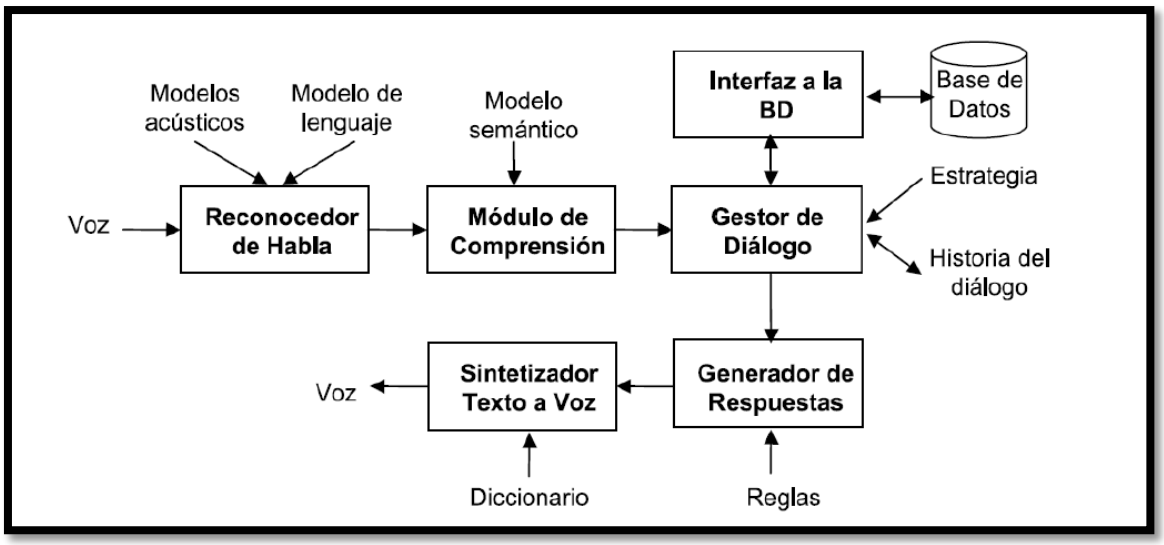


Figura 2.2: Arquitectura modular de un sistema de diálogo hablado

- **Módulo de reconocimiento automático del habla (RAH).**
  Este módulo se encarga de recibir la señal sonora emitida por el usuario y traducirla en una representación escrita que puedan utilizar los demás módulos del sistema. Para ello, la primera tarea que realiza es analizar la variación en el tiempo de la amplitud y frecuencia de la onda recibida y separarla en segmentos mínimos del habla. A partir de ellos, proporciona una o varias secuencias de palabras reconocidas con mayor probabilidad.

- **Módulo de Comprensión del Habla o Módulo de Procesamiento del Lenguaje Natural (PLN).**
  A partir de la(s) secuencia(s) proporcionadas por el módulo RAH, el sistema obtiene una representación semántica abstracta de dicha información que servirá para comprender la información que requiere el usuario.

- **Gestor de diálogo**.
  Se considera el elemento principal del sistema con respecto a la toma de decisiones, ya que es el que controla y coordina los procesos de los diferentes módulos. El gestor de diálogo se encarga de decidir qué paso debe dar el sistema tras cada intervención del usuario. Para ello, se apoya en la interpretación semántica de la petición del usuario, la historia del proceso de diálogo, la información de la aplicación disponible en ese punto y el estado del sistema.

- **Módulo de Consulta a la Base de Datos de la Aplicación.**
  Este módulo recibe peticiones de consulta a la base de datos por parte del gestor de diálogo, las procesa, y devuelve el resultado al gestor.

- **Módulo de Generación de Respuestas.**
  Se encarga de recibir la respuesta del sistema en forma de cierta representación formal y genera un frase, gramaticalmente correcta y en un lenguaje los más cercano posible al lenguaje natural, que transmita el mensaje generado por el gestor de diálogo. Esta respuesta puede incorporar otras modalidades de información como vídeos, tablas con datos, gesto a reproducir por un avatar, etc.

- **Sintetizador de texto a voz (TTS, Text-to-Speech).**
  Es el último paso de un sistema de diálogo ya que se encarga de hacer llegar al usuario el resultado generado. Esta tecnología transforma textos escritos en su equivalente sonora, es decir, efectúa la lectura en voz alta de un texto escrito. El sintetizador de texto a voz se compone de dos partes:
    - *Front-end:* toma como entrada un texto y lo convierte en su representación lingüística fonética, es decir, asigna una transcripción fonética a cada palabra y divide el contenido del mismo en frases y

oraciones. Realiza, para ello, un análisis espectral de la señal de voz (transformada de Fourier).

- *Back-end:* convierte la representación fonética anterior en sonido, simulando el emitido por las cuerdas vocales.

2.1.3 Aplicaciones de los sistemas de diálogo

La cantidad de entornos en los que se pueden aplicar los sistemas de diálogo es muy amplio. En la Tabla 2.1, se listan algunos ejemplos de aplicaciones reales de sistemas de diálogo (y sus desarrolladores) existentes en la actualidad, clasificadas según su funcionalidad y ordenados de menor a mayor complejidad [4] [5].

Funcionalidad	Tarea desarrollada	Ejemplos
Gestión de información	Son el caso de los servicios de atención al cliente que le orientan en la solución de un problema o de las centralitas automáticas que dirigen al usuario de la persona adecuada. También pueden gestionar datos propios del usuario, como en los sistemas de consulta de correo electrónico a través del teléfono.	*Verbio technologies* [36]: Servicios de atención al cliente. España *AthosMail* [37]: Sistema multilingüe y adaptativo para la lectura de correos electrónicos utilizando teléfonos móviles. EE.UU.
Consulta de información	Facilita cierta información requerida por el usuario: horarios y precios de transportes, información meteorológica, consulta de notas, oferta cultural y de ocio, etc. Por lo general, dicha información se encuentra en una base de datos a la que accede el sistema de diálogo.	*Jùpiter* [38]: Sistemas de información meteorológica por teléfono. EE.UU. *Dihana* [5]: Información de viajes en tren. España. *Pegasus* [39]: Información de viajes en avión. EE.UU
Reservas	Además de proporcionar información, realizan tareas algo más complejas como son reservas a través del sistema.	*Mercury* [40]: Proporciona información sobre vuelos y permite hacer reservas. EE.UU
Transacciones	Relacionado con el comercio electrónico (por	*Cinentradas* [41]: Venta telefónica de entradas de

	ejemplo, venta de entradas o de billetes de transportes) y la banca electrónica.	cine. España *Línea BBVA* [43]: permite a los clientes de BBVA realizar operaciones y consultas relacionadas con sus cuentas bancarias. España.
Control remoto	Control remoto de dispositivos inteligentes.	*MIMUS* [42]: Sistema de diálogo multimodal para controlar un hogar inteligente. España
Tutorización	Empleados en el ámbito de la educación, particularmente con el fin de mejorar habilidades fonéticas y lingüísticas de los usuarios.	*LISTEN* [44]: tutor de lectura. España. *VOCALIZA* [45]: Sistema para terapias de voz. España. *ITSPOKE* [46]: Tutor para dar información y corregir errores de aprendizaje. EE.UU.
Robótica	Incorporación de los sistemas de diálogo en la robótica.	*Aisoy 1* [47]: Robot-mascota calificado como uno de los primeros robots emocionales para el mercado de consumo. España.
Asistentes virtuales	Agentes conversacionales que ayudan a los usuarios en las tareas que realicen a través de Internet. Se trata de la aplicación más compleja de los sistemas de diálogo.	*Pregunta a Irene* [48]*:* asistente virtual de la compañía RENFE. España

Tabla 2.1: Ejemplos de aplicaciones de sistemas de diálogo

2.2 Reconocimiento Automático del Habla en Android

2.2.1 Introducción al reconocimiento automático del habla en Android

El reconocimiento automático del habla o ASR (*Automatic Speech Recognition*) es una herramienta computacional capaz de transcribir el contenido de una señal de voz. Es

decir, recibe como entrada una onda sonora y obtiene a la salida su representación escrita. Se trata del primer módulo en un sistema de diálogo.

Desde los inicios, los desarrolladores de Android junto con Google han trabajado por integrar estos sistemas en los dispositivos móviles Android y aprovechar todo su potencial con el objetivo de buscar la comodidad en el uso de estos dispositivos.

El primer contacto de la plataforma Android con este tipo de sistemas se produjo con la aplicación de búsqueda por voz Google Search, instalada por defecto en la mayoría de los dispositivos [6]. Esta aplicación consiste en una pantalla inicial con el texto "Habla ahora" que indica al usuario cuándo el dispositivo se encuentra "escuchando". El audio recogido es transferido a los servidores de Google para que se haga el reconocimiento por lo que es necesario tener acceso a Internet en el momento en que se esté utilizando la aplicación.
En las siguientes versiones, Android ha ido incorporando nuevas características y funcionalidades al reconocimiento de voz [7] [8]:

- **Android 2.1** incorpora la entrada de voz por teclado que permite dictar aquello que se quiera escribir en cualquier campo de texto que utilice el teclado del dispositivo como entrada (Ej. SMS, WhatsApp, e-mail, etc).

- A partir de la versión **Android 2.2,** Google lanza las acciones de voz para Android (*Voice Actions for Android)*, un sistema que permite al usuario dar órdenes comunes al dispositivo a través de la voz y que éste las ejecute.

- Con la versión **Android 4.0** se introduce un nuevo sistema de reconocimiento de voz que corrige y subraya el texto dictado en tiempo real, sin esperar que el usuario haya terminado.

- Además de importantes mejoras en la búsqueda por voz, la versión **Android 4.1 Jelly Bean** permite activar por primera vez el dictado de voz *offline*, lo que permite el uso del reconocimiento de voz sin necesidad de conexión a Internet y acelera el proceso. No obstante, no existe ninguna API disponible para implementar esta funcionalidad, únicamente es posible activarla desde el menú de opciones del dispositivo la cual no está disponible en todos los modelos.

- La última novedad importante viene con la versión **Android 4.3 KitKat**, que permite la activación de la búsqueda por voz en cualquier momento pronunciando el comando "OK Google", sin necesidad de pulsar el icono del micrófono, ya que el dispositivo se encuentra constantemente a la escucha.

2.2.1.1 Entrada de voz por teclado

La entrada de voz por teclado [9] se activa en todas aquellas aplicaciones que permita la entrada de texto con el teclado en pantalla. Se realiza a través de un botón con el icono de un micrófono insertado en el teclado táctil del dispositivo y que al ser pulsado muestra la imagen de un micrófono indicando que el dispositivo está "escuchando". En el momento en que el usuario hace una pausa, el servicio de reconocimiento de voz transcribe el texto que se ha enunciado. Hay que tener en cuenta que el reconocimiento de voz se hace desde los servidores de Google por los que es necesario tener acceso a Internet. Además, la aplicación de búsqueda por voz debe estar instalada en el dispositivo.

La Figura 2.3 ilustra el proceso de introducción de texto por voz en un campo de texto. En primer lugar, el usuario debe pulsar el campo de texto para activar el teclado que es donde está alojado el botón que activa la escucha. Pulsando este botón, aparece una pantalla que indica al usuario que el dispositivo está preparado para recoger el mensaje. En el momento en que se produzca una pausa, el sistema transcribe el mensaje de voz en el campo de texto. En el caso de dispositivos Android 4.0 o superiores, los resultados del reconocimiento de voz se van escribiendo a medida que el usuario hable, sin necesidad de que éste haya terminado.

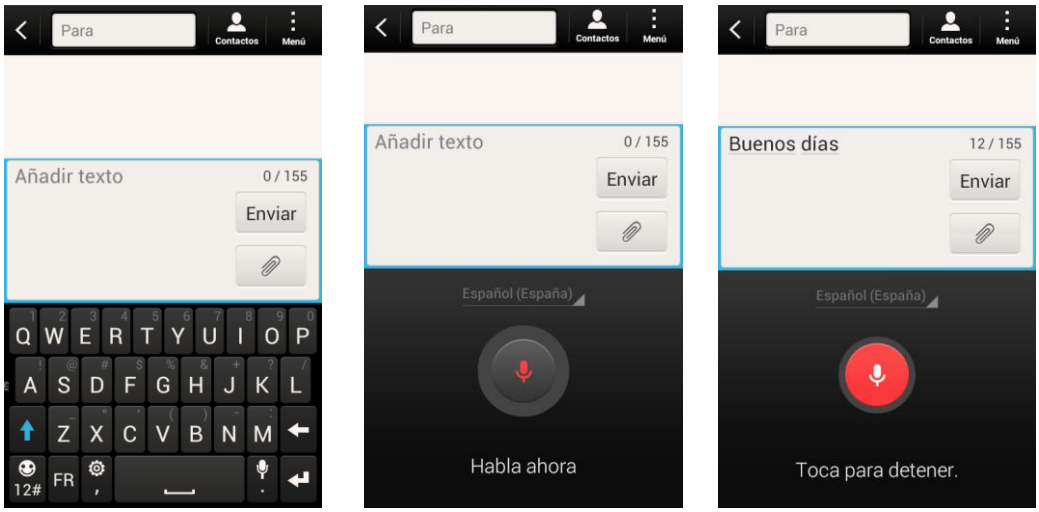


Figura 2.3: Introducción de un texto por voz

Cabe destacar que el menú de configuración del dictado por voz de Google permite la selección del idioma de entrada así como el bloqueo de palabras ofensivas, que censura el resultado cuando éste se trata de un lenguaje ofensivo, tal y como se observa en la Figura 2.4.

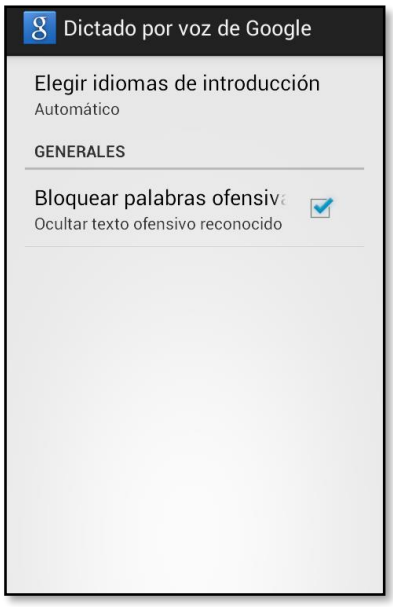


Figura 2.4: Ajustes de dictado por voz de Google

2.2.1.2 Evolución de las acciones de voz de Google para Android

A partir de la versión **Android 2.2,** Google lanza las acciones de voz para Android (*Voice Actions for Android)*, un sistema que permite al usuario dar órdenes comunes al dispositivo a través de la voz y que éste las ejecute [10]. Inicialmente, se encontraba disponible únicamente en inglés pero más tarde se lanzó una actualización para que la aplicación se pudiese utilizar en otros idiomas.

Las acciones de voz se activan a través de comandos de voz establecidos previamente y que son entendidos por el sistema. La Tabla 2.2 muestra las acciones que incorporó Android 2.2 para el idioma español.

Acción	Comando de voz	Ejemplo
Enviar mensajes	Enviar mensaje a [contacto] [mensaje]	Enviar mensaje a Marcos. Llegaré a casa a las 9.
Realizar llamadas	Llamar a [nombre] (y opcionalmente a qué teléfono: móvil, casa, trabajo)	Llamar a Carolina en casa
Llamar a empresas	Llamar a [nombre de la empresa] (y opcionalmente la ubicación)	Llamar a Sara Peluqueros en Madrid
Mostrar mapa en *Google Maps*	Mapa de [dirección/ciudad]	Mapa de París
Cómo llegar en *Google Maps*	Cómo llegar a [dirección/ciudad/nombre	Cómo llegar a calle San Bernardo 55

	de la empresa]	
Visitar una página Web	Ir a la página web de [sitio web]	Ir a la página web de Wikipedia
Iniciar ruta con *Google Navigation*	Guíame a [destino]	Guíame a Plaza Mayor
Búsqueda en Google	[Consulta]	Imágenes de la Plaza Roja de Moscú

Tabla 2.2: acciones de voz para Android 2.2 para el idioma español

A parte de las acciones descritas, y para el idioma inglés, la versión 2.2 permitía las siguientes acciones:

- Escribir una nota: *note to self [note]*
- Escuchar música: *Listen to [artist/song/álbum].* Es necesario tener instalada alguna aplicación de música. (Ej: Pandora, Last.fm, Rdio, mSpot)
- Enviar un email: *Send email to [contact] [message]*

Posteriormente, las acciones de voz han ido evolucionando rápidamente con cada nueva versión de Android. La mejora más notoria se produjo recientemente con la llegada de **Android 4.1 Jelly Bean,** versión que permitía por primera vez el modo *offline* de la aplicación de reconocimiento de voz, es decir, posibilitaba el uso de la entrada de voz por teclado en cualquier momento, sin necesidad de encontrarse conectado a Internet. Además, se producen otras importantes mejoras en el reconocimiento automático del habla, todas ellas impulsada por dos grandes proyectos perseguidos por la plataforma Android:

- *Project Majel* [11] [12]: El proyecto Majel, conocido actualmente por Google Now, persigue el objetivo de transformar la búsqueda por voz en un asistente personal inteligente, similar a Siri de Apple. El asistente es manejado a través de una interfaz de usuario multimodal capaz de interpretar un lenguaje natural y cotidiano y frases complejas, de forma que el usuario no tiene que memorizar complejos comandos específicos.

- *Google's Knowledge Graph Project* [13]*:* ambicioso proyecto de Google que tiene como objetivo la búsqueda web semántica, es decir, entender la relación de personas, lugares y cosas de la vida real, combinado con bases de datos de páginas como Wikipedia, para ofrecer un resultado más próximo a la mejor respuesta humana.

Es especialmente interesante mencionar el trabajo realizado por los desarrolladores de Google en las aplicaciones de reconocimiento de voz. El asistente personal de Google está considerado como uno de los favoritos por los usuarios. Actualmente, es uno de los que cuentan con mayor número de descargas y se encuentra en funcionamiento en prácticamente todas las plataformas, tanto en Android como en Windows, integrado en Google Chrome, y otros sistemas operativos como iOS.

2.2.2 Integración del reconocimiento de voz en una aplicación para dispositivos móviles Android

Para realizar el reconocimiento del habla en un dispositivo Android, se debe lanzar una aplicación preparada para realizar esta acción. Esto se efectúa a partir del uso de intenciones (*intents*) que permiten lanzar una actividad o servicio de alguna aplicación existente en el dispositivo [14].

Por lo tanto, es necesario que el dispositivo cuente con una aplicación que pueda procesar la Intención para el reconocimiento del habla. Una de esas aplicaciones, y la que será utilizada en la aplicación para dispositivos móviles Android del presente proyecto, es *Google Voice Search*, uno de los mejores reconocedores de voz para Android, desarrollado por Google, que es compatible en varios idiomas y que se encuentra instalada por defecto en la mayoría de los dispositivos.

Así mismo, existen dos alternativas más desarrolladas por otras compañías para la integración del reconocimiento de voz en una aplicación Android [15]:

- *ISpeech* [18]: SDK de reconocimiento de voz y síntesis de texto a voz para Android desarrollado por la empresa iSpeech. Permite integrar el reconocimiento de voz y la síntesis de texto a voz a cualquier aplicación Android mediante la nube de iSpeech. Dispone de 27 idiomas para el reconocimiento de voz y la síntesis de texto a voz y 15 idiomas para el dictado en la entrada por voz.
- *CmuSphinx* [50]: Kit de herramientas de código abierto desarrollado por la Carnegie Mellon University para la integración de reconocimiento de voz en modo *offline* a una aplicación, sin necesidad de disponer de acceso a Internet.

Aunque existen varias alternativas para la integración del reconocimiento de voz en una aplicación Android, la opción más simple consiste en enviar un objeto de la clase `android.speech.RecognizerIntent` a la aplicación de búsqueda por voz de Google, nombrada anteriormente.

En la presente sección se resume las principales características del paquete `android.speech` del API de Android y del uso de la clase `android.speech.RecognizerIntent` para integrar el reconocimiento de voz en una aplicación para dispositivos móviles Android.

2.2.2.1 Descripción del paquete android.speech

Las clases disponibles en el paquete `android.speech` permiten integrar en una aplicación Android servicios de reconocimiento de voz existentes, así como la creación de servicios de reconocimiento de voz completamente nuevos.

La Tabla 2.3 [16] muestra un resumen de las clases e interfaces que comprender el paquete android.speech. Para una explicación más detallada de las funcionalidades de dicho paquete se puede consultar el API de Android.

Interfaces	Definición
RecognitionListener	Utilizado para recibir notificaciones desde el reconocedor de voz (SpeechRecognition) cuando se capta algún evento relacionado con dicho reconocedor (Ej: el usuario ha dejado de hablar)

Clases	Definición
RecognitionService	Clase que sirve de base para implementar un servicio de reconocimiento de voz. Hay que tener en cuenta que esta clase sólo debe ser extendida en el caso de que se desee implementar un nuevo servicio de reconocimiento de voz.
RecognitionService.Callback	Clase que recibe las notificaciones del servicio de reconocimiento de voz y se las envía al usuario. Se deberá pasar una instancia de esta clase al método onStartListening(Intent, Callback) de la clase RecognitionService.
RecognizerIntent	Clase que define las constantes necesarias para integrar el reconocimiento de voz iniciado desde un *intent.*
RecognizerResultsIntent	Clase que define las constantes asociadas a *intents* que se encargan de mostrar los resultados del reconocimiento de voz.
SpeechRecognizer	Clase que proporciona acceso al servicio de reconocimiento de voz, el cual permite tener acceso al reconocedor de voz. No se debe crear un objeto de esta clase directamente, sino que se deberá llamar al método estático createSpeechRecognizer(Context). Permite crear el RecognitionListener que recibirá todas las notificaciones en respuesta a los eventos asociados al reconocedor de voz. Para poder utilizar esta clase, la aplicación debe tener el permiso RECORD_AUDIO

Tabla 2.3: Clases e interfaces del paquete android.speech

2.2.2.2 Descripción de la clase android.speech.RecognizerIntent

Como se explicó en la Sección 2.2.2, la manera más fácil para realizar el reconocimiento de voz es lanzando una aplicación que pueda procesar el *intent* del reconocimiento del habla, que es de tipo android.speech.RecognizerIntent. Esta actividad es llamada por la acción RecognizerIntent.ACTION_RECOGNIZE_SPEECH.

En la aplicación para dispositivos móviles Android del presente trabajo se utilizará Google Voice Search, una sencilla aplicación que consiste en un diálogo que solicita al usuario la entrada por voz. En el momento en que el locutor hace una pausa, el fichero de audio recogido se envía a los servidores de Google para ser procesado. Esto implica que sea necesaria una conexión a Internet. Finalmente, los resultados del reconocimiento de voz son devueltos a la actividad que lanzó dicho *intent.*

La Tabla 2.4 [16] resume las constante de la clase `Android.speech.RecognizerIntent` que se utilizan en la integración del reconocimiento de voz en una aplicación.

Tipo	Nombre	Definición
String	ACTION_GET_LANGUAJE_DETAILS	Invoca un *intent* de tipo *broadcast* que se envía a un objeto de la clase `BroadcastReceiver` con el fin de solicitar la lista de los idiomas disponibles del reconocedor de voz.
String	ACTION_RECOGNIZE_SPEECH	Inicia la actividad encargada de solicitar al usuario que realice su entrada por voz y de enviar dicha entrada al reconocedor de voz.
String	ACTION_VOICE_SEARCH_HANDS_FREE	Inicia la actividad encargada de solicitar al usuario que realice su entrada por voz sin requerir atención visual por parte del usuario o entrada táctil.
String	ACTION_WEB_SEARCH	Inicia la actividad encargada de solicitar al usuario que realice su entrada por voz y de enviar dicha entrada al reconocedor de voz. Desencadena algún tipo de acción basado en la petición del usuario, como un resultado de búsqueda web.
String	DETAILS_META_DATA	Nombre de los metadatos mediante el cual una actividad que implementa ACTION_WEB_SEARCH puede utilizar para revelar el nombre de la clase `BroadcastReceiver`,

		encargada de responder a cualquier solicitud de información por parte de los *intents* de tipo *broadcast* existentes en la clase `RecognizerIntent`
String	EXTRA_CALLING_PACKAGE	Se utiliza como clave extra en el *intent* de búsqueda por voz. Se añade a través de la llamada al método `putExtra(String name, String value)`
String	EXTRA_CONFIDENCE_SCORES	*Array* de datos de tipo *float* que representan la fiabilidad de cada resultado devuelto por la actividad correspondiente al *intent* ACTION_RECOGNIZE_SPEECH
String	EXTRA_LANGUAGE	Indica al reconocedor que realice el reconocimiento de voz en el idioma especificado por un código IETF, tal como define el estándar BCP 47.
String	EXTRA_LANGUAGE_MODEL	Informa al reconocedor sobre el modelo del lenguaje utilizado por el ACTION_RECOGNIZE_SPEECH: LANGUAGE_MODEL_WEB_SEARCH para frases de búsqueda o LANGUAJE_MODEL_FREE_FORM para dictado. La especificación de dicho *extra* es obligatoria.
String	EXTRA_LANGUAGE_PREFERENCE	Clave que identifica el contenido extra del resultado de tipo Bundle devuelto por el *intent* ACTION_GET_LANGUAJE_DETAILS a través del `BroadcastReceiver`. Se trata de una cadena que identifica el idioma que ha especificado el usuario.

String	EXTRA_MAX_RESULTS	Límite opcional del máximo número de resultados a devolver por el reconocimiento de voz. Si se omite es el reconocedor el que elige este límite.
String	EXTRA_ONLY_RETURN_LANGUAGE_PREFERENCE	Se especifica este booleano como extra al *intent* ACTION_GET_LANGUAGE_DETAILS para indicar que sólo se necesita la preferencia del idioma actual en la respuesta.
String	EXTRA_ORIGIN	Valor opcional que indica la URL de la página de la cual se solicita el habla
String	EXTRA_PARTIAL_RESULTS	Valor booleano opcional que indica si se deben devolver resultados parciales a medida que el usuario está hablando. Su valor por defecto es *false.*
String	EXTRA_PROMPT	Texto opcional que se muestra al usuario cuando se deba comenzar a hablar.
String	EXTRA_RESULTS	Los resultado del reconocimiento de voz de tipo ArrayList<String> cuando se ha realizado el *intent* ACTION_RECOGNIZE_SPEECH. Generalmente dicha lista se ordena de forma decreciente según la fiabilidad dado por EXTRA_CONFIDENCE_SCORES. Únicamente se devuelven estos resultados cuando la actividad devuelve el código RESULT_OK
String	EXTRA_RESULTS_PENDINGINTENT	Extra que sirve para proporcionar un `PendingIntent` de forma que los resultados del reconocimiento se añadan a su parámetro Bundle.

		Posteriormente, el `PendingIntent` se envía al destino.
String	EXTRA_RESULTS_PENDINGINTENT_BUNDLE	Si se utiliza EXTRA_RESULTS_PENDING_INTENT para proporcionar un `PendingIntent`, se puede utilizar EXTRA_RESULTS_PENDING_INTENT_BUNDLE para añadir extras adicionales al `PendingIntent` que se envía al destino además de los resultados de reconocimiento de voz.
String	EXTRA_SECURE	Valor booleano opcional que indica si una búsqueda si un búsqueda por voz en modo manos libres se efectúo en modo seguro (Ej. Pantalla del dispositivo bloqueada)
String	EXTRA_SPEECH_INPUT_COMPLETE_SILENCE_LENGTH_MILLIS	Extra que especifica el tiempo que debe pasar durante una pausa del usuario para considerar que la entrada por voz se ha completado.
String	EXTRA_SPEECH_INPUT_MINIMUM_LENGTH_MILLIS	Extra que especifica la longitud mínima (en tiempo) de una elocución. La grabación no para hasta finalizar este periodo de tiempo.
String	EXTRA_SPEECH_INPUT_POSSIBLY_COMPLETE_SILENCE_LENGTH_MILLIS	Extra que especifica el tiempo que debe pasar durante una pausa del usuario para considerar que la entrada por voz se ha completado posiblemente.
String	EXTRA_SUPPORTED_LANGUAGES	La clave que identifica el contenido extra del resultado de tipo Bundle devuelto por el *intent* ACTION_GET_LANGUAGE_DATEILS a través del `BroadcastReceiver`. Es un

		ArrayList<String> y contiene los idiomas disponibles en la implementación actual del reconocimiento.
String	EXTRA_WEB_SEARCH_ONLY	Booleano opcional que se usa junto a ACTION_WEB_SEARCH para indicar que se muestren solamente búsquedas web como respuesta a la entrada de voz del usuario.
String	LANGUAGE_MODEL_FREE_FORM	Establece el modo de lenguaje FREE_FORM para dictado. Se usa junto a la clave EXTRA_LANGUAGE_MODEL
String	LANGUAGE_MODEL_FREE_FORM	Establece el modo de lenguaje WEB_SEARCH para búsquedas en la web. Se usa junto a la clave EXTRA_LANGUAGE_MODEL
Int	RESULT_AUDIO_ERROR	Código de respuesta cuando se da un error de audio
Int	RESULT_CLIENT_ERROR	Código de respuesta cuando se da un error genérico del cliente.
Int	RESULT_NETWORK_ERROR	Código de respuesta cuando se da un error en la red.
Int	RESULT_NO_MATCH	Código de error cuando no se ha encontrado ninguna coincidencia con las palabras dichas.
Int	RESULT_SERVER_ERROR	Código de respuesta cuando un servidor de reconocimiento de voz devuelve un error.

Tabla 2.4: Constante de la clase `Android.speech.RecognizerIntent`

La clase Android.speech.RecognizerIntent consta de un único método público descrito en la Tabla 2.5 [16]

Final static Intent getVoiceDetailsIntent (Context context)
Devuelve un *intent* de tipo broadcast que se lanza mediante el método sendOrderBroadcast(Intent, String, BroadcastReceiver, android.os.Handler, int, String, Bundle) con el objetivo de recibir información del paquete que implementa la búsqueda por voz. Está basado en el valor especificado en los metadatos DETAILS_META_DATA de la actividad encargada de la búsqueda por voz. Si dicho valor no está especificado se devuelve *null.* También devuelve *null* si no se elige una actividad que responda por defecto al ACTION_WEB_SEARCH.

Tabla 2.5: Método *getVoiceDetailsInten* de la clase *RecognizerIntent*

2.2.2.3 Integración del reconocimiento de voz en una aplicación Android mediante el uso de la clase android.speech.RecognizerIntent

En esta sección se describen los pasos a seguir para integrar el reconocimiento de voz en una aplicación para dispositivos móviles Android.

En primer lugar, hay que dar a la aplicación los permisos necesarios para que soporte el reconocimiento de voz. Dado que el reconocimiento de voz se hace a través de los servidores de Google y es necesario que la aplicación tenga acceso a internet, se dará dicho permiso en el fichero AndroidManifest.xml con la línea de código mostrada en la Figura 2.5.

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

Figura 2.5: Permiso de acceso a Internet en el archivo AndroidManifest.xml

Una vez realizado este paso previo, se puede comenzar con la implementación del reconocedor de voz. Toda aplicación que integre reconocimiento de voz implementa tres funciones diferenciadas que se describen a continuación [17]:

Comprobar que existe la actividad de reconocimiento de voz

Antes de lanzar la aplicación de reconocimiento de voz, es necesario comprobar que dicha aplicación está instalada y disponible en el dispositivo. Para ello se puede utilizar el método `queryIntentActivities` de la clase `PackageManager`, clase que obtiene información de los paquetes que están instalados en el dispositivo. Este método devuelve una lista de actividades existentes en el dispositivo que pueden procesar la Intención especificada en su primer parámetro. En el caso de la aplicación de

reconocimiento de voz, el *intent* a especificar es `RecognizerIntent.ACTION_RECOGNIZE_SPEECH`. Si el tamaño de la lista devuelta por dicho método es distinto de cero, implica que el servicio de reconocimiento de voz está disponible en el dispositivo. Para recibir una instancia de `PackageManager` se utiliza el método `getPackageManager`. La Figura 2.6 muestra el código que implementa esta funcionalidad.

```
private boolean ExisteActividadReconocerVoz(){
    PackageManager pm = getPackageManager();
    List<ResolveInfo> activities =pm.queryIntentActivities(
            new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH), 0);
        if(activities.size() != 0){
            return true;
        }else{
            return false;
        }

}
```

Figura 2.6: Método que comprueba la existencia de la actividad de reconocimiento de voz en el dispositivo

Iniciar la actividad de reconocimiento de voz

Una vez que se ha verificado que existe la aplicación de reconocimiento de voz se puede proceder a invocar tal actividad. En la Figura 2.7 se muestra el código que implementa esta acción.

```
private void iniciarActividadReconocimientoDeVoz(){
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_WEB_SEARCH);
    intent.putExtra(RecognizerIntent.EXTRA_PARTIAL_RESULTS, false);
    intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 3);
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT,
            "Demostración del reconocimiento de voz");

    startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
}
```

Figura 2.7: Método que inicial la aplicación de reconocimiento de voz

En primer lugar, se crea un objeto de tipo *Intent* pasándole como parámetro la acción `RecognizerIntent.ACTION_RECOGNIZE_SPEECH`, que será la encargada de solicitar al usuario que realice su entrada por voz y de enviar dicha entrada al reconocedor de voz.

A continuación, a través de este *Intent* se configuran los parámetros de la actividad de reconocimiento de voz utilizando el método `putExtra(String name, String value)`. Este par de parámetros <clave, valor> pueden tomar cualquiera de los valores de la Tabla 2.4, siendo únicamente obligatorio especificar el modelo del lenguaje utilizado con EXTRA_LANGUAGE_MODEL.

En el ejemplo de la imagen, se especifica el modelo del lenguaje con LANGAUGE_MODEL_WEB_SEARCH, indicado para frases cortas típicas de búsquedas web. Además, se indica que no se devuelvan resultados parciales con EXTRA_PARTIAL_RESULTS, se establece el número máximo de resultados devueltos a tres y se añade un mensaje para mostrar al usuario a través de EXTRA_PROMPT.

Por último, se invoca la actividad de reconocimiento de voz indicada en el *Intent* a través del método `startActivityForResult(Intent intent, int requestCode)` de manera que los resultados del reconocimiento se recuperen en el método `onActivityResult(int requestCode, int resultCode, Intent data)`. La constante VOICE_RECOGNITION_REQUEST_CODE se utiliza para identificar a la actividad que invoca a dicho *Intent* y se declara al principio de la actividad.

Procesar los resultados

Como último paso, se procesan los resultados devueltos por la actividad del reconocedor de voz en el método `onActivityResult`. En la Figura 2.8 se muestra un código de ejemplo de este método.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data){

    if (requestCode == VOICE_RECOGNITION_REQUEST_CODE && resultCode == RESULT_OK) {

        ArrayList<String> matches = data.getStringArrayListExtra
                (RecognizerIntent.EXTRA_RESULTS);
        super.onActivityResult(requestCode, resultCode, data);

    }

}
```

Figura 2.8: Método onActivityResult para procesar los datos del reconocimiento de voz

En el método se comprueba en primer lugar si la actividad invocada ha sido el reconocedor de voz y que el resultado haya sido satisfactorio comprobando el valor de las variables *requestCode* y *resultCode,* respectivamente. Si se dan estas condiciones, se procede a recuperar el resultado del reconocedor a través del método getStringArrayListExtra y se guardan en una cadena de objetos (*ArrayList*) de tipo *String.* A partir de este punto, es el desarrollador el que debe decidir qué hacer con estos resultados.

2.3 Síntesis de Texto a Voz en Android

2.3.1 Introducción a la síntesis de texto a voz en Android

La síntesis de texto a voz (TTS*, Text-to-Speeh Synthesis*) [4] es un proceso relacionado con las tecnologías del habla que tiene como objetivo transformar textos escritos en su realización oral, es decir, efectuar la lectura en voz alta de un texto. En un sistema de diálogo, se trata del último módulo, encargado de hacer llegar al usuario por vía oral el resultado obtenido por el sistema.

A partir de la versión 1.6, se incorpora la síntesis de texto a voz en la plataforma Android lo que permite a las demás aplicaciones beneficiarse de este tipo de interfaz. Esto se realiza a través de un motor TTS que está, generalmente, instalado por defecto en el sistema y que es utilizado por las demás aplicaciones. El más habitual es el denominado Pico TTS, desarrollado por la compañía SVOX y Google, que se encuentra instalado en la mayoría de los dispositivos, pero existen multitud de alternativas que serán estudiadas en la Sección 2.3.3.

Figura 2.9: Menú de ajuste de la síntesis de voz

La Figura 2.9 muestra el menú de ajustes de la síntesis de voz en dispositivos Android 1.6 o superior, que consta de las siguientes opciones:

- **Escuchar un ejemplo:** Permite escuchar un ejemplo del motor de síntesis que esté seleccionado en el dispositivo como motor predeterminado y en el idioma que se encuentre seleccionado en la opción Idioma.

- **Usar siempre mi configuración:** Permite anular la configuración de cualquier aplicación que utilice el motor TTS. Por ello, es preferible dejar esta opción sin seleccionar si se desea que la aplicación tenga el comportamiento esperado.
- **Motor predeterminado:** establece el motor de síntesis de voz utilizado. En los dispositivos Android 1.6 o superior viene incorporado el motor Pico TTS por defecto por lo que solo aparece esta opción si no se ha instalado ningún otro motor. En las últimas versiones de Android, el motor de síntesis que viene incorporado por defecto es Google TTS que ofrece unas voces de calidad muy similar a las del motor Pico TTS.
- **Instalar archivos de datos de voz:** Permite instalar los archivos de voz necesarios para la síntesis de voz. Es necesario que esté instalado el paquete *SpeechSynthesis Data Installer,* que contiene los ficheros necesarios para la síntesis de texto a voz en los idiomas inglés EEUU/UK, francés, italiano, alemán, y español.
- **Velocidad de voz:** Permite seleccionar la velocidad a la que se lee el texto, a elegir entre cinco niveles.
- **Idioma:** Permite establecer el idioma al que se lee el texto. En el caso de Pico TTS, las opciones son: Inglés EEUU/UK, francés, italiano, alemán y español.

2.3.2 Integración de la síntesis de texto a voz en una aplicación para dispositivos móviles Android

La lógica interna utilizada por el motor de síntesis de voz es bastante compleja. Sin embargo, la plataforma Android permite integrar en cualquier aplicación este sistema de una forma sencilla sin necesidad de que el desarrollador conozca su metodología. Para la implementación de una aplicación basada en la síntesis de texto a voz, únicamente es necesario que ésta se encargue de enviar el texto que se quiere leer en voz alta al motor de síntesis para que lo reproduzca. Esta funcionalidad se consigue a partir del paquete `android.speech.tts` de la API de Android, en particular, la clase `android.speech.tts.TextToSpeech`. Aunque existen otras alternativas, como la utilización del SDK desarrollado por la empresa iSpeech [18], la utilización de dicho paquete es la más simple y es la que se utilizará en la aplicación para dispositivos móviles Android del presente trabajo de fin de grado.

En las siguientes secciones se resumen las clases e interfaces que componen dicho paquete, así como los aspectos a tener en cuenta a la hora de integrar la síntesis de texto a voz en una aplicación Android.

2.3.2.1 Descripción del paquete android.speech.tts

Tal y como se ha comentado previamente, el paquete `android.speech.tts` proporciona las clases necesarias para integrar la síntesis de voz en una aplicación Android a partir de un motor de síntesis disponible en el dispositivo.

La Tabla 2.6 [16] describe las clases e interfaces disponibles en el paquete `android.speech.tts`. Para una información más detallada de dicho paquete se puede consultar el API de Android [16].

Interfaz	Definición
SinthesisCallback	Define el método que devuelve los datos de voz sintetizada por el motor de síntesis.
TextToSpeech.OnInitListener	Define el método que es invocado cuando termina la inicialización del motor de síntesis.
TextToSpeech.OnUtteranceCompletedListener	Deprecada a partir del nivel 18 del API. En su lugar se usa UtteranceProgressListener

Clase	Definición
SinthesisRequest	Contiene los datos requeridos por el motor para realizar la síntesis de texto a voz como son el texto a sintetizar, el idioma, la velocidad o el tono de voz, entre otros.
TextToSpeech	Se encarga de la síntesis de texto a voz cuyo resultado será reproducido o guardado en un fichero de audio. Una instancia de la clase TextToSpeech puede únicamente sintetizar un texto cuando se haya completado su inicialización. Se debe implementar la interfaz TextToSpeech.OnInitListener para notificar a la aplicación una vez que la inicialización haya finalizado.
TextToSpeech.Engine	Contiene las constantes y nombres de parámetros que se utilizan para el control de la síntesis de texto a voz.
TextToSpeech.EngineInfo	Contiene las constantes y parámetros necesarios para el control del sintetizador de texto a voz.
TextToSpeechService	Obtiene la información sobre el motor de síntesis de texto a voz instalado en el dispositivo.
UtteranceProgressListener	Controla los eventos relacionados con el progreso de una cadena de texto a través de la pila cola que contiene los textos a ser sintetizados.
Voice	Características y funciones de la voz de un

	sintetizador de texto a voz. Cada motor TTS puede tener múltiples voces para cada localidad, con un conjunto diferente de características cada una.

Tabla 2.6: Clases e interfaces del paquete android.speech.tts

2.3.2.2 Comprobación de los recursos necesarios para la síntesis de texto a voz e inicialización de la instancia TextToSpeech

Los motores de síntesis de texto a voz soportan diferentes idiomas y acentos, recursos que deben ser especificados antes de que el motor de síntesis pueda comenzar a hablar. Sin embargo, algunos dispositivos pueden no tener disponible alguno de estos recursos por limitaciones de espacio en memoria. Por este motivo, la API de Android permite consultar a la plataforma la disponibilidad de los archivos relacionados con el lenguaje elegido e iniciar su descarga en el caso de que estos no se encuentren instalados en el terminal.

En el desarrollo de una aplicación que integre la síntesis de texto a voz, una buena práctica es verificar antes de nada la existencia de los recursos mencionados. Esto se realiza con el uso del *intent* `TextToSpeech.Engine.ACTION_CHECK_TTS_DATA` tal y como se muestra en la Figura 2.10. Al lanzar este *intent* con el método `startActivityForResult` de la clase `Activity`, se devuleve un código de respuesta resultado de la comprobación que es recogido por el método `onActivityResult`, también de la clase `Activity`. La constante MY_DATA_CHECK_CODE sirve como identificador de este *intent* y se declara al principio de la actividad.

```
Intent checkIntent = new Intent();
checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
startActivityForResult(checkIntent, MY_DATA_CHECK_CODE);
```

Figura 2.10: Código desarrollado para comprobar la disponibilidad de los recursos TTS

```
private TextToSpeech mTts;
protected void onActivityResult(
        int requestCode, int resultCode, Intent data) {
    if(requestCode == MY_DATA_CHECK_CODE) {
        if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS) {
            //existen los recursos, se crea la instancia del TTS
            mTts = new TextToSpeech(this, this);
        }else{
            //no existen los recursos, se solicita su instalación
            Intent installIntent = new Intent();
            installIntent.setAction(
                    TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
            startActivity(installIntent);
        }
    }
}
```

Figura 2.11: Código desarrollado para crear la instancia de la clase TextToSpeech y solicitar la instalación de los recurso si estos no existen.

El método `onActivityResult` se muestra en la Figura 2.11. En primer lugar, se comprueba que el valor de la variable `requestCode` coincide con el de la constante MY_DATA_CHECK_CODE y de ser así se procede a comprobar el código del resultado devuelto por la actividad. En caso de que los recursos solicitados se encuentren disponibles en el dispositivo, el código resultado toma el valor `TextToSpeech.Engine.CHECK_VOICE_DATA_PASS`, lo que significa que el dispositivo está preparado para utilizar el sintetizador una vez que se haya creado el objeto `android.speech.tts.TextToSpeech`. Si el código resultado toma cualquier otro valor significa que los recursos del motor de síntesis no están disponibles por lo que se procede a lanzar el *intent* `TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA` que redirige al usuario a *Play Store* [56] para solicitar la descarga e instalación de los mismos.

El constructor de la clase `TextToSpeech` consta de dos parámetros de entrada: una referencia a la clase `Context` y una referencia a la interfaz `TextToSpeech.OnInitListener` [16] que, en el ejemplo de la Figura 2.11, es la propia actividad en ambos casos.

Es necesario que la actividad implemente la interfaz `TextToSpeech.OnInitListener` para que se notifique a la aplicación cuando el objeto `TextToSpeech` ha sido inicializado. Esta notificación se realiza a través del método `onInit(int status)` de dicha interfaz, del cual se muestra un ejemplo en la Figura 2.13. Su parámetro de entrada permite verificar el estado de la instanciación y tomará el valor `TextToSpeech.SUCCESS` en el caso de que se haya realizado con éxito, lo que indica que el motor de síntesis está preparado para ser configurado y utilizado. Es por esto que es aconsejable especificar en este punto las propiedades del objeto `TextToSpeech`, que son las características del motor de síntesis como el idioma o la velocidad.

2.3.2.3 Configuración del motor de síntesis: selección del idioma

La clase `TextToSpeech` proporciona varios métodos que permiten la configuración del idioma del sintetizador de voz:

- `int isLanguageAvailable(Locale loc)` [16]: permite comprobar si el idioma representado por la variable de tipo *Locale* se encuentra disponible. La clase `Locale` permite especificar un idioma, un país y un tercer parámetro que representa una variante del idioma a través del constructor `public Locale (String Language, String country, String variant)`. El resultado devuelto por el método es un código que representa cuáles de los parámetros del constructor están disponibles y pueden tomar los siguientes valores:
  - `LANG_AVAILABLE`: indica que el idioma está disponible pero no el país ni la variante.
  - `LANG_COUNTRY_AVAILABLE`: indica que el idioma y el país están disponibles pero no la variante.
  - `LANG_COUNTRY_VAR_AVAILABLE`: indica que el idioma está disponible exactamente como se especifica en el constructor.
  - `LANG_MISSING_DATA`: los recursos del idioma solicitado no están instalados.
  - `LANG_NOT_SUPPORTED`: el idioma no está disponible.

- `int setLanguage(Locale loc)`: establece el idioma al motor TTS representado en el parámetro de entrada. Los códigos de retorno de este método son los mismos que los del método anterior.

Utilizando el método `getDefault()` de la clase `Locale`, se obtiene el valor de la configuración local actual en el dispositivo. De esta manera, junto con el método `getLanguage()` de esta misma clase, se puede obtener el idioma configurado en el dispositivo tal y como se muestra en la Figura 2.12.

```
String language = Locale.getDefault().getLanguage();
```

Figura 2.12: Línea de código para obtener el idioma configurado en el dispositivo

La Figura 2.13 muestra un ejemplo en el que se selecciona como idioma del motor TTS el español hablado en España una vez que se ha comprobado que éste está disponible.

```
@Override
public void onInit (int status){
    if(status == TextToSpeech.SUCCESS){
        if(tts.isLanguageAvailable(new Locale("spa","ESP")) >= 0){
            tts.setLanguage(new Locale("spa", "ESP"));
        }
    }
}
```

Figura 2.13: Código desarrollado para seleccionar el idioma del motor TTS

En el ejemplo de la imagen, esta acción se encuentra dentro del método `onInit()` de la interfaz `TextToSpeech.OnInitListener`. Como se ha visto en la sección anterior, es aconsejable realizar la configuración del TTS dentro de este método.

2.3.2.4 Reproducción de una cadena de texto con el sintetizador de texto a voz

Una vez que se ha instanciado y configurado el objeto `android.speech.tts.TextToSpeech`, el motor de síntesis está preparado para reproducir cualquier cadena de texto. Para ello, se pueden utilizar dos métodos llamados a través del objeto mencionado, el primero de ellos remplazado por el segundo en la versión 21 del API de Android:

`int speak (String text, int queueMode, HashMap<String, String> params)`

Con este método, el motor de síntesis trata y reproduce una cola de cadenas de textos que van a ser sintetizadas, denominadas, en términos TTS, elocuciones. Este método es asíncrono, es decir, cuando se invoca simplemente añade el texto a reproducir a la cola y finaliza incluso aunque la síntesis no haya empezado. La función y descripción de cada parámetro de entrada es la siguiente:

- *text:* especifica el texto que va a ser reproducido. La longitud de éste, es decir, el valor entero retornado por el método `lenght()`, llamado a través de dicho parámetro, no puede ser mayor al valor devuelto por el método `getMaxSpeechInputLenght()` de la clase `TextToSpeech`.
- *queueMode:* especifica la estrategia de encolamiento de las diferentes elocuciones. Este parámetro puede tomar dos valores: `QUEUE_ADD` y `QUEUE_FLUSH`. Si se utiliza el primero de ellos, la elocución se añade a la cola en la última posición y espera su turno para ser reproducida. Por el contrario, `QUEUE_FLUSH` indica que la presente elocución se debe reproducir en el instante, interrumpiendo cualquier proceso de síntesis de texto a voz que se estuviese efectuando en dicho momento.
- *params:* permite indicar al motor TTS parámetros opcionales, que se especifican a través del par clave/valor de una estructura HashMap. Se puede

no especificar ninguna opción y poner este parámetro a *null.* Los valores permitidos para la clave de la estructura HashMap son los siguientes:

- KEY_PARAM_STREAM: indica el canal de salida de audio al que dirigir la voz sintetizada. Los posibles valores que puede tomar esta opción se muestran en la Tabla 2.7 [16].

Constante	Descripción
STREAM_ALARM	Salida de audio para alarmas
STREAM_DTMF	Salida de audio para tonos DMTF
STREAM_MUSIC	Salida de audio para la reproducción de música
STREAM_NOTIFICATION	Salida de audio para las notificaciones
STREAM_RING	Salida de audio para el tono de llamadas
STREAM_SYSTEM	Salida de audio para sonidos del sistema
STREAM_VOICE_CALL	Salida de audio para la voz en las llamadas

Tabla 2.7: Posibles valores para la clave KEY_PARAM_STREAM

- KEY_PARAM_UTTERANCE_ID: permite identificar una elocución cuando esta finaliza, lo que resulta útil si se quiere realizar alguna acción inmediatamente después de haberse realizado la síntesis de texto a voz o para detectar errores durante la síntesis. Para ello, el objeto `android.speech.tts.TextToSpeech` debe invocar al *listener* `android.speech.tts.UtteranceProgressListener` a través del método `setOnUtteranceProgressListener(UtteranceProgressListener mProgressListener)`, donde el parámetro de entrada es una instancia de la clase mencionada. Esta clase consta de tres métodos abstractos que deben ser implementados: `onDone(String utteranceId)`, `onError (String utteranceId)` y `onStart(String utteranceId)` a los que se les pasa como parámetro el identificador de la cadena de texto que acaba de ser sintetizada. El primero de ellos es invocado cuando una elocución es completada con éxito por lo que dentro de él se ubica el código de la acción que se quiere realizar en dicho momento.
  Para niveles del API inferiores a 15 se puede utilizar el `setOnUtteranceCompletedListener(TextToSpeech.onUtteranceCompletedListener listener)` en vez de `setOnUtteranceProgressListener (UtteranceProgressListener mProgressListener)`, sin embargo fue deprecado a partir de este nivel y reemplazado por el anteriormente explicado [16]

- KEY_PARAM_VOLUME: permite especificar el volumen de voz relativo de la elocución actual a partir de un valor de tipo *float* cuyo rango oscila

entre 0 y 1, siendo 0 silencio y 1 el valor máximo de volumen (valor por defecto si no se especifica este parámetro).

- KEY_PARAM_PAN: parámetro relacionado con la panoramización del sonido. Se especifica con un valor de tipo *float* cuyo rango oscila entre – 1 y +1, donde el valor -1 envía todo el sonido a la izquierda, el valor +1 envía todo el sonido a la derecha y el valor 0 representa el centro.

El valor retornado por el método es de tipo *int* y puede tomar los valores `TextToSpeech.ERROR` y `TextToSpeech.SUCCESS`, indicando de esta manera si la solicitud se ha llevado a cabo con éxito.

`int speak (CharSequence text, int queueMode, Bundle params, String utteranceId)`

Como se ha explicado, el método anterior está en desuso desde la versión 21 del API. La funcionalidad de este nuevo método es exactamente la misma que la del anterior aunque varíe el tipo de los parámetros de entrada. Además, se añade un cuarto parámetro de entrada que equivale a la constante KEY_PARAM_UTTERANCE_ID utilizada en el método anterior, es decir, se trata de un identificador único para la solicitud actual. Los posibles valores que devuelve el método una vez que se ha ejecutado son los mismos que en el caso anterior.

2.3.2.5 Almacenamiento del resultado de la síntesis de texto a voz en un fichero

En ocasiones es útil guardar el resultado de la síntesis de texto a voz en un fichero de audio [19]. Esto ocurre, por ejemplo, cuando la misma elocución va a ser repetida a menudo. De esta manera, no es necesario realizar la síntesis cada vez que se quiera reproducir cierto texto sino que basta con reproducir el fichero de voz obtenido, consiguiendo un ahorro en el consumo de recursos y, por tanto, una mayor rapidez en la ejecución de la aplicación.

Esto se realiza a través del método `public int synthesizeToFile(CharSequence text, Bundle params, File file, String utteranceId)` de la clase `TextToSpeech`. Este método reemplaza al método `public int synthesizeToFile(CharSequence text, HashMap<String, String> params, String filename)` que es deprecado a partir del nivel 21 del API, cuya funcionalidad es exactamente la misma diferenciándose en el tipo de parámetros de entrada.

Al igual que el método `speak`, este método funciona de forma asíncrona, es decir, añade el texto a reproducir a la cola cuando es invocado y finaliza incluso aunque la síntesis no haya empezado.

La función y descripción de cada parámetro de entrada es la siguiente:

- *text:* especifica la cadena de texto que va a ser sintetizado. La longitud de éste, es decir, el valor entero retornado por el método `lenght()` llamado a través de dicho parámetro, no puede ser mayor al valor devuelto por el método `getMaxSpeechInputLenght()` de la clase `TextToSpeech`.
- *params:* permite indicar al motor TTS parámetros opcionales, que se especifican precedidos por el nombre del motor al que van a ser asociados. Se puede no especificar ninguna opción y poner este parámetro a *null.*
- *file:* fichero en el que se almacenará el audio generado.
- *utteranceId:* identificador único para la solicitud.

El valor retornado por el método es de tipo *int* y puede tomar los valores `TextToSpeech.ERROR` y `TextToSpeech.SUCCESS`, indicando de esta manera si la solicitud se ha llevado a cabo con éxito.

La clase `TextToSpeech` ofrece otras dos posibilidades para asociar una locución con un fichero de audio. La primera de ellas consiste en invocar el método `public int addSpeech(String text, String filename)`, el cual realiza un mapeo entre la cadena de texto del primer parámetro y el fichero de audio cuyo nombre se especifica en el segundo parámetro. La segunda posibilidad es indicar el nombre del paquete en el que se encuentra el recurso de audio así como el identificador del recurso utilizando el método `public int addSpeech (CharSequence text, String packeganame, int resourceId)`. En ambos casos, después de la llamada a dichos métodos, cualquier invocación al método `speak(String, int, HashMap)` reproducirá el fichero de audio correspondiente a la cadena de texto si está disponible, o sintetizará el texto si no se encuentra tal recurso.

2.3.2.6 Reproducción de Earcons

Un *earcon,* o icono acústico, es un sonido breve y característico usado para representar un determinado evento. Para añadir un earcon a una aplicación, la instancia `android.speech.tts.TextToSpeech` dispone de dos métodos. El primero de ellos, `addEarcon(String earcon, File file)`, que sustituye al método `addEarcon(String earcon, String filename)` deprecado en la versión 21 del API, utiliza como parámetros de entrada la cadena de texto que representa el earcon y el nombre del fichero de audio al que se va a asociar, respectivamente. La representación del earcon debe estar contenido entre corchetes (E.j. “[tick]”). La segunda opción es el uso del método `addEarcon(String earcon, String packagename, int resourceId)` cuyos parámetros de entrada son el nombre del earcon, el nombre del paquete que contiene el recurso de audio y el identificador del recurso. Como se puede observar, estos métodos tienen cierta similitud con los métodos `addSpeech()` estudiados anteriormente.

Una vez que se ha añadido el earcon, se invoca el método `public int playEarcon(String earcon, int queueMode, Bundle params, String utteranceId)` cuyos parámetros son los mismos que los del método `int speak (CharSequence text, int queueMode, Bundle params, String utteranceId)` y su funcionamiento es similar a este.

2.3.2.7 Inserción de pausas entre dos elocuciones

El método `public int playSilentUtterance (long durationInMs, int queueMode, String utteranceId)` de la clase `android.speech.tts.TextToSpeech` es útil para forzar una pausa entre dos elocuciones, reproduciendo un silencio durante el tiempo, en milisegundos, indicado en el primer parámetro. El segundo y tercer parámetro son el modo de encolamiento y el identificador de la elocución, respectivamente, al igual que los análogos del método `speak(CharSequence text, int queueMode, Bundle params, String utteranceId)`. Además, al igual que el método `speak`, este método es también asíncrono.

2.3.2.8 Finalización del motor de síntesis

La síntesis de texto a voz está compartida por todas las aplicaciones que utilizan esta funcionalidad. Por ello, es aconsejable que una vez que la aplicación haya terminado de utilizar dicha funcionalidad, se invoque el método `shutdown()` de la instancia `android.speech.tts.TextToSpeech` dentro del método `onDestroy()` de la actividad, tal y como se muestra en la Figura 2.14. La llamada a `shutdown()` libera todos los recursos utilizados por el motor TTS.

```
public void onDestroy() {
    if(tts != null) {
        tts.stop();
        tts.shutdown();
    }
    super.onDestroy();
}
```

Figura 2.14: Código desarrollado para la finalización del motor de síntesis

La llamada al método `stop()` es útil antes de invocar el método `shutdown()` para interrumpir la cadena de texto que esté siendo procesada y descartar las restantes de la cola.

2.3.2.9 Otros métodos de interés de la clase TextToSpeech

A continuación se enumeran otros métodos de la clase `android.speech.tts.TextToSpeech` que resultan útiles a la hora de desarrollar una aplicación que integre un motor TTS [16]:

- `isSpeaking()`: Devuelve *true* o *false* indicando si el motor TTS está reproduciendo o no una cadena de texto en ese instante.
- `setPitch(float pitch)`: Especifica el tono de la voz, más grave o más agudo, a través del valor del parámetro de entrada. El valor por defecto es 1.0, valores menores disminuyen el tono y valores mayores lo aumentan.
- `setSpeechRate(float speechRate)`: Especifica la velocidad de la elocución a través del valor del parámetro de entrada. El valor por defecto es 1.0, valores menores disminuyen la velocidad y valores mayores la aumentan.

2.3.3 Motores de síntesis de voz (TTS) ofrecidas por la plataforma Android

Todo dispositivo Android, por lo general, lleva incorporado un motor TTS instalado por defecto, que es utilizado por las demás aplicaciones para leer textos en voz alta.

Como se describió en la Sección 2.3.1, el motor de síntesis de texto a voz Pico TTS es el más común. Sin embargo, Android ofrece la posibilidad de instalar y personalizar varios motores de síntesis, aunque siempre eligiendo uno como motor TTS principal a través del menú de ajustes del dispositivo[20]. Cabe destacar que la mayoría de las aplicaciones existentes que utilizan síntesis de texto a voz, se limitan a utilizar el motor de síntesis activado como principal en el dispositivo.

A continuación, se resumen las características de los principales motores TTS que ofrece la plataforma Android.

- **Pico TTS:** como se ha comentado anteriormente, es el sintetizador de voz por defecto en la mayoría de dispositivos Android. En comparación con otros motores de síntesis disponibles en Android, la calidad de la voz es regular, ya que resulta algo monótona y robótica. Sin embargo, al motor Pico TTS apenas consume memoria y espacio en disco, lo que supone una gran ventaja frente a otros sintetizadores. El motor Pico TTS está desarrollado por la compañía SVOX, está disponible en inglés, alemán, francés, italiano y español y su descarga es gratuita.
- **IVONA TTS HQ:** IVONA [21] es uno de los mejores sintetizadores de voz que existen para Android. Se trata de una voz muy clara, realista y con buen acento. Además, posee multitud de voces e idiomas que pueden ser testeados en su propia página web. En español se dispone de las voces de Conchita y Penélope.

A pesar de su alta calidad, IVONA es actualmente gratuito, ya que está en fase beta.

- **SVOX Classic TTS:** sintetizador de texto creado por la compañía SVOX, una de las empresas con más tiempo y experiencia en el sector. Al igual que IVONA, dispone de multitud de idiomas y voces. En español dispone de las voces de Noelia y Pablo, ambas de muy buena calidad. Además, las opciones de este sintetizador permiten aumentar o reducir el volumen de voz, el tono o mejorar el rendimiento en el caso de móviles de altas prestaciones. Su precio actual es de 2,99 euros cada voz.
- **Samsung TTS:** sintetizador de voz creado por la compañía Samsung que se encuentra incorporado en los dispositivos de dicha marca. Está disponible en coreano, inglés EEUU, francés, alemán y español, todas ellos con voces femeninas. La calidad de la voz es regular, similar a la del sintetizador de voz Pico TTS. Como novedad, permite activar ciertos efectos de voz: normal, profunda, alta, fina, gruesa, robótica o con efecto Helio.
- **CereProc:** se trata de una empresa escocesa de amplia experiencia en todos los campos relacionados con la tecnología de síntesis de voz. Para Android, CereProc dispone de varios sintetizadores de voz entre los que se encuentran Caitlin (irlandés), Adam, Isabella y Katherine (inglés) y Stuart y Heather (escocés) a un precio de 1,45 euros cada uno. Estos sintetizadores son de gran calidad ya que están basados en las soluciones de síntesis de voz que utiliza CereProc para sus ordenadores de mesa. Por ejemplo, Adam se trata de la mejor voz en cuanto a calidad existente para inglés americano.

  También ofrece algunas voces gratuitas, de poca utilidad pero curiosas, como PigLatin (un juego infantil, *el juego de los cerdos*), Dog (sintetizador a voz de perro), Dodo (una especie de broma similar al pato Donald) y Nicole Sexy French.
- **eSpeak TTS:** Sintetizador de texto basado en el software de código abierto *eSpeak* [22], diseñado por *Eyes-Free Project* para funcionar en plataformas Android. Este sintetizador utiliza como método la síntesis por formantes, por lo que la calidad y naturalidad de las voces disminuye, sin embargo, esto permite proporcionar numerosos idiomas ocupando poco espacio. El sintetizador permite cambiar algunas opciones como tono, velocidad o género. Su descarga está disponible gratuitamente.
- **FLite TTS:** FLite es un sintetizador de voz desarrollado por la Carnegie Mellon University, originalmente desarrollado en Festvox y, posteriormente, portado a Android. Se encuentra sólo disponible en diferentes acentos de inglés, la calidad de la voz es relativamente buena y su descarga es gratuita
- **Loquendo TTS:** El conocido sintetizador de voz *Loquendo TTS* [23] tiene también su versión para Android aunque esta lleva un tiempo sin estar disponible en Google Play [56] por razones desconocidas.

  Loquendo dispone de más de 70 voces en más de 30 idiomas, incluyendo también voces en gallego, catalán, valenciano, etc. Sin embargo, para Android solo están disponibles las voces de Susan (Inglés EEUU) y Paola (Italiano).

Además, Loquendo utiliza un sistema para reproducir ciertas emociones, como risas, llantos o besos.

- **Ekho TTS:** Ekho TTS es un proyecto Open Source para crear un motor TTS para invidentes en chino, que funciona tras el proyecto eGuideDog [24] dedicado a desarrollar software gratuito para invidentes. Permite como opciones de idioma el cantonés y el mandarín. Actualmente, no se encuentra disponible para la versión 4 de Android (*Ice Cream)* o superiores pero se está trabajando en la compatibilidad con estas versiones.
- **Vaja TTS:** Vaja TTS [25] es un sintetizador que incluye idiomas inglés y tailandés. Tiene una única voz, Nok, que habla ambos idiomas. Además, la aplicación permite ajustar el volumen, velocidad y tono de la voz.

2.4 Aplicaciones para dispositivos móviles Android con ASR y TTS

2.4.1 Aplicaciones que integran motores de síntesis de texto a voz.

La mayor parte de aplicaciones Android que integran síntesis de texto a voz, se limitan a utilizar el motor TTS para realizar lecturas de texto. Incorporar motores TTS en las aplicaciones hace el manejo de estás más intuitivo y completo. Esta función es especialmente útil para personas con problemas de visión o para su utilización en entornos en los que no es posible o poco aconsejable el uso de interfaces visuales tradicionales.
A continuación, se describen algunos ejemplos de aplicaciones TTS en Android disponibles en Google Play [56], clasificadas por su funcionalidad [26]:

- Indicaciones en voz alta de direcciones de lugares. Ejemplos:
  - *Google Maps* [51]*:* servidor de aplicaciones de mapas en la web perteneciente a Google. Junto con *Google Navigation,* es capaz de dar indicaciones GPS por voz sin necesidad de que el conductor lea la pantalla del dispositivo.

- Lectura en voz alta de textos o libros:
  - *Speak Text Easy:* Permite escuchar un texto, un libro, en formato epub o txt. La aplicación es capaz de hablar varias lenguas en el mismo texto siempre y cuando estén instalados dichos lenguajes sobre el dispositivo.
  - *exPDF Reader:* Lector de ficheros PDF.
  - *PDF to Speech:* Reproduce archivos PDF y archivos de texto (.txt).
  - *Wiki Droyd:* Lector *offline* de la Wikipedia.

- *Moon + Reader Pro:* Lector de *ebooks.*

- TTS incorporado en aplicaciones comunes de los dispositivos:
  - *Announcify:* Lee la identidad del llamante, los mensajes de texto, email, etc.
  - *Talking Caller ID:* Reproduce en voz alta eventos del calendario.
  - *Enhanced SMS & Calles ID:* Proporciona notificaciones de voz para las llamadas entrantes, mensajes SMS/MMS, Google Talk, Gmail, los nuevos mensajes K9/Kaiten/AquaMail y recordatorios de eventos de Google Calendar.
  - *Handcent SMS:* Incorpora TTS a la mensajería del dispositivo. Tiene más funcionalidades que la aplicación de mensajes por defecto y la posibilidad de personalización completa.
  - *DriveCarefully:* Permite leer los mensajes de texto y el email durante la conducción.
  - *Talking Calendar*: Reproduce en voz alta eventos del calendario.
  - *Good Morning:* Reloj de alarma que habla y puede funcionar como asistente personal.

- Lectura en voz alta de notificaciones de redes sociales:
  - *iHear Network:* Permite escuchar en voz alta las notificaciones de Facebook y Twitter.

- Automatización:
  - *ElkDroid:* Automatización del hogar con TTS incorporado.
  - *Tasker:* Automatización total del dispositivo Android con TTS incorporado.

- Herramientas educativas:
  - *Cue Brain:* Permite aprender idiomas de manera interactiva.
  - *Traductor*: Traductor que incorpora más de 50 idiomas.

- Orientadas a personas con discapacidad:
  - *Intersection Explorer:* Ayuda a las personas invidentes a desplazarse por su vecindario ya que dicta en voz alta el mapa de las calles e intersecciones al tocar y desplazar el dedo sobre la pantalla.

2.4.2 Asistentes de voz que integran reconocimiento de voz y síntesis de texto a voz

Hoy en día, los asistentes personales son la principal aplicación de la integración del reconocimiento de voz y la síntesis de texto a voz en aplicaciones para dispositivos

móviles Android. Con el lanzamiento de *Siri*, el asistente personal del iPhone 4S, las aplicaciones de este tipo se popularizaron y empezaron a surgir también para la plataforma Android. A continuación se resumen algunos ejemplos de dichos asistentes desarrollados para dispositivos móviles Android [27].

- **Cloe:** Cloe es el primer asistente personal por voz para dispositivos móviles en español. Está orientado a utilizarse exclusivamente en España ya que el funcionamiento se basa en un sistema de "módulos de aprendizaje", con información preparada de antemano. Algunos de estos módulos son:
  - Traducción de expresiones.
  - Cambio de divisas.
  - Previsión meteorológica.
  - Horarios y tarifas de RENFE.
  - Liga española BBVA y Adelante.
  - Programación TV nacional.
  - Letras de canciones.
  - Sinopsis de películas.
  - Direcciones y teléfonos.
  - Personajes, lugares y definiciones.
  - Reloj mundial.
  - Recetas de cocina.
- **Alicoid** [52]**:** Alicoid es un asistente virtual alemán que también está disponible en español y otros idiomas. Este asistente permite investigar hechos así como mandar mensajes cortos, realizar llamadas, etc. Se maneja a través del lenguaje natural con expresiones como "¿Quién es Barack Obama?", ¿Cómo llego a Sevilla?, "Envía un mensaje a Marcos" o "Abre calendario".
- **Sherpa** [53]**:** Sherpa es un avanzado asistente de voz en español para Android. Este asistente permite realizar una amplia variedad de acciones como consultar información sobre diversos temas, realizar llamadas, abrir aplicaciones, ver resultados deportivos, interactuar con redes sociales, realizar operaciones matemáticas, localizar lugares, consultar el clima, etc. Una de las características más importantes de Sherpa es que, gracias a la tecnología *Intelligent Proactive System Shertab,* es capaz de aprender de los hábitos del usuario para ofrecerle información personalizada y sin la necesidad de que éste la solicite. Además, el módulo informacional *Multiknowledge System* exclusivo de Sherpa, al nutrirse de múltiples fuentes de información, es capaz de entender semánticamente la pregunta y ofrecer resultados más exactos.
- **Vita:** Se trata de uno de los asistentes que mejor funcionamiento presenta en español y uno de los más avanzados para Android. Incluye funciones como llamar por teléfono, enviar SMS o correos electrónicos, mostrar información de contactos, actualizar el perfil en Facebook o Twitter, realizar operaciones matemáticas, consultar la predicción del tiempo, geolocalización, fijar alarmas o recordatorios, búsquedas en Google o Youtube, etc.

- **Skyvi** [54]**:** Skyvi es un asistente de voz para dispositivos Android con funcionalidades muy similares a las mencionadas anteriormente. Su principal ventaja es su gran precisión para entender lo que dice el usuario, la cual es superior al resto de las aplicaciones. La mayor desventaja es que está únicamente disponible en inglés.
- **Assistant (Speaktoit)** [55]**:** Con casi 6 millones de descargas, una puntuación promedio de 4,5 en Google Play [56] y elegida como una de las 10 mejores aplicaciones del 2011 según *The New York Times*, Speaktoit Assistant es uno de los mejores asistentes de voz para dispositivos móviles Android. Este asistente utiliza la tecnología del lenguaje natural para responder preguntas, encontrar información, lanzar aplicaciones y conectar con diferentes servicios web por lo que no es necesario que el usuario utilice comandos predeterminados para su manejo. Es capaz de realizar complejas tareas, como mantener una conversación con el usuario, con una alta comprensión del lenguaje.

  Además, tiene memoria de las acciones y aprende de las preferencias del usuario así como tiene en cuenta el entorno actual y el calendario, lo que le permite realizar sugerencias y tener un comportamiento personalizado.

  Está disponible en diversos idiomas, entre ellos español.

Capítulo 3

Descripción general del sistema

En este capítulo se realiza una descripción general de la aplicación para dispositivos móviles del presente Trabajo de Fin de Grado y de las tecnologías empleadas para su desarrollo.

En primer lugar, se realiza una presentación general del sistema y de su arquitectura modular. Con este objetivo, se realiza una descripción de cada uno de los módulos que la componen en cuanto a funcionalidad, arquitectura y tecnologías utilizadas.

Finalmente, se realiza un análisis de las tecnologías empleadas en el desarrollo de la aplicación así como una descripción general de la implementación de sus funcionalidades.

3.1 Presentación del sistema

La aplicación para dispositivos móviles Android desarrollada en el presente Trabajo de Fin de Grado consiste en una agenda personal avanzada multimodal que permite al usuario almacenar todo tipo de información referente a sus actividades diarias, a modo de recordatorio o diario, de forma organizada y permitiendo un acceso sencillo y rápido a la misma.

Al tratarse de una aplicación multimodal, el sistema permite al usuario el manejo de la aplicación mediante el uso de los canales habituales táctiles (pantalla o teclado) o mediante la voz. Así mismo, las respuestas generadas por la aplicación pueden ser representadas tanto de forma visual como oral.

En la Figura 3.1 se muestra un esquema general del sistema diseñado. Cuando el usuario realiza una petición mediante la voz, el módulo de reconocimiento automático del habla procesa la señal de voz emitida por el usuario y reconoce la información contenida en ésta, convirtiéndola en su representación escrita la cual será utilizada como entrada a la aplicación. De la misma forma, la salida del sistema, en forma de texto, es procesada por el sintetizador de texto a voz (TTS) generando su representación oral.

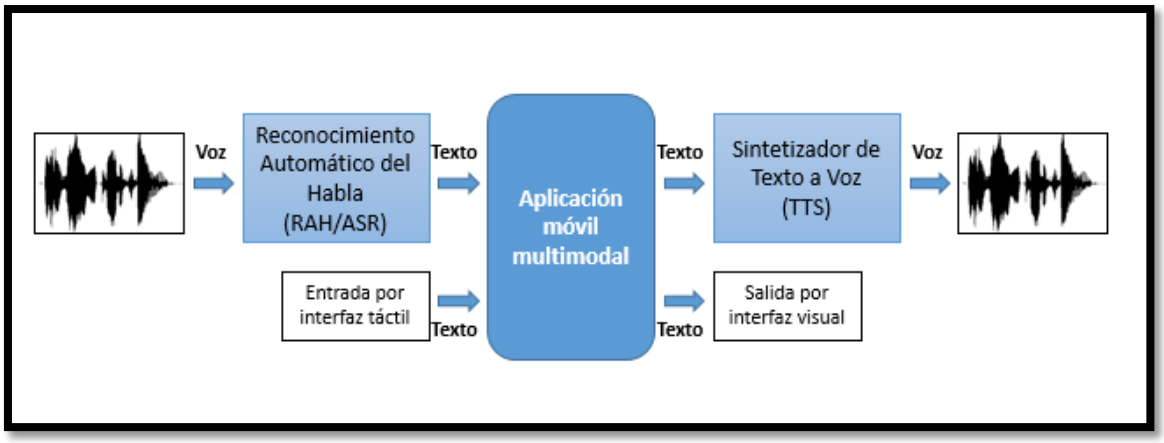


Figura 3.1: Arquitectura del sistema multimodal diseñado

3.1.1 Implementación de los módulos de Reconocimiento Automático del Habla y Síntesis de Texto a Voz

En el Capítulo 2, se ha realizado un estudio de las posibilidades que ofrece Android para el desarrollo de aplicaciones móviles que permitan la interacción oral con el usuario a modo de interfaz. A continuación, se describen las opciones seleccionadas para la implementación de los módulos de Reconocimiento Automático del Habla (RAH/ASR) y de Síntesis de Texto a Voz (TTS) que forman la presente aplicación para dispositivos móviles Android.

3.1.1.1 Módulo de Reconocimiento Automático del Habla (RAH/ASR)

La opción más simple para la integración del reconocimiento de voz en una aplicación Android se basa en el uso de un servicio de reconocimiento de voz ya existente capaz de recibir un *RecognizerIntent,* a través del envío de un objeto de la clase `android.speech.RecognizerIntent` del SDK de Android. El servicio de reconocimiento que se va a utilizar es *Google Voice Search* ya que se trata de uno de los mejores reconocedores de voz para Android, compatible en varios idiomas y que se encuentra instalada por defecto en la mayoría de los dispositivos. Esta aplicación consiste en una pantalla inicial con el texto "Habla ahora", que indica al usuario cuándo el dispositivo se encuentra "escuchando". Por tanto, es necesario que dicha aplicación se encuentre previamente instalada en el dispositivo.

El funcionamiento del servicio de reconocimiento Google Voice Search consiste en, una vez recibido el *RecognizerIntent,* transferir el audio recogido a los servidores de Google para que se realice el reconocimiento. Esto implica que sea necesario disponer de acceso a Internet en el momento en que se esté utilizando la aplicación ya que muchos dispositivos no soportan el modo *offline.* Finalmente, los resultados del

reconocimiento de voz son devueltos a la aplicación en forma de representación escrita para que puedan ser procesados.

3.1.1.2 Módulo de Síntesis de Texto a Voz (TTS)

A partir de la versión 1.6 de Android, se incorpora en la mayoría de los dispositivos un motor de síntesis denominado Pico TTS. Esto permite la integración de la síntesis de texto a voz en cualquier aplicación a través del uso del paquete `android.speech.tts` del SDK de Android. Sin embargo, la plataforma Android ofrece multitud de motores de síntesis alternativos.

La aplicación desarrollada en el presente trabajo, al igual que la mayoría de las aplicaciones existentes en el mercado, utiliza para realizar la síntesis el motor TTS que esté seleccionado como principal en el dispositivo. De esta forma, se asegura que la aplicación va a funcionar correctamente en cualquier dispositivo, sin necesidad de descargar ninguna aplicación adicional.

3.1.2 Implementación de la aplicación multimodal para dispositivos móviles Android

La estructura modular de la aplicación para dispositivos móviles Android desarrollada en el presente Trabajo de Fin de Grado se ilustra en la Figura 3.2. A continuación, se exponen los aspectos más importantes en cuanto a funcionalidad e implementación de cada uno de los módulos que la forman.

Las características de estos módulos serán descritas en mayor detalle en el Capítulo 4.

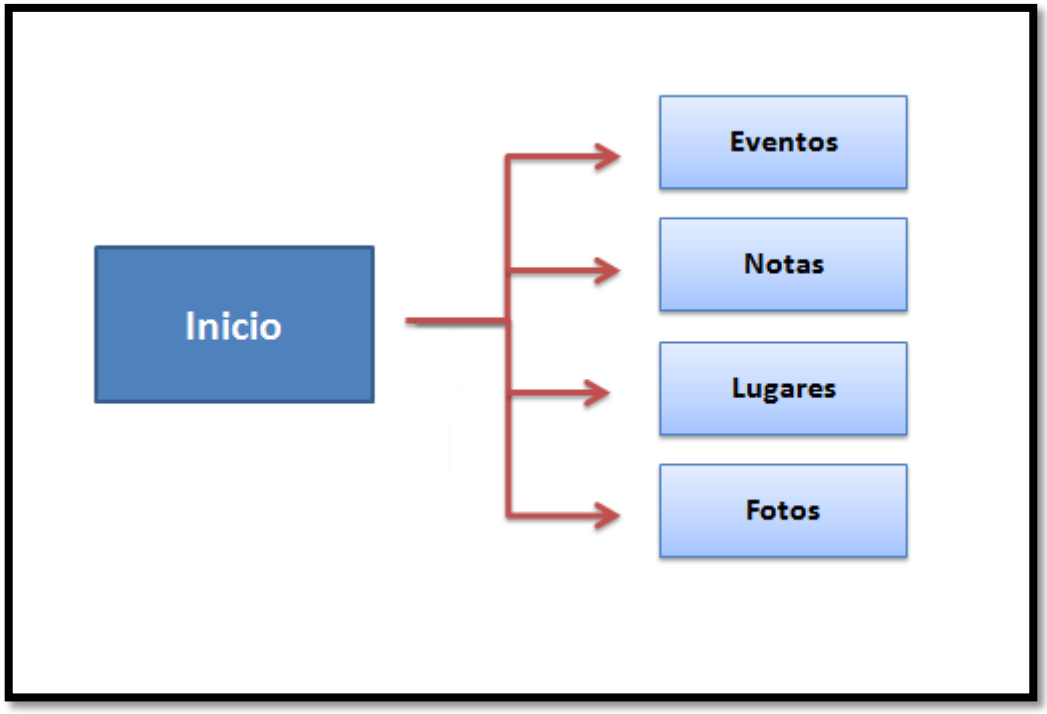


Figura 3.2: Estructura modular de la aplicación desarrollada

Módulo principal o de inicio

Se trata de la pantalla inicial de la aplicación, que muestra toda la información almacenada por el usuario y desde la que se puede acceder a todos los demás módulos que componen la aplicación.

Se trata de una pantalla deslizable con cuatro pestañas que representan cada una de las categorías por las que está organizada la información según la naturaleza de ésta: Eventos, Notas, Lugares y Fotos. Desde esta pantalla se puede consultar toda la información almacenada por el usuario, que aparece listada y ordenada cronológicamente en la pestaña correspondiente. Dicha información está almacenada en una base de datos interna SQLite, que es manejada directamente desde la aplicación a través de una clase auxiliar o clase "adaptadora".

Por otro lado, a través de esta actividad se pueden realizar búsquedas de la información permitiendo al usuario que elija el modo de interfaz que mejor se adapte a sus necesidades o entorno: interfaz táctil o interfaz oral.

- Búsqueda por interfaz táctil: se realiza a través de una barra de búsqueda (*SearchView)* situada en la parte superior de la pantalla. A medida que el usuario escribe la palabra clave que quiere buscar, la aplicación lanza consultas a la base de datos para mostrar aquellos resultados que coinciden con el texto escrito hasta el momento. Esto facilita la búsqueda y la hace más dinámica ya que el usuario puede observar como el descarte de la información se va realizando al mismo momento que teclea. Además, se puede filtrar la

información por una determinada fecha, accediendo a la opción “Filtrar por fecha” situado en el menú de la pantalla principal.

- Búsqueda por interfaz oral: se accede a través de un icono situado, también, en la parte superior de la pantalla, que lanza la aplicación Google Voice Search y que es desde la que se accede a las opciones generales que ofrece la interacción oral. Según el patrón que describa la locución del usuario, el sistema reconoce si se trata de una búsqueda u otra petición. Al igual que ocurre con la interfaz táctil, el usuario puede realizar búsquedas a través de palabras claves simplemente mencionándolas cuando la aplicación Google Voice Search lo indique. Además, la búsqueda por interfaz oral permite consultar las entradas de un día concreto dictando la fecha completa correspondiente, accediendo, como en el caso anterior, a la opción “Filtrar por fecha”.

Módulo de Eventos

Comprende todas aquellas acciones relacionadas con la información asociada a la categoría *Evento*. Este tipo de información está caracterizada por la fecha y hora del evento, la descripción y la ubicación del mismo.

A continuación se presentan las acciones que ofrece este módulo, todas ellas accesibles desde la pantalla principal:

- **Crear:** con esta opción, el usuario puede acceder a la pantalla de creación de un nuevo evento en la cual se rellena la información característica del mismo. Una vez introducida la información correspondiente, el usuario puede indicar a la aplicación que desea guardarla, de manera que se almacena en la base de datos SQLite.
  Esta acción permite ser manejada también por interfaz oral accediendo a Google Voice Search a través de un icono ubicado en la pantalla de creación, de manera que el usuario puede dictar el asunto y ubicación del evento, así como si desea guardar o no el mismo, sin necesidad de tocar la pantalla.
- **Ver:** la acción Ver permite al usuario consultar los datos que caracterizan el evento. Para enriquecer la información aportada por el usuario, en el caso de que la ubicación del evento se trate de una localización existente, se muestra en la parte inferior de la pantalla un fragmento de un mapa indicando la ubicación. Esta funcionalidad se realiza a través del uso del API de Google Maps para Android (*Google Maps Android V2*) [28]
- **Editar:** a través de esta opción se pueden editar los datos de un determinado evento ya almacenado. La pantalla utilizada para esta acción es similar a la de creación vista anteriormente con la diferencia de que los componentes de entrada de los datos muestran por defecto los antiguos datos del evento. También puede ser manejada a través de interfaz oral con el mismo funcionamiento que la pantalla de creación.

- **Eliminar:** permite al usuario eliminar permanentemente de la base de datos un determinado evento.

Módulo de Notas

Comprende todas aquellas acciones relacionadas con la información asociada a la categoría *Notas.* Este tipo de información está definida por una fecha, un texto libre y una categoría a elegir que caracterice el contenido de la nota, a modo de información adicional que ayude a su posterior búsqueda. La aplicación sugiere las siguientes categorías: Libros, Cine, Música, Compras o Ninguna (categoría libre). Este tipo de ítem está diseñado para aquellas ideas, frases, pensamientos o simples recordatorios que el usuario quiera almacenar, emulando un cuaderno de notas.

A continuación se presentan las acciones que ofrece este módulo, todas ellas accesibles desde la pantalla principal:

- **Crear:** con esta opción, el usuario puede acceder a la pantalla de creación de una nueva nota en la cual se rellena la información característica de la misma. Una vez introducida la información correspondiente, el usuario puede indicar a la aplicación si desea guardarla. En caso afirmativo, los datos se almacenan en la base de datos SQLite.
  Esta acción permite ser manejada también por interfaz oral accediendo a Google Voice Search a través de un icono ubicado en la pantalla de creación, de manera que el usuario puede dictar la nota, así como si desea guardar o no la misma, sin necesidad de tocar la pantalla.
- **Ver:** la acción Ver permite al usuario consultar el texto de la nota, así como la fecha y la categoría asociadas a ella. Para enriquecer la información aportada por el usuario, la aplicación añade, bajo el texto anotado, información adicional en el caso de que la categoría elegida se trate de *Cine*, *Música* o *Libros*. En el caso de que la categoría sea *Cine*, se muestra en pantalla una imagen con el cartel de la película e información de la misma (director, guionistas, actores, etc) extraída a través de una petición al servidor IMDb (Internet Movie Database). Si la categoría es *Música*, la pantalla muestra un enlace a la aplicación Youtube con una búsqueda que contenga el texto anotado. Si por el contrario la categoría es *Libros*, se muestra en pantalla un enlace a una búsqueda con el texto anotado en la página web de La Casa del Libro
- **Editar:** a través de esta opción se pueden editar los datos de una determinada nota ya almacenada. La pantalla utilizada para esta acción es similar a la de creación vista anteriormente con la diferencia de que los componentes de entrada de los datos muestran por defecto los antiguos datos de la nota. También puede ser manejada a través de interfaz oral con el mismo funcionamiento que la pantalla de creación.
- **Eliminar:** permite al usuario eliminar permanentemente de la base de datos una determinada nota.

Módulo de Lugares

Comprende todas aquellas acciones relacionadas con la información asociada a la categoría *Lugar*. Este tipo de información está caracterizada por la geolocalización del usuario, la fecha actual y, opcionalmente, una breve descripción del lugar en forma de texto y/o audio. Para obtener la geolocalización actual del dispositivo se ha utilizado el proveedor de localización GPS ya que es uno de los más comunes y el que mejores prestaciones da en términos de precisión. Para representar el mapa y la geolocalización del dispositivo, tanto para las funcionalidades *Crear* como *Ver,* que se explicaran a continuación, se ha requerido del uso del API de Google Maps para Android (*Google Maps Android V2*) [28].

A continuación se presentan las acciones que ofrece este módulo, todas ellas accesibles desde la pantalla principal:

- **Crear:** con esta opción, el usuario puede acceder a la pantalla de creación de una nueva localización en la cual aparece dibujado en un mapa la geolocalización actual del usuario, siempre y cuando esta se encuentre disponible. Si se ha encontrado la geolocalización, la aplicación permite guardarla, dando la posibilidad al usuario de añadir una nota de texto como descripción del lugar antes de ser almacenado. Además, la pantalla de creación permite la grabación de un fichero de audio, que se asociará al lugar una vez se haya guardado.
  Una vez pulsado el botón guardar, la geolocalización actual se almacena en la base de datos SQLite junto con la información asociada.
- **Ver:** la acción Ver permite al usuario consultar los datos que caracterizan el lugar. Esta pantalla consta de una imagen de un mapa con una marca en la geolocalización guardada y la dirección de la misma en la parte superior de la pantalla. Además, se puede acceder a la nota asociada pulsando la marca sobre el lugar, y al fichero de audio a través de un botón en la parte inferior de la pantalla. Esta funcionalidad se realiza a través del uso del API de Google Maps para Android (*Google Maps Android V2*).
- **Editar:** a través de esta opción se puede editar la nota descriptiva de un determinado lugar ya almacenado. Esto se realiza a través de una pequeña pantalla emergente (*Dialog)* la cual permite insertar un nuevo texto y guardarlo, haciendo que se asocie automáticamente el lugar con esta nueva nota.
- **Eliminar:** permite al usuario eliminar permanentemente de la base de datos un determinado lugar.

Módulo de Fotos

Comprende todas aquellas acciones relacionadas con la información asociada a la categoría *Fotos*. Este tipo de información está caracterizada por una imagen realizada

por el dispositivo, la fecha en la que se toma la imagen y, opcionalmente, una descripción de dicha imagen.

A continuación se presentan las acciones que ofrece este módulo, todas ellas accesibles desde la pantalla principal:

- **Crear:** con esta opción, el usuario accede directamente a la aplicación de la cámara del dispositivo. Una vez tomada la fotografía, la aplicación lanza una actividad la cual permite al usuario añadir un texto como descripción de la imagen, así como guardar o descartar la imagen. Una vez pulsado el botón guardar, la imagen se almacena en la base de datos SQLite junto con la información asociada.
  Esta acción permite ser manejada también por interfaz oral accediendo a Google Voice Search a través de un icono ubicado en la pantalla de creación, de manera que el usuario puede dictar la nota sin necesidad de tocar la pantalla.
- **Ver:** la acción Ver permite al usuario acceder a la imagen y a su nota asociada. Esta pantalla consta de una imagen en miniatura de la foto tomada y, bajo esta, la descripción que introdujo el usuario. Además, se puede acceder a la imagen dentro de la galería del dispositivo pulsando sobre dicha imagen.
- **Editar:** a través de esta opción se puede editar la nota descriptiva de una determinada imagen ya almacenado. La pantalla utilizada para esta acción es similar a la de creación vista anteriormente con la diferencia de que los componentes de entrada de los datos muestran por defecto los antiguos datos del evento. También puede ser manejada a través de interfaz oral con el mismo funcionamiento que la pantalla de creación.
- **Eliminar:** permite al usuario eliminar permanentemente de la base de datos una determinada foto.

Cabe destacar que las funcionalidades de creación de los módulos Eventos, Notas, Lugares y Fotos explicados anteriormente pueden ser accedidos desde el módulo principal por interfaz oral a través del icono de la aplicación Google Voice Search situado en la pantalla principal. Para ello se debe hacer uso del comando “Crear” seguido del nombre de la categoría correspondiente (ej. Crear foto).

Además, la aplicación desarrollada está implementada en dos idiomas: español, en el caso de que el idioma del dispositivo esté configurado como español, e inglés en cualquier otro caso. Esto implica que tanto el vocabulario de la aplicación como las locuciones generadas por el sistema y la metodología seguida para el reconocimiento de voz funcionen acorde al idioma seleccionado.

3.2 Tecnologías utilizadas

En esta sección se realiza un análisis de las tecnologías utilizadas en el desarrollo de la aplicación multimodal para dispositivos móviles Android del presente trabajo. Dado

que la aplicación ha sido desarrollada para la plataforma Android, se va a dedicar la primera sección al estudio de dicha plataforma. Además, se describe el proceso seguido para preparar el entorno de desarrollo de la aplicación.

En la siguiente sección, se realiza un estudio del sistema de gestión de base de datos relacionales SQLite y se explica cómo crear y gestionar una base de datos SQLite en Android.

Por último, se estudian las tecnologías utilizadas en las funcionalidades de la aplicación que incluyen mapas y servicios basados en localización. Para ello se realizará un análisis del API de Google Maps para Android (*Google Maps Android V2*), así como de los proveedores y gestores de localización que ofrece la plataforma Android.

3.2.1 Plataforma Android

Al igual que Windows Phone o Apple iOS, la plataforma Android forma parte de una nueva generación de sistemas operativos desarrollados para dispositivos móviles con un hardware más potente. La principal diferencia frente a estos sistemas operativos es que ofrece una plataforma abierta de desarrollo para crear aplicaciones, así como un sistema operativo con núcleo Linux, también de código abierto.

La plataforma Android consta de las siguientes características:

- Un diseño hardware de referencia que describe los requisitos mínimos para soportar la pila software.
- Un núcleo de S.O Linux optimizado para móviles que proporciona un interfaz de bajo nivel de acceso al hardware, gestión de memoria y control de procesos.
- Bibliotecas de código abierto para el desarrollo de aplicaciones como SQLite, WebKit, OpenGl y gestores multimedia.
- Un entorno de ejecución, diseñado para ser pequeño y eficiente, que permite ejecutar aplicaciones Android (máquina virtual Dalvik y biblioteca básica).
- Un entorno de aplicaciones (*application framework*) que proporciona a las aplicaciones acceso a los servicios del sistema: manejador de ventanas, manejador de localización, proveedores de contenido, sensores y telefonía.
- Un interfaz de usuario para lanzar aplicaciones.
- Algunas aplicaciones preinstaladas para facilitar el uso.
- Un SDK (*Software Development Kit)*, con documentación, para crear aplicaciones.

Además, el SDK de Android proporciona:

- Acceso al hardware: a través de sensores como cámara, GPS, acelerómetro, etc.
- Soporte nativo de mapas (aplicación Google Maps) y servicios basados en localización.

- Soporte para construir aplicaciones (*Services*) diseñadas para trabajar en segundo plano, no visibles.
- Almacenamiento y recuperación de datos en una base de datos relacional ligera (SQLite).
- Transmisión de datos entre aplicaciones mediante *Intents* y *Content Providers.*
- Creación de aplicaciones en el escritorio de la pantalla de inicio en forma de *Widgets, Live Folders* o *Live Wallpapers.*
- Soporte de diversos formatos multimedia y de gráficos 2D y 3D.
- Gestión optimizada de memoria y procesos.

3.2.1.1 Arquitectura de la plataforma Android

La Figura 3.3 muestra un gráfico con la estructura en capas de la plataforma Android. Cada una de las capas hace uso de elementos de la capa anterior, es por ello por lo que a este tipo de estructura también se le denomina "pila".

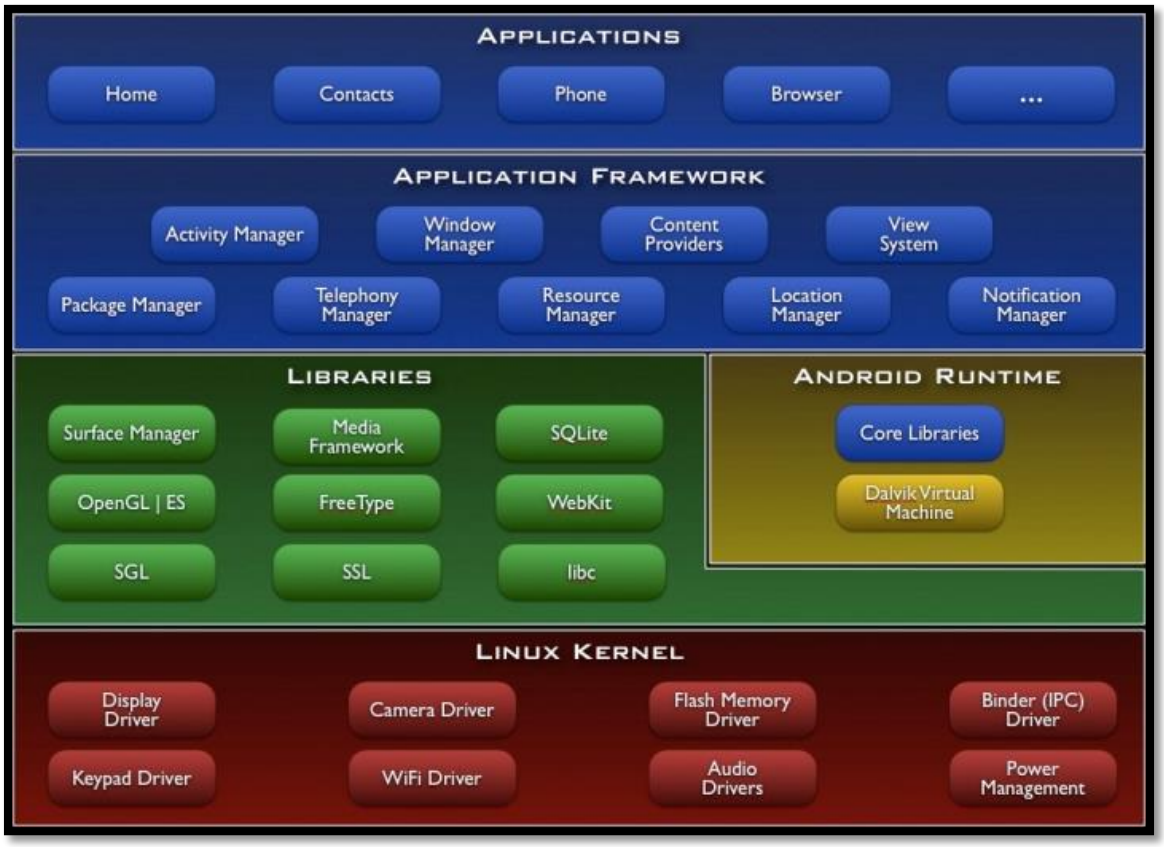


Figura 3.3: Estructura en capas de la plataforma Android

La distribución que se observa en la Figura 3.3 permite acceder a las capas más bajas mediante el uso de librerías, sin necesidad de que el desarrollador programe a bajo nivel para acceder a los componentes hardware del dispositivo.

A continuación, se describen en mayor detalle las características de cada capa [29]:

Kernel de Linux

El núcleo del sistema operativo Android es un kernel Linux versión 2.6 adaptado a las características del hardware en el que se ejecutará (normalmente, un dispositivo móvil). Proporciona una capa de abstracción para los elementos hardware a los que tienen que acceder las aplicaciones, es decir, sirve como conexión entre la parte hardware y software. De esta forma, para cada elemento hardware del dispositivo existe un controlador (o *driver*) dentro del kernel que permite utilizarlo desde el software.

Además, el kernel se encarga de gestionar los diferentes recursos del dispositivo (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación (*networking*), etc.

Librerías

La capa que se sitúa justo sobre el kernel la componen las librerías (*libraries*). Estas librerías están escritas en C o C++ y compiladas para la arquitectura hardware específica del dispositivo y se encuentran instaladas en el terminal antes de ponerlo a la venta. Su cometido es proporcionar funcionalidad a las aplicaciones para tares que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando su eficiencia.

En la tabla 3.1 se presentan algunas de las librerías que se incluyen habitualmente:

Librería	Función
Gestor de superficies (*Surface Manager*)	Se encarga de componer las imágenes que se muestran en la pantalla a partir de capas gráficas 2D y 3D.
SGL (Scalable Graphics Library)	Es el motor gráfico 2D de Android, es decir, se encarga de representar elementos en dos dimensiones.
OpenGL | ES (OpenGL for Embedded Systems)	Motor gráfico 3D basado en las APIs de OpenGL ES 1.0, 1.1 (desde la versión 1.6 de Android) y 2.0 (desde la versión 2.2 de Android)
Bibliotecas multimedia	Permiten visualizar, reproducir y grabar numerosos formatos de imagen, vídeo y audio como JPG, GIF, PNG, MPEG4, AVC, (H.264), MP3, AAC o AMR.
WebKit	Motor web utilizado por el navegador (tanto como aplicación independiente como embebido en otras aplicaciones).
SSL (Secure Sockets Layer)	Proporciona seguridad al acceder a Internet por medio de criptografía.
FreeType	Permite mostrar fuentes tipográficas, tanto basadas en mapas de bits como vectoriales.
SQLite	Motor de bases de datos relacionales.
Biblioteca C de sistema (libc)	Proporciona funcionalidad básica para la ejecución de las aplicaciones.

Tabla 3.1: Librerías que forman parte de la arquitectura Android

Entorno de ejecución

El entorno de ejecución de Android, que se apoya en las librerías mencionadas anteriormente, también está formado por librerías por lo que no se considera una capa en sí misma. En concreto, contiene las librerías esenciales de Android, que incluyen la mayoría de la funcionalidad de las librerías habituales de Java así como otras específicas de Android.

El componente principal del entorno de ejecución de Android es la máquina virtual Dalvik, componente que ejecuta todas y cada una de las aplicaciones no nativas de Android. Las aplicaciones se codifican normalmente en Java y son compiladas en un formato específico para la máquina virtual Dalvik. Esto permite compilar una única vez las aplicaciones y distribuirlas ya compiladas teniendo la garantía de que podrán ser ejecutadas en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera cada aplicación.

Aunque las aplicaciones se escriben en Java, Dalvik no es compatible con el *bytecode* Java que ejecutan las máquinas virtuales Java convencionales. Por tanto, Java se usa únicamente como lenguaje de programación. Durante el proceso de compilación de los programas Java (normalmente archivos .java) se genera, de forma intermedia, el bytecode habitual (archivos .class) los cuales son convertidos al formato específico de Dalvik en el proceso final (.dex, de *Dalvik executable*). La ventaja reside en que los archivos .dex son mucho más compactos que los .class equivalentes (hasta un 50% menos de tamaño), lo que permite ahorrar en espacio en memoria y acelerar el proceso de carga. Además, a diferencia de las máquinas virtuales tradicionales, Dalvik se basa en registros en lugar de una pila para almacenar los datos, lo que requiere menos instrucciones. Esto permite ejecuciones más rápidas en un entorno con menos recursos.

Las aplicaciones Android se ejecutan cada una en su propia instancia de la máquina virtual Dalvik, evitando así interferencias entre ellas, y tienen acceso a todas las librerias mencionadas anteriormente y, a través de ellas, al hardware y al resto de recursos gestionados por el kernel.

Marco de aplicación

Esta capa está formada por todas las clases y servicios que utilizan directamente las aplicaciones para realizar sus funciones. La mayoría de los componentes de esta capa son librerías Java que acceden a los recursos a través de la máquina virtual Dalvik.

En la Tabla 3.2 se resumen algunas de las librerías más importantes que forman el marco de aplicación:

Librería	Función
Administrador de actividades (*Activity Manager*)	Se encarga de controlar el ciclo de vida de las actividades y la pila de actividades.
Administrador de ventanas (*Windows Manager*)	Se encarga de organizar lo que se muestra en pantalla, creando superficies que puedan ser *rellenadas* por las

	actividades.
Proveedor de contenidos (*Content Provider*)	Permite encapsular un conjunto de datos que va a ser compartido entre aplicaciones.
Vistas (*Views*)	Son las que construyen las interfaces de usuario: botones, cuadros de texto, listas, navegadores web, etc.
Administrados de notificaciones (*Notification Manager*)	Proporciona servicios para notificar al usuario cuando algo requiera su atención.
Administrador de paquetes (*Package Manager*)	Permite obtener información sobre los paquetes actualmente instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes.
Administrador de telefonía (*Telephony Manager*)	Proporciona acceso a la pila hardware de telefonía del dispositivo Android.
Administrador de recursos (*Resource Manager*)	Proporciona acceso a todos los elementos propios de una aplicación que se incluyen directamente en el código: cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos, disposiciones de las vistas dentro de una actividad (*layouts*), etc.
Administrador de ubicaciones (*Location Manager*)	Permite determinar la posición geográfica del dispositivo Android (utilizando el GPS o las redes disponibles) y trabajar con mapas.
Administrador de sensores (*Sensor Manager)*	Permite gestionar todos los sensores hardware disponibles en el dispositivo Android: acelerómetro, sensor de luminosidad, sensor de campo magnético, brújula, sensor de temperatura, etc.
Cámara	Proporciona acceso a las cámaras del dispositivo Android, tanto para tomar fotografías como para grabar vídeo.
Multimedia	Conjunto de bibliotecas que permiten reproducir y visualizar audio, vídeo e imágenes en el dispositivo.

Tabla 3.2: Librerías que forma el Marco de Aplicación

Aplicaciones

La capa superior de esta pila software está formada por todas las aplicaciones: tanto las que tienen interfaz de usuario como las que no, tanto las nativas (programadas en C o C++) como las administrativas (programadas en Java), las que vienen por defecto en el dispositivo o las instaladas por el usuario, etc.

El hecho de que todas las aplicaciones utilicen el mismo marco de aplicación para acceder a los servicios que proporciona el sistema operativo implica dos ventajas: se pueden crear aplicaciones que usen los mismos recursos que utilizan las aplicaciones

nativas y se posibilita reemplazar cualquiera de las aplicaciones del teléfono por otra si el usuario lo desea.

3.2.1.2 Componentes de una aplicación en Android

Las aplicaciones Android están basadas en cuatro componentes: *Activities, Services, Broadcast receivers* y *Content providers.* Toda aplicación Android es siempre una combinación de uno o más de estos componentes los cuales deben estar declarados de forma explícita en el fichero AndroidManifest.xml de la aplicación, junto a otros datos asociados como valores globales, clases que implementa, datos que puede manejar, permisos, etc.

En las siguientes secciones se resumen los componentes que forman una aplicación para dispositivos móviles Android.

Activity

Son las que modelan una actividad llevada a cabo por una aplicación y que llevan asociadas, normalmente, una ventana o elemento de interfaz gráfica. Una aplicación puede tener una o varias actividades, existiendo una principal que es la que se presenta al usuario cuando se arranca la aplicación. Este componente se implementa mediante la clase **Activity.**

El ciclo de vida de una actividad se compone de tres estados:

- **Activa o en ejecución:** la actividad se encuentra en primer plano y tiene el foco de interacción con el usuario.
- **Pausada:** la actividad ha perdido el foco de interacción pero sigue siendo visible para el usuario. En este caso, se considera una actividad "viva" pero puede ser eliminada por el sistema en cualquier momento, normalmente, por situaciones extremas de baja memoria.
- **Parada:** la actividad se encuentra totalmente oculta por otra. Se mantiene la información de su estado por si es necesario recuperarla pero son frecuentemente eliminadas por el sistema para liberar memoria.

En la Figura 3.4 se muestra un esquema del ciclo de vida de una actividad en el que se muestran los métodos que son invocados cada vez que una *Activity* cambia de estado.

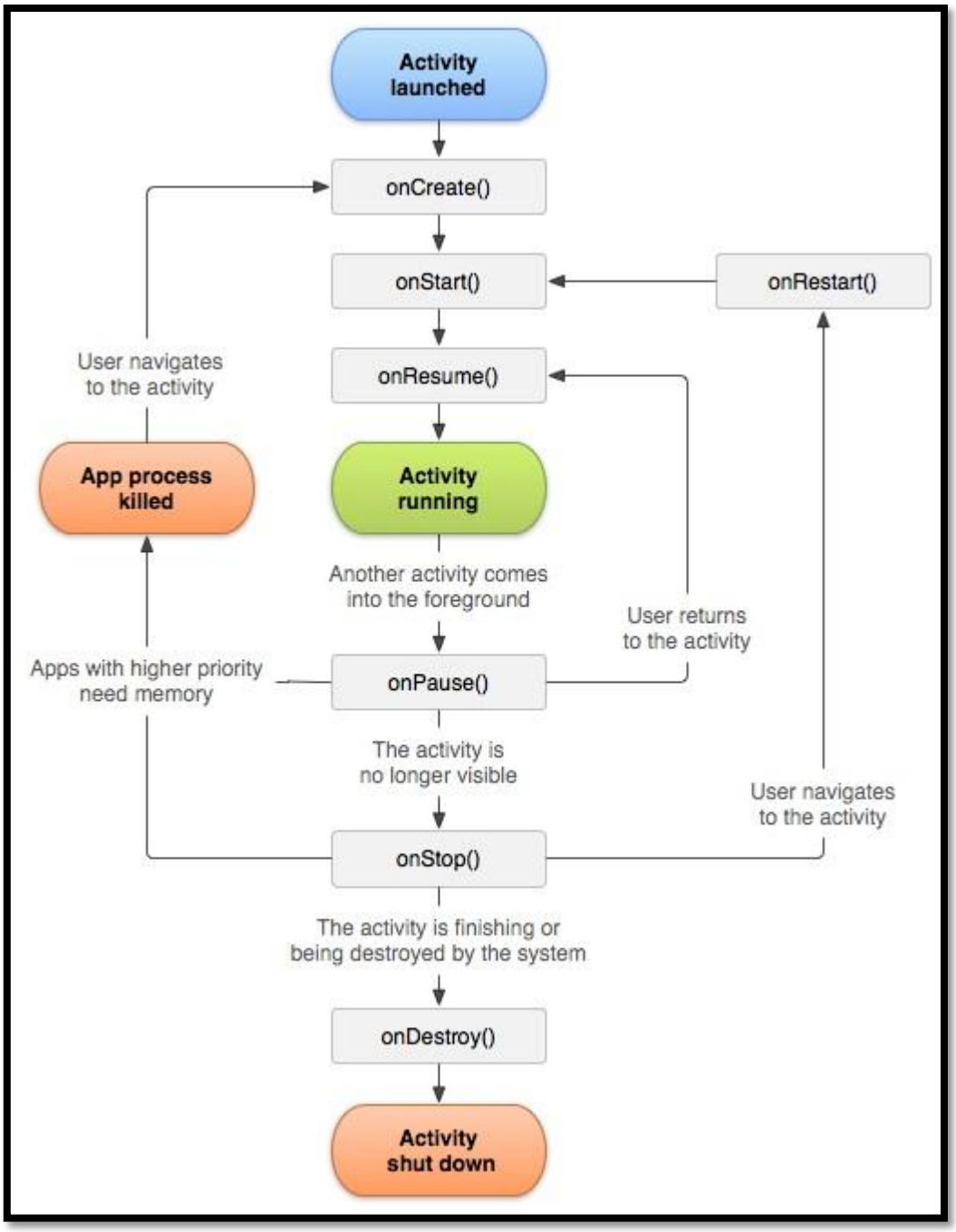


Figura 3.4: Ciclo de vida de una actividad

Los métodos que definen el ciclo de vida de una actividad son los siguientes:

- **onCreate(), onDestroy()**: abarcan todo el ciclo de vida. Representan el principio y fin de la actividad.
- **onStart(), onStop()**: representan la parte visible para el usuario. Entre la ejecución de estos dos métodos, la actividad será visible para el usuario aunque es posible que no tenga el foco de acción por existir otras actividades superpuestas con las que el usuario está interactuando. Estos métodos pueden ser invocados múltiples veces.

- **onResume(), onPause()**: abarcan la parte útil del ciclo de vida. Durante este periodo, la actividad es visible para el usuario y además tiene el foco de la acción.
- **onRestart()**: despierta una actividad que está en segundo plano o invisible.

Service

El componente *Service* representa una actividad ejecutada en segundo plano sin interfaz gráfico. Los Services pueden ser útiles tanto para casos en los que el usuario no necesita interactuar con la actividad como para suministrar funcionalidad a otras aplicaciones. Este componente se implementa como subclase de la clase *Service.*

Broadcast receiver

Este tipo de componente no realiza ninguna acción si no que está diseñado para recibir y reaccionar ante anuncios enviados por difusión. Dichos anuncios son en su mayoría generados por el propio sistema (cambios en la zona horaria, batería baja, etc.) aunque también pueden ser generados por las aplicaciones. Por lo general, los Broadcast Receiver reaccionan a estos anuncios lanzando alguna actividad que procese la información recibida. Este componente se implementa extendiendo de la clase *BroadcastReceiver.*

Content provider

Este componente modela un conjunto de datos de una aplicación que pueden estar almacenados en el sistema de ficheros, en una base de datos o cualquier otro tipo de almacenamiento. Para acceder a dichos datos, se debe hacer uso de un objeto *ContentResolver,* el cual sirve de comunicación entre la aplicación y el proveedor de contenidos como una relación cliente-servidor. Este componente extiende de la clase *ContentProvider.*

Cualquiera de estos cuatro componentes debe ser activado explícitamente durante el proceso de ejecución de la aplicación cuando este se requiera. Para ello, se hace uso de los llamados **ContentResolver** e **Intents**, dependiendo de qué tipo de componente se trate. El primero de ellos, como se ha visto anteriormente, se encargan de activar los componentes de tipo Content Provider. Por otro lado, un Intent es una descripción abstracta de una operación a realizar y, a través de él, se puede activar los componentes de tipo Activity, Service o Broadcast Receiver.

3.2.1.3 Preparación del entorno de desarrollo para aplicaciones Android en Windows.

Los pasos seguidos para la preparación del entorno de desarrollo para aplicaciones Android, que se describirán en mayor detalle a continuación, son los siguientes:

1. Descarga e instalación del ADT (*Android Developer Tools*) Bundle, que incluye:

- Eclipse (herramienta de desarrollo de software) como IDE, con los plug-in ADT necesario para el desarrollo de aplicaciones Android.
- SDK (*Software Development Kit*): de Android: kit de desarrollo de aplicaciones Android.

2. Descarga de plataformas Android de desarrollo utilizando el SDK Manager.

Cabe destacar que, antes de realizar el primer paso, hay que asegurarse que se encuentra instalado en el ordenador JDK 6 o más. Para aplicaciones que serán ejecutadas en la versión 5 (o más) de Android se requiere tener instalado JDK 7.

Descarga e instalación del ADT Bundle para Windows

La página web de descarga del SDK de Android [57] proporciona todas las herramientas necesarias para el desarrollo de una aplicación para dispositivos móviles Android, incluyendo Eclipse IDE y las herramientas SDK para Android. Una vez descargado el paquete .zip desde el enlace indicado en dicha página, se procede a la instalación siguiendo los pasos que se describen a continuación:

1. Se ejecuta el fichero .exe.
2. Siguiendo el asistente de configuración, se instala Eclipse y todas las herramientas SDK necesarias. En algunos sistemas Windows, es necesario indicar la localización de Java en el equipo como una variable de entorno. Para ello, se selecciona Inicio>Equipo>Propiedades de Sistema>Propiedades de Sistema Avanzadas. Entonces, se abre la pestaña Avanzadas > Variable de Entorno y se añade una nueva variable de JAVA_HOME que apunte a la carpeta donde se encuentra el JDK

Descarga de plataformas Android de desarrollo utilizando el SDK Manager

Una vez que está instalada en el equipo la herramienta de desarrollo de software (Eclipse), se procede a la descarga de aquellas herramientas, plataformas y otros componentes del SDK de Android que no vienen por defecto. Esto se realiza a través de la herramienta Android SDK Manager, que se encuentra dentro de la aplicación Eclipse, cuya pantalla de manejo se muestra en la Figura 3.5.

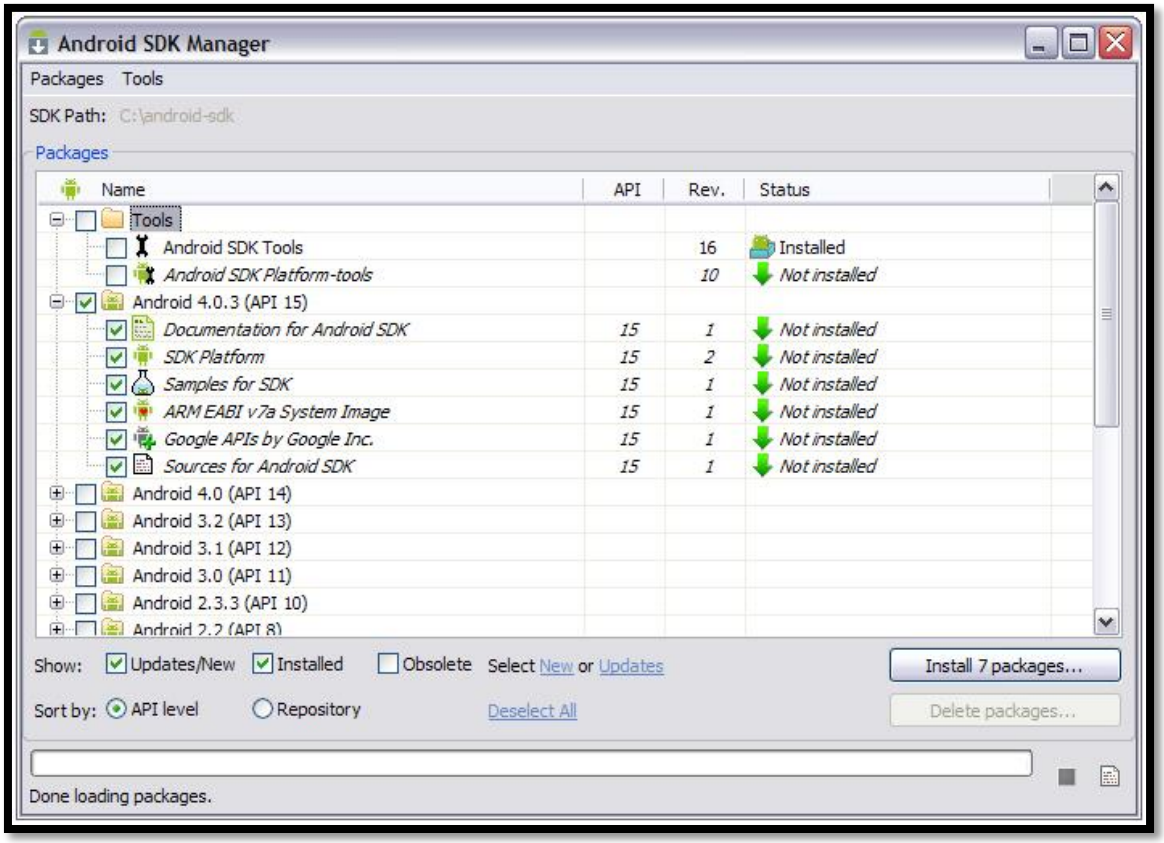


Figura 3.5: Herramienta Android SDK Manager

La primera vez que se abre dicha herramienta, algunos paquetes están seleccionados por defecto. Además, es importante que se añadan las últimas versiones de las herramientas del SDK, las librerías de soporte para APIs adicionales (Android Support Repository y Android Support Library) los servicios de Google (Google Repository y Google Play services) para la utilización de sus APIs.

3.2.1.4 Desarrollo de una aplicación Android en Eclipse

Las herramientas del SDK en Eclipse permiten crear fácilmente un nuevo proyecto Android con un conjunto de ficheros y carpetas por defecto. En primer lugar, se debe seleccionar *File -> New -> Project -> Android Application Project.*

A continuación, se pide rellenar un formulario, el cual se muestra en la Figura 3.6, con la siguiente información:

- **Nombre de la aplicación (*Application name):*** Nombre de la aplicación, que será el mostrado a los usuarios. En el caso de este trabajo, el nombre es *DayByDay.*
- **Nombre del proyecto (*Project Name):*** Nombre del directorio que se creará para el proyecto y el nombre con el que se le hará referencia en Eclipse.

- **Nombre del paquete (*Package Name*):** Nombre del paquete a utilizar para la aplicación. El nombre del paquete debe ser único. Para este trabajo, se ha utilizado el nombre *com.example.daybyday.*
- **SDK mínimo requerido (*Minimum Required SDK):*** Versión mínima del API de Android que soporta la aplicación que se va a desarrollar. Para garantizar la compatibilidad con el mayor número de dispositivos posibles, es aconsejable establecer este valor a la mínima versión que permita implementar las características principales de la aplicación. En la presente aplicación se ha utilizado el nivel 10 del API, de forma que funcione a partir de la versión 2.3.3 de la plataforma Android.
- **SDK objetivo (*Target SDK*):** es la versión más moderna de Android en la que la aplicación puede funcionar, por lo que en el presente trabajo se ha elegido la versión 20 del API.
- **Compilar con (*Compile With):*** Es la versión de la plataforma con la que se quiere compilar la aplicación. En este caso, se ha elegido la versión más reciente permitida (API 20).
- **Theme (*Tema*):** Especifica el estilo del interfaz visual para la aplicación. En la presente aplicación se ha utilizado el tema *Holo Light.*

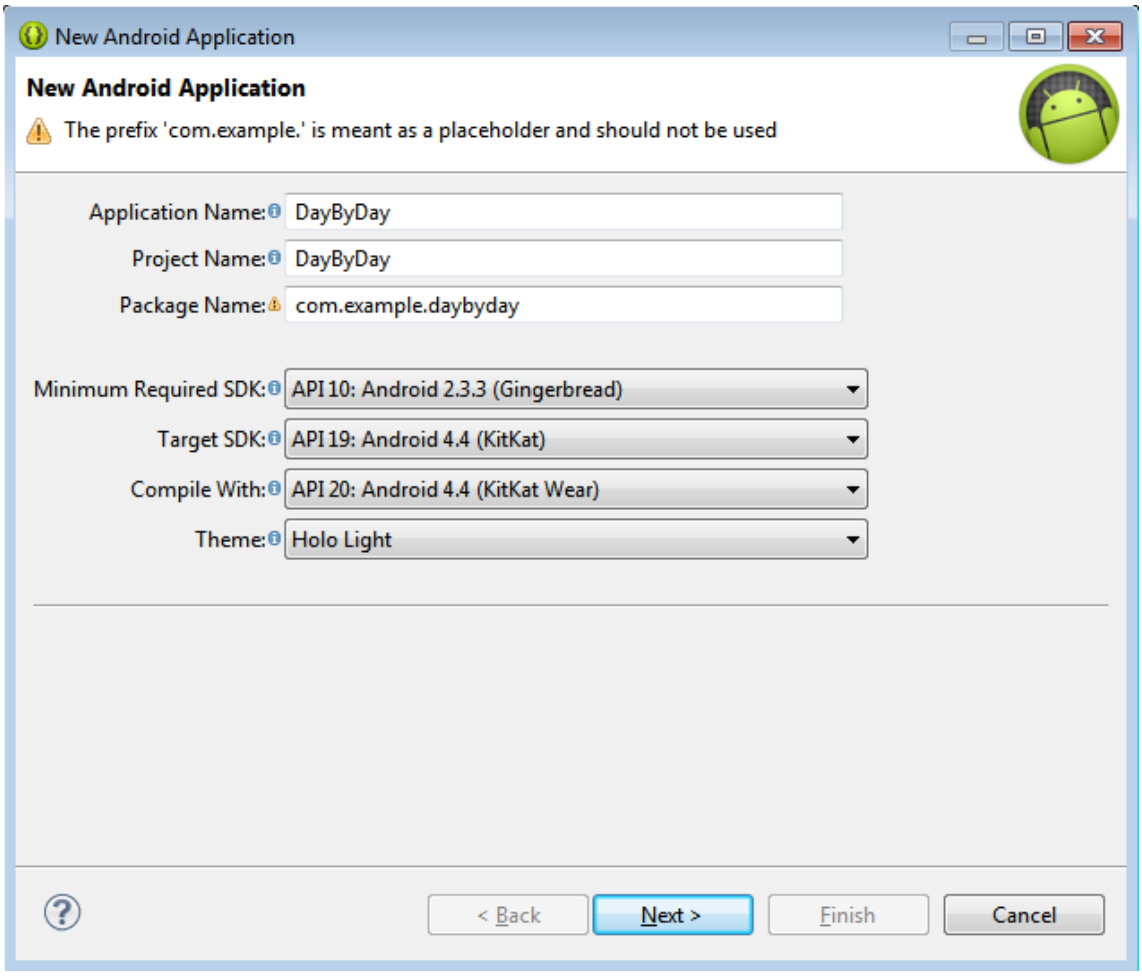


Figura 3.6: Formulario para la creación de un proyecto Android en Eclipse

Una vez se han rellenado estos campos, se deja el resto a los valores por defecto y se pulsa *Next.* En las siguientes pantallas permiten, de forma opcional, crear un icono para la aplicación personalizado, así como la creación de la primera actividad utilizando una plantilla. En el caso de que se quieran dejar estas acciones para más adelante, es el momento de pulsar *Finish* y el nuevo proyecto se habrá creado.

Estructura de un proyecto Android en Eclipse

Una vez se ha generado el nuevo proyecto Android, se generan automáticamente una serie de carpetas. La Figura 3.7 muestra la estructura típica de un proyecto Android en Eclipse.

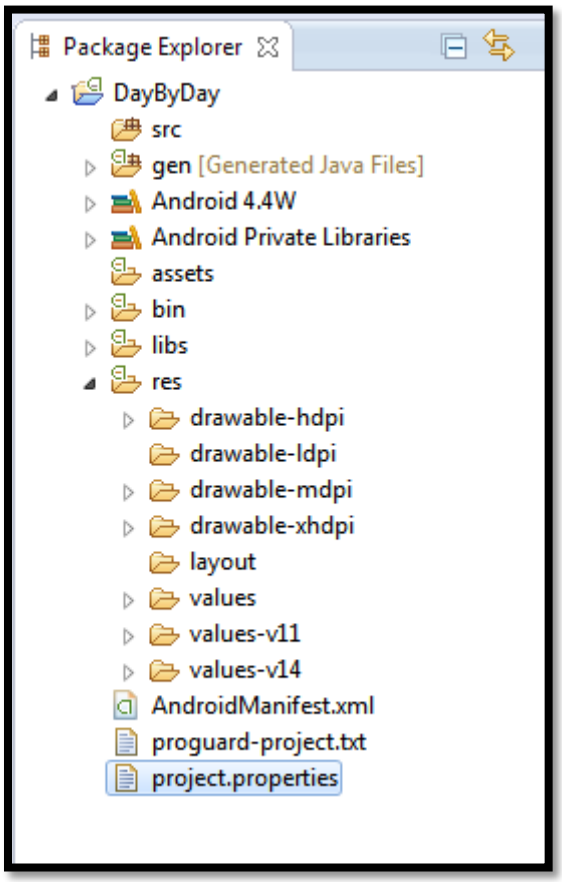


Figura 3.7: Estructura de un proyecto Android en Eclipse

A continuación se describe la funcionalidad de las carpetas y ficheros generados:

- Carpeta *src*: Almacena todas las clases de la aplicación (ficheros .java).
- Carpeta *res*: En ella se guardan todos los recursos que necesita la aplicación (imágenes, ficheros .xml de interfaz, ficheros .xml de *strings,* sonidos, audios, etc.).
- Carpeta *gen*: en ella se encuentran las clases generadas automáticamente por Android, entre ellas la clase R.java en la que se almacena, en forma de atributo, una dirección asociada a cada recurso de la carpeta *res* de forma que se puede acceder a los recurso de la aplicación desde cualquier clase de la forma R.subcarpeta_res.nombre_recurso.

- Fichero AndroidManifest.xml: Es un fichero que debe existir en toda aplicación Android en el que se indican cada uno de los componentes de la aplicación y relaciones entre ellos, librerías necesarias y permisos que necesita la aplicación para su ejecución.

Ejecución de una aplicación Android en Eclipse

A continuación se resumen las dos alternativas existentes para la ejecución de una aplicación Android en Eclipse:

- Ejecución de la aplicación en un AVD (*Android Virtual Device*):
  Consiste en la configuración de un dispositivo virtual en el que se puede emular la aplicación. Para crear un AVD en Android se han seguido los siguientes pasos:

  Para crear el AVD:
  1. Se ejecuta el Gestor de dispositivos virtuales de Android a través del botón de la barra de herramientas Android Virtual Device Manager.
  2. Se crea un nuevo AVD seleccionando la opción *Create.* En el formulario que aparece se rellenan los detalles del dispositivo: nombre, plataforma objetivo, tamaño de la tarjeta SD y una piel (por defecto es HVGA). Al finalizar, se pulsa *OK.*
  3. El nuevo AVD creado aparece entonces en el Gestor de dispositivos virtuales de Android. Para iniciar dicho dispositivo sólo hay que ser seleccionado y lanzado (opción *Start*).
  4. Una vez haya terminado de arrancar el emulador, se puede desbloquear y manejar igual que un teléfono móvil corriente.

  Para ejecutar la aplicación:

  1. Se abre uno de los archivos del proyecto y se pulsa sobre el botón *Run* de la barra de herramientas.
  2. En la ventana *Run as* que aparece a continuación, se selecciona la opción Android Application y se pulsa *OK* ordenando a Eclipse que instale la aplicación en el AVD.

- Ejecución de la aplicación en un dispositivo real:

  En el presente Trabajo de Fin de Grado, esta ha sido, la metodología aplicada para la ejecución de la aplicación desarrollada ya que se puede comprobar en su totalidad el resultado final de la implementación de una aplicación. A continuación se describen los pasos seguidos:

  1. Se conecta el dispositivo al ordenador mediante un cable USB. En ocasiones, es necesario instalar el driver USB apropiado para el dispositivo.
  2. Se habilita la opción Depuración USB que se encuentra en el menú Ajustes -> Aplicaciones -> Desarrollo.

Para ejecutar la aplicación se siguen los mismos pasos que se han explicado anteriormente. En este caso, Eclipse instala la aplicación en el dispositivo conectado.

Depuración de una aplicación Android en Eclipse

La depuración de la aplicación se realiza a través de la opción *Run -> Debug.* Previamente, se debe colocar un punto de ruptura en la línea de código en la que se desea parar la ejecución haciendo doble clic junto a esta en la barra de la izquierda. Una vez lanzada la depuración, la aplicación se reinicia, quedándose suspendida cuando alcanza el punto de interrupción. Con la tecla F6 del teclado se avanza en el código línea a línea y pulsando la tecla F8 se continúa hasta el siguiente punto de interrupción.

3.2.2 Base de datos SQLite

3.2.2.1 Introducción a las Bases de Datos SQLite

SQLite es un sistema de gestión de bases de datos relacionales de código abierto, ligero, conforme a estándares y con un solo salto [30]. Está implementada como una librería C dentro de la pila software lo que implica que cada base de datos forma parte de la aplicación que la crea, es decir, sólo la aplicación que la crea tiene acceso a su base de datos. Por otro lado, al proporcionar funcionalidad a través de una librería y no como un proceso separado, se reducen las dependencias externas, minimiza latencia y simplifica la transacción de bloqueo y sincronización. Estas características, junto a su compacto tamaño (inferior a 500KiB), la convierte es una base de datos muy apropiada para usarla en dispositivos con poca memoria como teléfonos móviles, PDAs o reproductores MP3. Otra importante ventaja de las bases de datos SQLite frente a otros motores de bases de datos convencionales es que es débilmente tipado, es decir, los valores de las columnas no tienen que ser del mismo tipo.

Tal y como se ha descrito en la Sección 3.1.2, se ha utilizado una base de datos SQLite en la aplicación desarrollada para almacenar toda la información introducida por el usuario. Esta base de datos está formada por cuatro tablas, una para cada tipo de ítem a guardar: tabla de Eventos, tabla de Notas, tabla de Lugares y tabla de Fotos.

3.2.2.2 Implementación de una base de datos SQLite en el desarrollo de una aplicación para dispositivos móviles

A continuación, se explican los conceptos básicos necesarios a la hora de trabajar con una base de datos SQLite en el desarrollo de una aplicación para dispositivos móviles Android, los cuales se han seguido en la implementación del presente trabajo. [30]

***ContentValues* y la clase Cursor**

En Android, para insertar filas en una tabla se utiliza un objeto *ContentValue*. Cada objeto de este tipo representa una fila de una tabla como un mapeo entre nombre de columna y valor.

Por otro lado, las consultas en Android devuelven objetos de tipo *Cursor*. En lugar de extraer y devolver una copia de los valores consultados, los cursores actúan como punteros a un subconjunto de datos subyacentes. Es decir, los cursores son una forma de controlar la posición (fila) en el conjunto de resultados de una consulta a la base de datos. A continuación, se listan algunos de los métodos más importantes que ofrece la clase cursor para navegar por la lista de resultados:

- `moveToFirst`: Desplaza el cursor a la primera fila de la lista de resultados.
- `moveToNext`: Desplaza el cursor a la siguiente fila.
- `moveToPrevious`: Desplaza el cursor a la fila previa.
- `getCount`: Devuelve el número de filas de la lista de resultados.
- `getColumnIndexOrThrow`: Devuelve un índice de una columna con el nombre especificado como parámetro. Lanza una excepción si no existe una columna con dicho nombre.
- `getColumnName`: Devuelve el nombre de la columna especificada por su índice.
- `getColumnNames`: Devuelve un array de objetos de tipo *String* de todas las columnas contenidas el resultado.
- `moveToPosition`: Mueve el cursor a la fila especificada.
- `getPosition`: Devuelve la posición actual del cursor.

Cabe destacar que, una vez que se han realizado las funciones necesarias con el cursor, es importante cerrar dicho objeto haciendo uso del método `close`() con el fin de liberar sus recursos.

Acceso a la base de datos: clase adaptadora y *SQLiteOpenHelper*

Se considera una buena práctica crear una clase auxiliar para simplificar las interacciones con las bases de datos creadas para una aplicación. Es por ello por lo que se ha utilizado este método en la creación de la base de datos de la aplicación desarrollada en el presente trabajo.

Esta clase añade una capa de abstracción que encapsula las interacciones con las bases de datos. Además, proporciona métodos intuitivos y fuertemente tipados para crear, abrir y cerrar la base de datos, insertar, actualizar o borrar entradas y manejar

consultas. Es aconsejable, además, definir en ella las constantes estáticas de la base de datos como el nombre de la tabla y las columnas.

Es usual integrar en ella la clase auxiliar `SQLiteOpenHelper.` En Android, es típico el uso de una clase personalizada que derive de esta para crear, actualizar y conectar con la base de datos. A continuación, se describen los métodos más importantes de esta clase [16]:

- onCreate(): Es llamado cuando la base de datos se crea por primera vez. Debe ser personalizado por el desarrollador con el código necesario para crear la base de datos.
- onUpgrade(): Es llamado cuando la base de datos necesita ser actualizada. Debe ser utilizado al eliminar o añadir tablas de forma que se actualice su estructura.
- getWritableDatabase(): Crea y/o abre una base de datos que será usada tanto para leer como para escribir. Puede ser llamada cada vez que se necesite escribir en la base de datos.
- getReadableDataBase(): Tiene el mismo funcionamiento que el método getWritableDatabase(): a menos que haya algún problema, normalmente relacionado con falta de memoria en disco. En tal caso, la base de datos se abre en forma de sólo lectura.
- `close():` Se encarga de cerrar la base de datos. Es importante llamar a este método cuando la base de datos no se vaya utilizar más con el objetivo de liberar recursos.

Consulta a la base de datos

Las consultas a la base de datos se realizan utilizando el método `query(boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)`, sobre el objeto de la base de datos. Los parámetros de entrada de dicho método se describen a continuación:

- *distinct:* Un objeto de tipo *boolean* opcional que especifica si el resultado debe incluir sólo valores únicos.
- *table:* Nombre de la tabla a consultar.
- *columns:* Array de Strings con la lista de columnas a incluir en el resultado.
- *selection:* Cláusula "where" que define las filas a ser devueltas. Puede incluir '?'.
- *selectionArgs:* Array de argumentos de selección almacenados como strings, que sustituyen el '?' de la cláusula "where".
- *groupBy:* Cláusula "group by" que definen cómo deben estar agrupadas las filas del resultado.
- *having:* Claúsula "having" que define qué grupos de columnas se deben incluir si se ha especifica la cláusula "group by".
- *orderBy:* objeto de tipo String que indica el orden de las filas devueltas.
- *limit:* objeto de tipo String opcional indicando el límite del número de filas devueltas.

Como se ha visto anteriormente, la consulta devuelve un objeto de tipo Cursor. Para extraer los resultados del objeto Cursor, primero se ha de mover el cursor a la posición desde la que queramos empezar a extraer utilizando el método `moveTo<First/Next/Previous/Position>`. A continuación, se hace uso del método `get<Type>` para obtener el valor de una columna al tipo que deseemos. Esto es posible ya que SQLite es débilmente tipado, es decir, un valor se puede leer en un tipo distinto del que fue almacenado, se convierte automáticamente. En la Figura 3.8, se muestra un ejemplo en el que se extrae un objeto de tipo String de un objeto Cursor.

```
String nota = "";

Cursor cursor = db.query(DATABASE_TABLE_NOTE, new String[] { KEY_NOTE }, KEY_ID + "=?",
        new String[] { String.valueOf(id) }, null, null, null, null);
if (cursor != null)
    cursor.moveToFirst();
nota = new String(cursor.getString(0));
cursor.close();
```

Figura 3.8: Ejemplo de extracción de resultados de un Cursor.

Como se muestra en la imagen, en el momento en que se deja de utilizar el objeto Cursor es importante cerrarlo a través del método close() con el objetivo de liberar recursos.

Insertar, actualizar y borrar filas de la BD

Se puede hacer uso del método `execSQL(String sql)` [16] de la clase `SQLiteDatabase` para ejecutar cualquier sentencia sql manualmente. No obstante, esta clase exporta los métodos `insert`, `update` y `delete` que encapsulan las sentencias SQL para hacer dichas operaciones de una forma más sencilla. A continuación, se describen los pasos a seguir para insertar, actualizar y borrar filas de una base de datos utilizando los métodos mencionado.

Insertar filas

Como hemos visto anteriormente, para insertar datos en una base de datos se hace uso de un objeto `ContentValue`. Por tanto, el primer paso es construir un objeto de este tipo para crear la fila. A continuación, se utiliza el método `put (String key, <Type> value)` de esta clase para dar un valor a cada columna. Por último, se llama al método `insert (String table, String nullColumnHack, ContentValues values)` para insertar la fila en la base de datos. La Figura 3.9. muestra un código de ejemplo en el que se implementa esta acción.

```
ContentValues values = new ContentValues();
values.put(KEY_DATE, date);
values.put(KEY_FECHA, fecha);
values.put(KEY_NOTE, note);
values.put(KEY_CATEGORIA, categoria);

db.insert(DATABASE_TABLE_NOTE, null, values);
```

Figura 3.9: ejemplo de inserción de una fila en base de datos

Actualizar filas

Al igual que el caso anterior, se realiza a través de un objeto ContentValues. A través del método `put (String key, <Type> value)` de esta clase se dan los nuevos valores a las columnas que se quieren modificar. Finalmente, se realiza una llamada al método `update (String table, ContentValues values, String whereClause, String[] whereArgs)` indicando en la cláusula *where* las filas en las que se debe aplicar el cambio. En la Figura 3.10 se muestra un ejemplo de código en la que se actualiza la fila que coincida con un identificador (*id*) dado.

```
ContentValues values = new ContentValues();
values.put(KEY_NOTE, texto);
values.put(KEY_CATEGORIA, categoria);
db.update(DATABASE_TABLE_NOTE, values, KEY_ID + "=?", new String[] { String.valueOf(id) });
```

Figura 3.10: Ejemplo de actualización de una fila en base de datos

Borrar filas

Para eliminar una o varias filas de la base de datos, simplemente se hace una llamada al método `delete(String table, String whereClause, String[] whereArgs)` sobre el objeto de tipo `SQLiteDatabase` indicando el nombre de la tabla y la cláusula *where* de las filas que se desean borrar. En la Figura 3.11, se muestra une ejemplo de código en la que se elimina la fila que coincida con un identificador (*id*) dado.

```
db.delete(DATABASE_TABLE_NOTE, KEY_ID + " =?", new String[] {String.valueOf(id)});
```

Figura 3.11: Ejemplo de eliminación de una fila en base de datos.

3.2.3 Google Maps Android API v2

En diciembre de 2012, Google presentó la segunda versión de su API de Google Maps para Android con importantes mejoras con respecto a la anterior. Esta API proporciona un servicio de cartografía que puede ser utilizado en el desarrollo de aplicaciones Android, de forma que se pueda utilizar un mapa como interfaz gráfico.

En esta sección, se describirán los pasos previos necesarios para hacer uso de este API en nuestra aplicación, así como las funcionalidades que ofrece dicha API y que han sido utilizadas en la aplicación desarrollada [28]

3.2.3.1 Preparativos del entorno

Descarga del paquete "Google Play Services"

En primer lugar, dado que la API v2 se proporciona como parte del SDK de *Google Play Services,* es necesario incorporar y configurar previamente en el entorno de desarrollo dicho paquete. Para ello, se descarga el paquete llamado "*Google Play Services"* contenido en la sección Extras del SDK Manager de Android. Una vez descargado, se debe importar en Eclipse utilizando la opción de menú File -> Import y seleccionando la opción Android / Existing Android Code Into Workspace tal y como se muestra en la Figura 3.12.

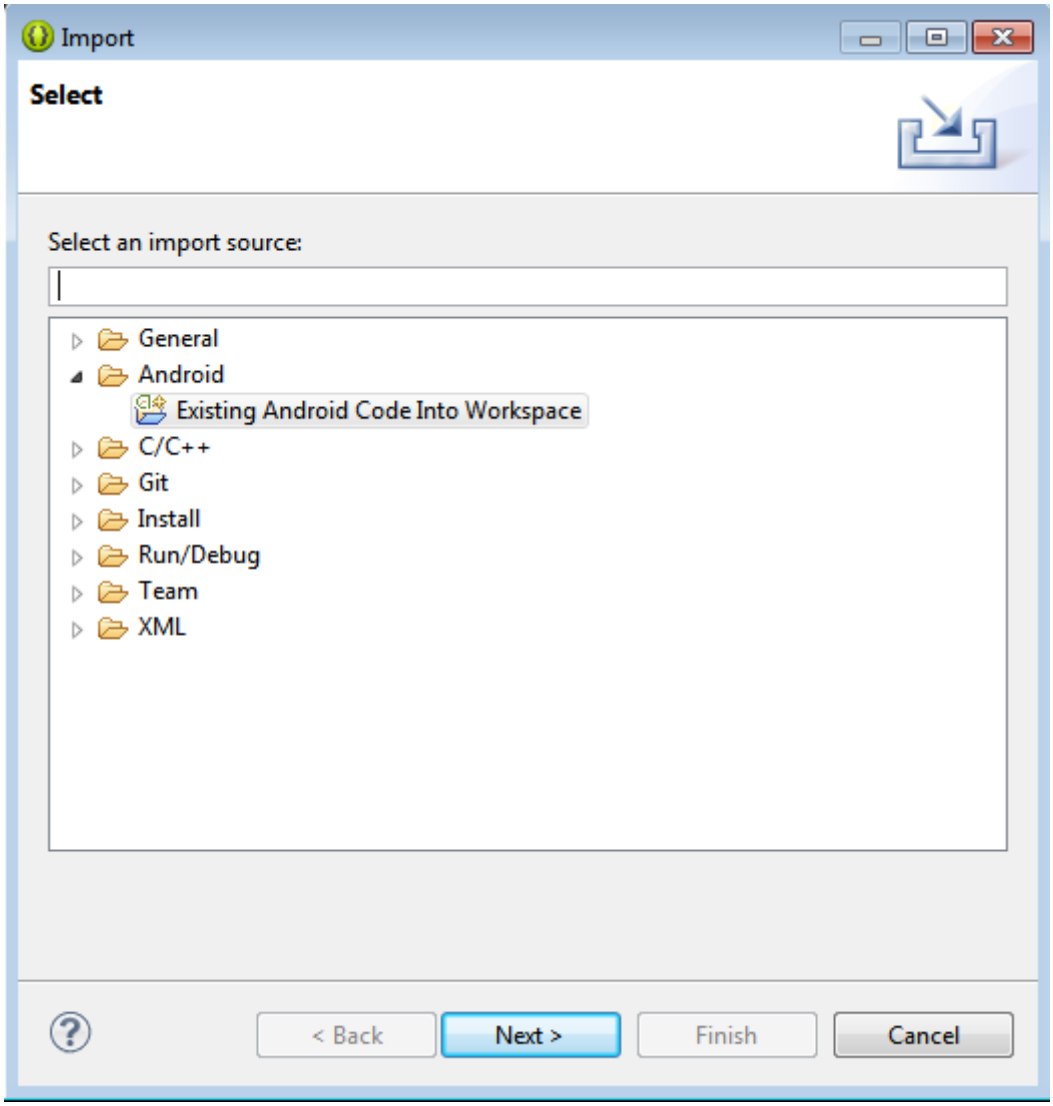


Figura 3.12: Pantalla de importación de archivos de Eclipse

Al pulsar el botón “Next”, se accede a las opciones de importación. En el campo “Root Directory” se indica la ubicación del paquete *Google Play Services* que se encuentra en la ruta:

```
<ruta-sdk>\extras\google\google_play_services\libproject\google-play-services_lib
```

Además, en esta pantalla, hay que asegurarse de que el proyecto llamado “google-play-services-lib” queda marcado en la lista de “Projects to import” y se marca la opción “Copy projects into workspace”. Finalmente, al pulsar “Finish”, el proyecto de librería de lo Google Play Services queda importado en el explorador de paquetes de Eclipse. Como último paso hay que acceder a las propiedades del proyecto importado (botón derecho / Properties), se selecciona la sección “Java Build Path” y nos aseguramos de que la opción “Android Private Libraries” está marcada en la ventana “Order and Export” tal y como se muestra en la Figura 3.13.

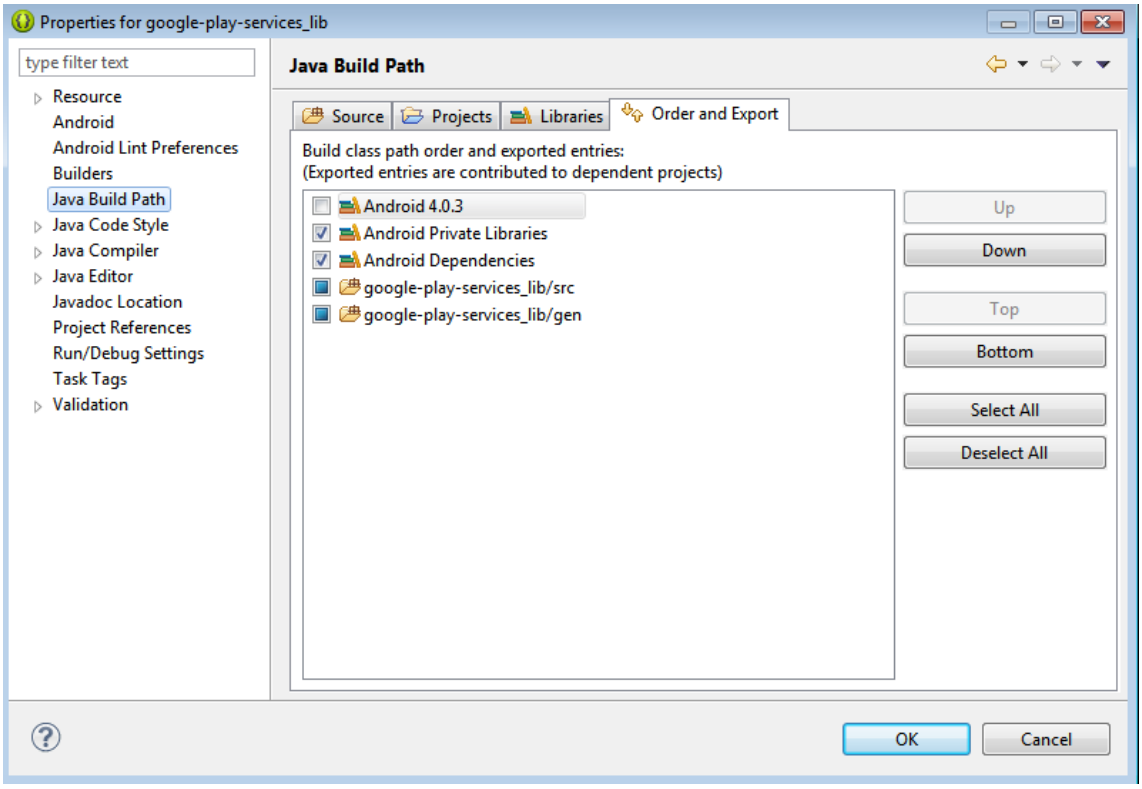


Figura 3.13: Menú de propiedades de la librería Google Play Services

Una vez que se acepte, ya se tiene el proyecto de librería preparado. A continuación, hay que añadir la referencia del proyecto Google Play Services al proyecto de la aplicación accediendo a las propiedades de este, en la sección Android y a través de la opción “Add…” como se muestra en la Figura 3.14.

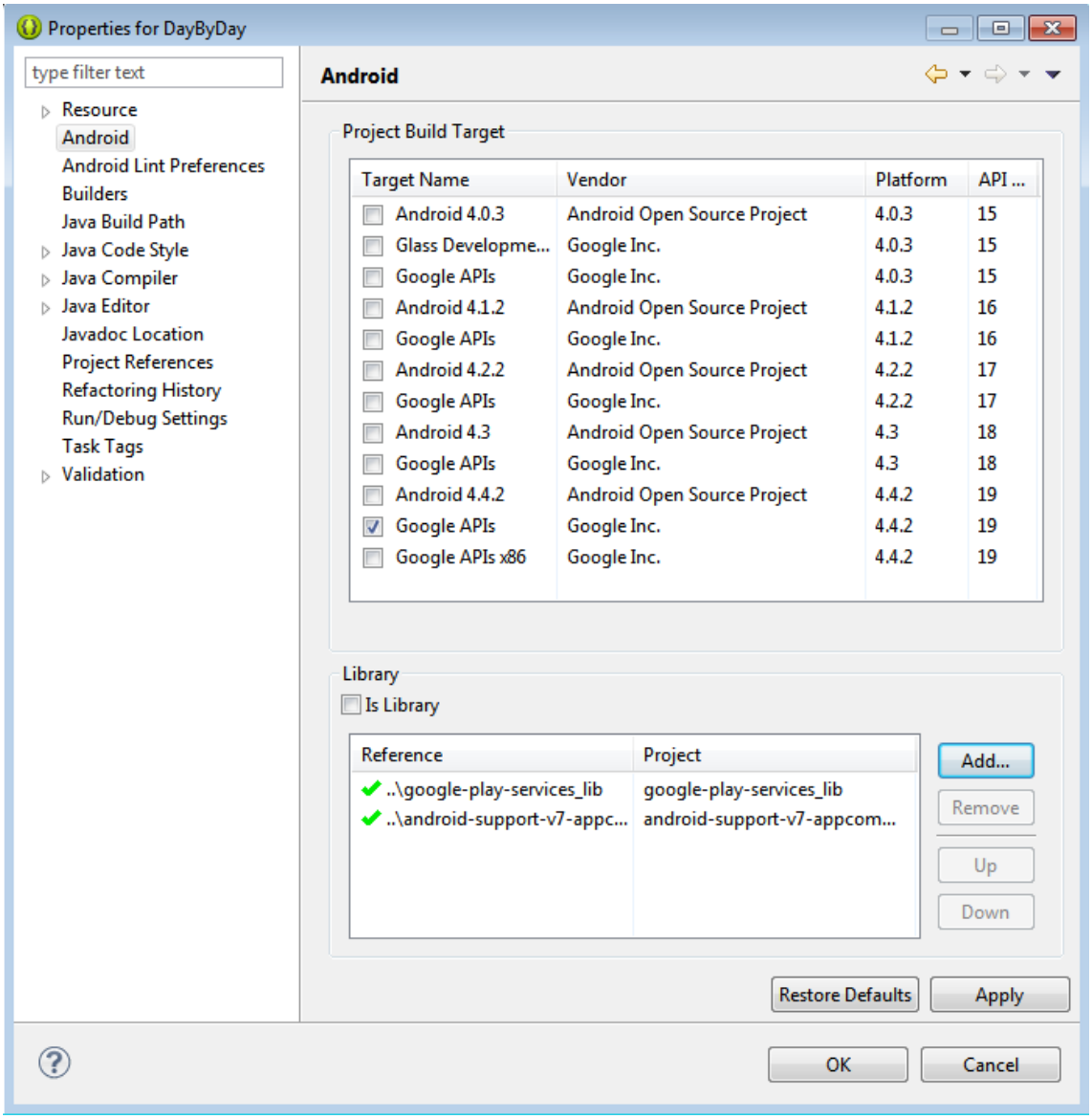


Figura 3.14: Menú de propiedades del proyecto desarrollado

Por último, es necesario editar el fichero AndroidManifest.xml de la aplicación añadiendo la cláusula `<meta-data>` dentro del elemento `<application>`, tal y como se muestra en la Figura 3.15:

```
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
```

Figura 3.15: Cláusula *meta-data* en el fichero AndroidManifest.xml para integrar la referencia del proyecto Google Play Service

Además, es necesario añadir al fichero proguard-project.txt las líneas de código que se muestran en la Figura 3.16 para evitar que esta herramienta elimine algunas clases necesarias:

```
-keep class * extends java.util.ListResourceBundle {
    protected Object[][] getContents();
}

-keep public class com.google.android.gms.common.internal.safeparcel.SafeParcelable {
    public static final *** NULL;
}

-keepnames @com.google.android.gms.common.annotation.KeepName class *
-keepclassmembernames class * {
    @ccom.google.android.gms.common.annotation.KeepName *;
}

-keepnames class * implements android.os.Parcelable {
    public static final ** CREATOR;
}
```

Figura 3.16: Fichero proguard-project.txt de un proyecto Android

Obtención de la API Key

Para poder utilizar el servicio de mapas de Google, es necesario obtener la clave *API Key*. Para ello, hay que acceder a la consola de APIs de Google (https://code.google.com/apis/console) y crear un nuevo proyecto mediante el botón "CREATE PROJECT". A continuación, tal y como se muestra en la Figura 3.17, se pide un introducir un nombre descriptivo para el proyecto así como un identificador.

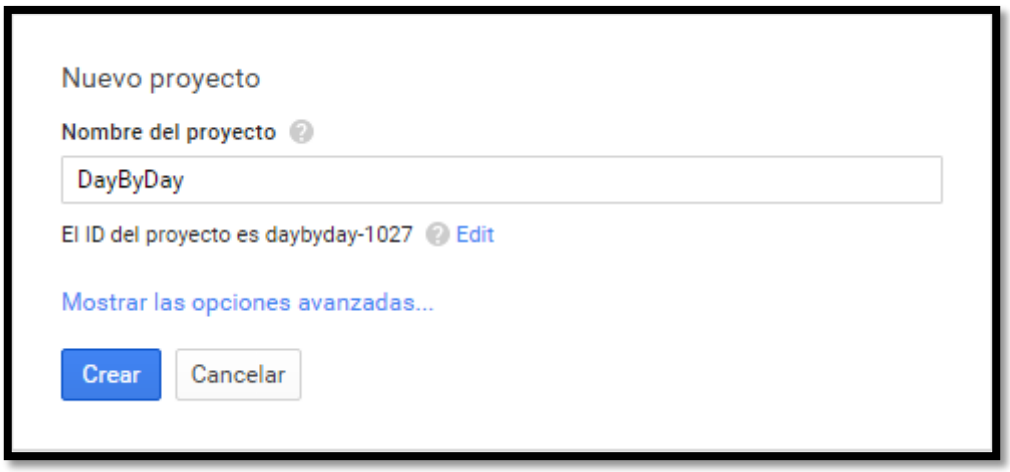


Figura 3.17: Pantalla de creación de un proyecto Google

Una vez creado el proyecto, se ha de activar la API "Google Maps Android API v2" a través del menú APIs & Auth / APIs. El siguiente paso es registrar la aplicación que se va a desarrollar a través de la opción "REGISTER APP"del menú APIs & Auth / Registered Apps. Accediendo a dicha opción se tiene la posibilidad de obtener una nueva API Key que permita utilizar el servicio de mapas desde la aplicación a desarrollar. Se debe indicar el nombre de la aplicación, su tipo ("Android"), el modo de acceso a la API (en este caso "Accessing APIs directly from Android"), el paquete java utilizado en la aplicación (que en la presente aplicación es "com.example.daybyday) y la huella digital SHA1 del certificado con el que se firma la aplicación. Este último dato

es una propiedad que debe llevar toda aplicación Android antes de ser distribuida públicamente y ejecutada en un dispositivo. En la últimas versiones de Eclipse y el plugin ADT de Android, se puede consultar directamente la huella SHA1 del certificado de pruebas de la aplicación accediendo al menú Window -> Preferences y entrando a la sección Android -> Build, tal y como muestra la Figura 3.18.

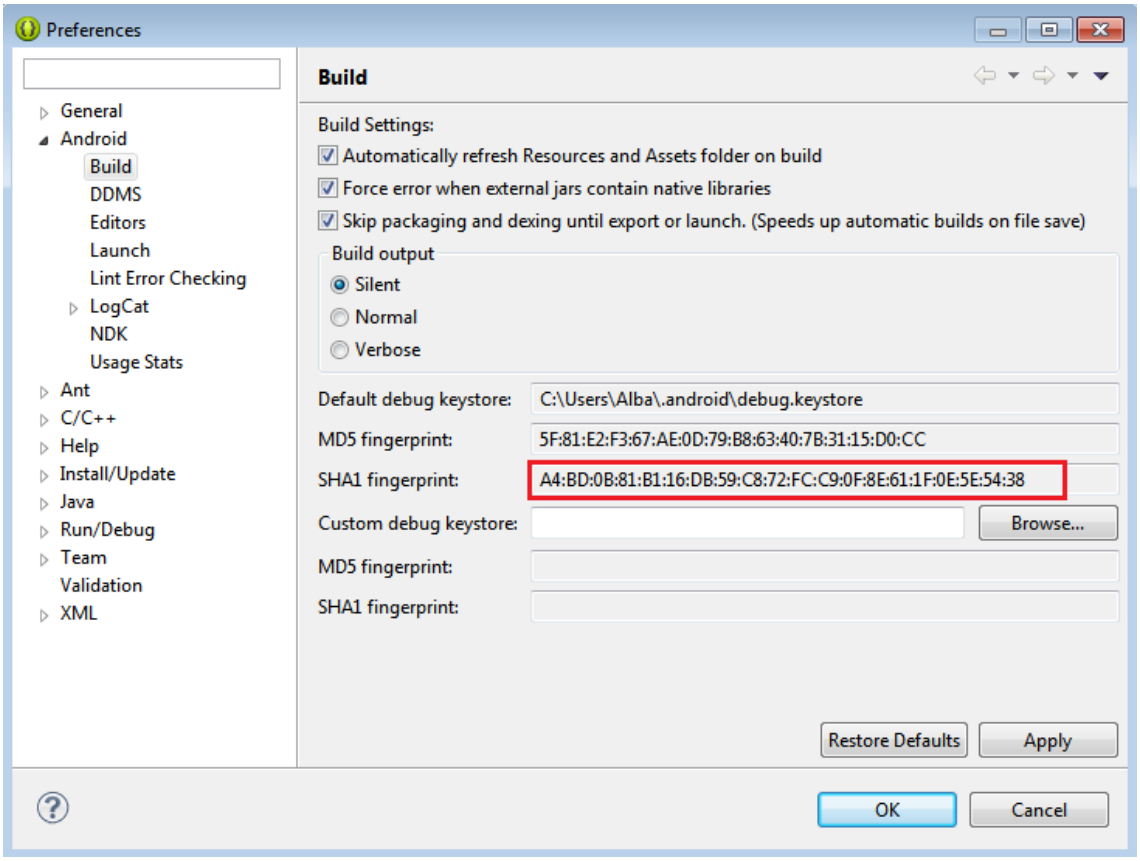


Figura 3.18: Huella digital SHA1 del certificado de la aplicación

Una vez rellenados los datos de la aplicación y aceptando el formulario, se genera la API Key necesaria, la cual se puede consultar en la pantalla siguiente dentro del apartado “Android Key”.

Configuración del proyecto para el uso de Google Maps Android API v2

Para utilizar el servicio de mapas en la aplicación a desarrollar es necesario añadir al fichero AndroidManifest.xml la API Key generada dentro de un nuevo elemento <meta-data> en la etiqueta <application> como se muestra en la Figura 3.19.

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyCXBkZvYJC_OWTHMMVB8nB2No_OwTKAuKA" />
```

Figura 3.19: API Key generada en el fichero AndroidManifest.xml

Además, se deben incluir en este mismo fichero una serie de permisos que permita el acceso a internet de la aplicación (INTERNET), conocer el estado de la red (ACCES_NETWORK_STATE), acceder al almacenamiento externo del dispositivo para la caché de mapas (WRITE_EXTERNAL_STORAGE) y hacer uso de los servicios web de Google (READ_GSERVICES). La Figura 3.20 muestra las líneas de código necesarias para incluir dichos permisos en el fichero AndroidManifest.xml.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
```

Figura 3.20: Permisos necesarios para la utilización de Google Maps Android v2

Por último, dado que el API v2 de Google Maps Android utiliza OpenGL ES versión 2, se debe especificar también dicho requisito en el fichero AndroiManifest.xml como se muestra en la Figura 3.21.

```
<uses-sdk
    android:minSdkVersion="10"
    android:targetSdkVersion="21" />
```

Figura 3.21: elemento <uses-sdk> del fichero AndroidManifest.xml

Por último, si se desea que la aplicación a desarrollar se ejecute desde la versión 2.2 de Android, como es el caso de la presente aplicación, hay que asegurarse que el proyecto incluye la librería de soporte para versiones antiguas android-support-v4.jar, que debe aparecer en la sección “Android Private Libraries” o la carpeta “libs” del proyecto.

3.2.3.2 Google Maps Android v2

La funcionalidad principal que nos ofrece este API es la de proporcionar la utilización de un mapa como interfaz gráfico, es decir, pintar un mapa en pantalla. Para dibujar un mapa en la pantalla de una actividad, se ha de añadir el control correspondiente, como un componente de tipo *fragment*, al fichero .xml de interfaz de la actividad (*layout*), tal y como se muestra en la Figura 3.22.

```
<fragment
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    class="com.google.android.gms.maps.SupportMapFragment" />
```

Figura 3.22: Componente de tipo *fragment* que represente un mapa

Para mostrar el mapa en la actividad, se asigna este *layout* a la actividad como una vista, es decir, un *ContentView* de la actividad, tal y como se muestra en la Figura 3.23.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.location_layout);
```

Figura 3.23: Asignación de un *ContentView* a una actividad

Con esto, ya se habría conseguido mostrar un mapa en pantalla el cual permite mover libremente el punto de vista (cámara) en tres dimensiones. Además, el API de Google Maps para Android permite realizar diferentes acciones con el mapa a través de un objeto de tipo GoogleMap que se describen a continuación:

- Modificar el tipo de mapa haciendo uso del método setMapType(), pasando como parámetro el tipo de mapa (MAP_TYPE_NORMAL, MAP_TYPE_HYBRID, MAP_TYPE_SATELLITE, MAP_TYPE_TERRAIN).
- La clase CameraUpdateFactory permite realizar movimientos básicos de la cámara:
    - CameraUpdateFactory.zoomIn(): Aumenta en 1 el nivel de zoom.
    - CameraUpdateFactory.zoomOut(): Disminuye en 1 el nivel de zoom.
    - CameraUpdateFactory.zoomTo(float zoom): Establece el nivel de zoom.
    - CameraUpdateFactory.newLatLng(double lat, double long): Establece la latitud y la longitud expresadas en grados.
    - CameraUpdateFactory.newLatLngZoom(double lat, double long, float zoom): Establece la latitud, la longitud y el zoom.

  Tras construir el objeto CameraUpdate con los parámetros de posición se han de llamar a los métodos moveCamera() o animateCamera() del objeto GoogleMap.

- Se pueden modificar otros parámetros de la cámara con el método más general CameraUpdateFactory.newCameraPosition(), que recibe como parámetro un objeto de tipo CameraPosition.

- Es posible conocer en un momento dado la posición de la cámara mediante el método getCameraPosition(), que devuelve un objeto de tipo CameraPosition.
- Se puede capturar el evento de pulsar en el mapa a través del uso del método setOnMapClickListener() sobre el objeto GoogleMap.
- Mostrar un marcador en una posición indicada y con un título a través de addMarker().
- Permite dibujar líneas y polígonos sobre el mapa utilizando la clase PolylineOptions.

3.3 Implementación de las operaciones generales

Esta sección describe la implementación de las siguientes operaciones generales que han sido necesarias en el desarrollo de la aplicación Android.

3.3.1 Servicios basados en localización

Para la implementación de servicios basados en localización, la plataforma Android nos ofrece los llamados proveedores de localización (*location providers*), diversas tecnologías utilizadas para obtener la localización del dispositivo (Ej.: GPS o localización por red) y gestores de localización (*location manager*), clases que facilitan el acceso al sistema de los proveedores de localización. Cómo se ha introducido en la Sección 3.1.2 del presente trabajo, tanto el módulo de Lugares como el módulo de Eventos hacen uso de estos servicios, junto con la API de Google Maps descrita en la Sección 3.2.3 para hacer la actividad más visual y completa.

A continuación, se describen en mayor detalle las funcionalidades y características que ofrecen los gestores y proveedores de localización de Android para la creación de servicios basados en localización y que han sido utilizados en la aplicación desarrollada.

Obtención de una localización

Para obtener la localización del dispositivo, implementado en el módulo Lugares, se hace uso de los diferentes proveedores de localización de los que dispone. Dependiendo del dispositivo, éste puede que disponga de varios proveedores, diferenciándose entre ellos en cuanto a consumo de energía, coste económico, precisión, capacidad de determinar la altitud, velocidad u orientación. Por tanto, el primer paso para obtener la localización es seleccionar el proveedor de localización que mejor se adapta a las características de la aplicación. Para obtener un proveedor de localización, Android ofrece como gestor de localización la clase LocationManager. Esta clase ofrece dos maneras de obtener un proveedor de localización:

- Obteniendo una instancia de un proveedor específico llamando al método `getProvider(String name)` como se muestra en la Figura 2.24. Los dos más comunes son `LocationManager.GPS_PROVIDER` y `LocationManager.NETWORK_PROVIDER`.

```
String providerName = LocationManager.GPS_PROVIDER;
LocationProvider gpsProvider = locationManager.getProvider(providerName);
```

Figura 3.24: Obtención de una instancia del gestor de localización de tipo GPS

- Obteniendo una instancia de un proveedor especificando los requisitos deseados. Los requisitos se especifican con la clase `Criteria`, en términos de precisión (fine, coarse), consumo de energía (low, médium, high), coste y capacidad de ofrecer altitud, velocidad y orientación. Una vez definidos los criterios, se hace uso del método `getBestProvider(Criteria criteria, boolean enabledOnly)` para obtener el que mejor se ajuste. En el caso de que ningún proveedor de localización cumpla todos los criterios, éstos se van relajando en el siguiente orden: energía, precisión y capacidad de obtener otros datos (el criterio de coste nunca se relaja).

```
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_COARSE);
criteria.setPowerRequirement(Criteria.POWER_LOW);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setSpeedRequired(false);
criteria.setCostAllowed(true);

String bestProvider = locationManager.getBestProvider(criteria, true);
```

Figura 3.25: Obtención de una instancia de un proveedor de localización utilizando la clase Criteria.

Una vez obtenido la instancia del proveedor de localización, se puede obtener la localización del dispositivo mediante el servicio LocationManager. En primer lugar, se solicita una instancia de dicha clase a través de LOCATION_SERVICE tal y como se muestra en la Figura 3.26. A continuación, se obtiene la localización del dispositivo a través del método `getLastKnownLocation (String provider)` utilizando el proveedor obtenido. Este método devuelve los datos de la última localización obtenida por el proveedor dado por lo que no hay garantía de que esta exista o que la localización que devuelva siga siendo relevante. No obstante, es la mejor opción que ofrece el API de Android para obtener una sola localización. La Figura 3.26 muestra un ejemplo de código en el que se obtiene la localización del dispositivo utilizando lo descrito anteriormente.

```
String serviceString = Context.LOCATION_SERVICE;
LocationManager locationManager = (LocationManager)getSystemService(serviceString);
String provider = LocationManager.GPS_PROVIDER;
Location location = locationManager.getLastKnownLocation(provider);
```

Figura 3.26: Ejemplo de obtención de una localización en Android

Traducción entre direcciones y coordenadas

La clase Geocoder permite traducir entre direcciones y coordenadas. Esta traducción la realiza un servidor por lo que es necesario incluir el permiso correspondiente (`android.permission.INTERNET`) en el fichero `AndroidMinfest.xml` de la aplicación. La petición de la traducción se contextualiza con la localización habitual e idioma configurados en el dispositivo y puede devolver varios resultados si se indica.

En el módulo *Lugares*, se muestra al usuario la dirección (calle) en la que se encuentra. Esto es posible gracias al *Reverse Geocoding*, que es la traducción de localizaciones en coordenadas a calles.

Por otro lado, en el módulo *Eventos*, se muestra al usuario un mapa con la localización del evento. Para ello, es necesario utilizar *Forward Geocoding* de forma que se traduzca la dirección introducida por el usuario a coordenadas, que son las utilizadas por el objeto GoogleMaps para pintar el mapa.

3.3.2 Grabación y reproducción de audio

En el módulo Lugares se hace uso de grabación y reproducción de audio en Android ya que esta actividad permite al usuario asociar un fichero de audio, a modo de descripción, a la localización guardada.

Grabación de audio con MediaRecorder

La grabación se inicia y se para con el botón de la izquierda, el usuario conoce la acción de dicho botón según la imagen correspondiente, tal y como se observa en la Figura 3.27.

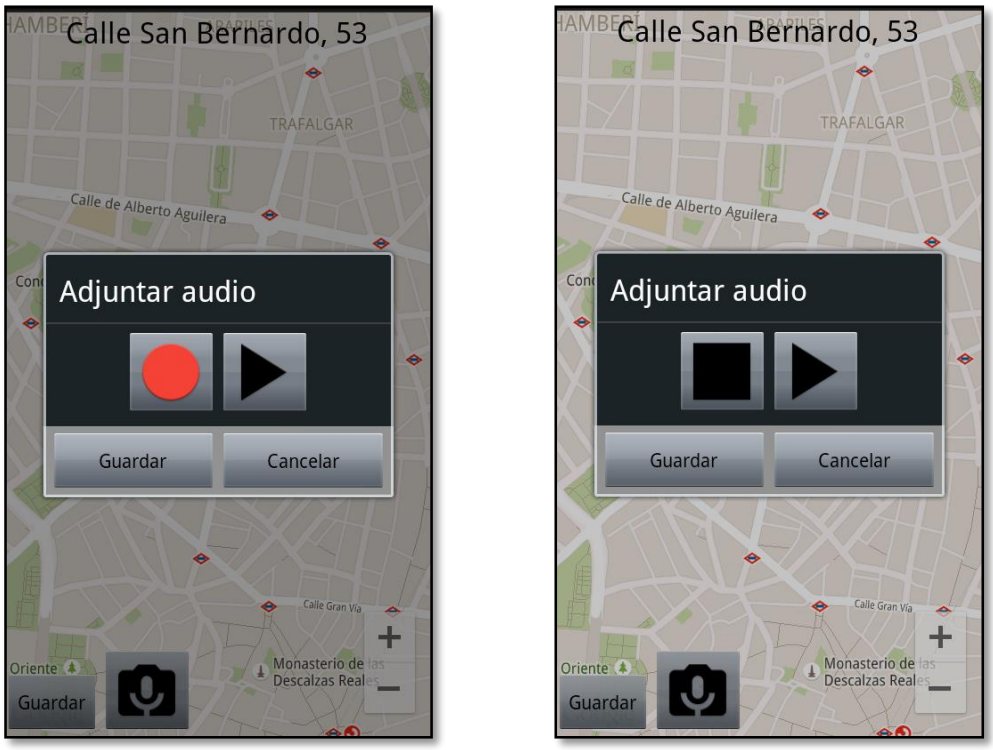


Figura 3.27: Actividad de grabación de audio en la aplicación desarrollada

La clase utilizada para la grabación de un fichero de audio en Android es `MediaRecorder`. Esta clase ofrece el método start(), el cual se encarga de comenzar la grabación. Previamente, hay que configurar algunos parámetros importantes del objeto `MediaRecorder`, los cuales se listan a continuación:

- La fuente desde donde se desea grabar (distintos micrófonos del dispositivo) con el método `setAudioSource (int audio_source)`.
- El decodificador de audio a utilizar a partir del método `setAudioEncoder (int audio_encoder)`.
- El formato de salida del fichero de audio (recomendable utilizar 3GP cuando se usa decodificador de audio AMR) con el método `setOutputFormat (int output_format)`.
- La ruta completa más el nombre del nuevo fichero utilizando `setUotputFile (String path)`.

Además, antes de comenzar la grabación es aconsejable asegurarse de que el grabador de audio está listo para la grabación utilizando el método `prepare()`. Cabe añadir que la grabación y almacenamiento de ficheros de audio en Android necesita dos tipos de permisos que se han de añadir al fichero `AndroidManifest.xml`: permiso para grabar audio (`RECORD_AUDIO`) y permiso para escribir en la memoria externa (`WRITE_EXTERNAL_STORAGE`).

Para parar la grabación, se hace uso del método `stop()`. A continuación, se liberan los recurso del objeto `MediaRecorder` llamando al método `release()`.

Reproducción de audio con MediaPlayer

Esta actividad permite al usuario reproducir el fichero de audio una vez grabado. Esto se realiza a través del botón derecho de la pantalla el cual, tal y como se muestra en la Figura 3.28, también se utiliza para parar la reproducción.

Figura 3.28: Actividad para la reproducción de audio en la aplicación desarrollada

La clase utilizada para la reproducción de un fichero de audio es `MediaPlayer` que ofrece el método `start()` sobre un objeto de esta clase para comenzar la reproducción. Previamente, es necesario indicar la ruta del fichero a través del método `setDataSource (String path)`, así como asegurarse de que el reproductor está listo para ser utilizado con el método `prepare()`.

3.3.3 Integración de un RESTful Web Service en Android para la extracción de contenido de páginas web

Para la extracción de contenido web en la aplicación desarrollada, se ha hecho uso de un servicio RESTful, un servicio web que implementa la arquitectura REST. Los servicios REST (*Representational State Transfer*) [31] se basan en una arquitectura para desarrollar servicios web de tipo cliente/servidor y son los más utilizados actualmente a la hora de crear un servicio Web. A continuación, se resumen las características de este tipo de servicios:

- **Arquitectura cliente-servidor:** consiste en una separación clara entre los dos agentes básicos en un intercambio de información.
- **Sin estado (*stateless):*** Son servicios web que no mantienen un estado asociado al cliente, es decir, cada petición que se realiza es completamente independiente de la siguiente.
- **Cacheable:** El contenido de los servicios web REST de puede cachear de tal forma que una vez realizada la primera petición al servicio el resto pueden apoyarse en el caché si fuera necesario.
- **Servicios uniformes:** Todos los servicios REST comparten una forma de invocación y métodos uniforme utilizando los métodos GET, POST, PUT y DELETE.
- **Arquitectura en capas:** un sistema por capas que permite la independencia entre el cliente y las capas por las que se tramita la información, lo que aumenta la escalabilidad.

En la aplicación desarrollada se utiliza este servicio para descargar la información de las películas que el usuario guarde en el módulo "Notas" cuando la categoría es "Cine". Con este objetivo, se ha utilizado la API REST OMDb [32], la cual permite acceder a información almacenada en la Web IMDb [33]. En resumen, la aplicación realiza una petición de tipo GET al servidor cuya respuesta es un objeto de tipo JSON con la información solicitada. A continuación, se explica en mayor detalle estas dos acciones de petición y respuesta.

Realización de la petición GET

Para realizar la petición GET y poder acceder a la información devuelta en la respuesta, se ha hecho uso de la librería http-request. Para instalar dicha librería a la aplicación desarrollada has sido necesario descargar el fichero *HttpRequest.java* [34] y añadirlo a la carpeta *src* del proyecto.

Una vez incluida la librería, se puede realizar la petición al servidor. La página web de OMDb API facilita la generación de la URL para realizar la petición especificando el título de la película y el tipo de respuesta (que en este caso será JSON). Un ejemplo de la URL generada es:

http://www.omdbapi.com/?t=Big+fish&y=&plot=short&r=json

Por tanto, únicamente es necesario parametrizar el valor especificado en el parámetro *title* que será el texto introducido por el usuario en la nota. Para realizar la petición GET a partir de la URL generada, es necesario la utilización de una tarea *AyncTask* en la propia actividad. La clase auxiliar AyncTask permite ejecutar tareas en segundo plano para realizar operaciones largas o costosas con el objetivo de evitar que se bloqueen las operaciones ejecutadas en el hilo principal de la aplicación. La forma básica de utilizar esta clase consiste en crear una nueva clase que extienda de AsyncTask y sobrescribir varios de sus métodos entre los que se reparte la funcionalidad de la tarea. En concreto, se ha sobrescrito el método doInBackGround() para realizar en él la

petición a través del método get() que proporciona la librería http-request, tal y como se muestra en la Figura 3.29.

```
public class LoadFilmTask extends AsyncTask<String, Long, String>{


    @Override
    protected String doInBackground(String... urls) {
        try {
            return HttpRequest.get(urls[0]).accept("application/json")
                    .body();
        } catch (HttpRequestException exception) {
            return null;
        }
    }
```

Figura 3.29: Método doInBackground() de la clase Asyntask

Procesamiento de la respuesta del servidor

Como se ha comentado anteriormente, la petición al servidor devuelve un objeto de tipo JSON por lo que es necesario hacer uso de la librería Gson para analizar sintácticamente dicho objeto, es decir, obtener su equivalente como un objeto Java. Por tanto, previamente es necesario descargar tal librería y añadirla al proyecto.

Para analizar un objeto JSON para obtener un objeto Java equivalente, es necesario crear una clase Java que incluya los campos del objeto JSON. En concreto, el objeto JSON devuelto por el servidor IMDb consta de *string,* números y arrays de *string,* a excepción del atributo *poster,* que es un objeto JSON anidado. Por lo tanto, en la aplicación desarrollada, se ha hecho uso de una clase para la información de la película y otra para la información del poster. Es necesario que el nombre de las propiedades de las clases debe coincidir con el nombre de los atributos del objeto JSON.

La respuesta del servidor se procesa en el método onPostExecute (String response) de la clase AsyncTask, siendo el atributo *response* el objeto JSON en forma de *String.* Por tanto, se puede obtener un objeto Película fácilmente de la siguiente manera:

```
Gson gson = new GsonBuilder().create();
Pelicula pelicula = gson.fromJson(response, Pelicula.class);
```

Una vez obtenido el objeto película, es fácil obtener cualquiera de sus atributos a partir de los métodos *get<atributo>()* para que sean mostrados por pantalla.

3.3.4 Implementación del reconocimiento de voz y de la síntesis de texto a voz

Exceptuando los módulos *Lugares* y *Filtrado por fecha*, que únicamente permiten la entrada y salida de datos mediante interfaz táctil y visual, el resto de módulos incorporan el reconocimiento de voz y la síntesis de texto a voz con el fin de dotar al sistema de las características propias de un sistema de diálogo.

Integración del reconocimiento de voz en la aplicación para dispositivos móviles Android

La opción más simple a la hora de integrar el reconocimiento de voz en una aplicación para dispositivos móviles Android consiste en utilizar cualquier servicio de reconocimiento de voz ya existente que pueda recibir un *RecognizerIntent,* siendo únicamente necesario el uso de la clase android.speech.RecognizerIntent descrita en la Sección 2.2.2.2 del presente trabajo. La aplicación utilizada para el reconocimiento de voz es *Google Voice Search.* Esta aplicación consiste en una pantalla inicial con el texto "Habla ahora" que indica al usuario cuándo el dispositivo se encuentra "escuchando". El audio recogido se transfiere a los servidores de Google para que se haga el reconocimiento y, posteriormente, los resultados son devueltos a la aplicación para que puedan ser procesados. Si bien, como se ha mencionado en la Sección 2.2, a partir de la versión **Android 4.1 Jelly Bean** está disponible el dictado por voz *offline,* por lo que, en este caso, el reconocimiento de voz se realiza en local, sin pasar por lo servidores de Google.

Para integrar el reconocimiento de voz mediante la clase android.speech.RecognizerIntent se deben seguir los pasos ya descritos en la Sección 2.2.2.3: verificar que existe la actividad para reconocer voz, invocar a la actividad de reconocimiento de voz y procesar los resultados. Además, dado que para que funcione el reconocimiento de voz es necesario tener acceso a Internet, en el archivo AndroidManifest.xml se añade dicho permiso (android.permission.INTERNET).

En cuanto a la configuración de la actividad de reconocimiento de voz, se ha especificado únicamente el parámetro correspondiente al modelo del lenguaje (`EXTRA_LANGUAJE_MODEL`) como `LANGUAJE_MODEL_FREE_FORM`.

El reconocimiento de voz está preparado para entender dos idiomas: español si el idioma del dispositivo está configurado como tal idioma o inglés en cualquier otro caso.

Integración de la síntesis de texto a voz en la aplicación para dispositivos móviles Android

En cuanto a la síntesis de texto a voz, se ha visto en la Sección 2.3 que a partir de la versión 1.6 de Android se incorpora un motor de síntesis que se encuentra, generalmente, instalado por defecto en el sistema y que es utilizado por las demás aplicaciones. El más habitual y el que se encuentra instalado en la mayoría de los dispositivos es el denominado Pico TTS, desarrollado por SVOX y Google. Sin embargo hemos estudiado multitud de motores de síntesis alternativos desarrollados para Android de forma que podamos elegir el que mejor se adapte a las necesidades de nuestra aplicación.

La funcionalidad de la síntesis de texto a voz en la aplicación desarrollada es la de preguntar al usuario sobre las acciones que desea realizar para guiar al usuario a través de la aplicación sin necesidad de utilizar interfaz táctil. Este servicio se activa siempre y cuando el usuario haga la petición de una acción a través de la voz. De esta manera, la aplicación entiende que el usuario prefiere este tipo de interactuación con el dispositivo y se emula un diálogo entre la aplicación y el usuario. Dependiendo de la actividad y del punto de ejecución en el que se encuentre la aplicación, existen distintos textos para ser reproducidos, ya sean para preguntar al usuario por la acción siguiente, para comunicar que la petición recibida no se ha entendido o no es correcta o para confirmar que una acción determinada ha sido realizada.

Capítulo 4

Descripción detallada de los módulos del sistema

En este capítulo se realiza una descripción detallada de cada uno de los módulos que componen la aplicación para dispositivos móviles Android desarrollada en el presente trabajo. Para cada módulo se detalla la funcionalidad, arquitectura, flujo de datos y ejemplos de posibles escenarios usos.

4.1 Módulo Principal

El módulo principal consta de la actividad principal o de inicio de la aplicación, desde la que el usuario tiene disponible toda la información que ha ido almacenando hasta la fecha. Dicha información puede ser filtrada con distintos métodos de búsqueda así como consultada, editada o eliminada. Desde este módulo se tiene acceso a todos los demás módulos del sistema.

Figura 4.1: Captura de pantalla del módulo Principal

La Figura 4.1 muestra la pantalla que representa el módulo principal. Como se observa, dicha pantalla está dividida en cuatro pestañas, una por cada tipo de categoría de información que ofrece la aplicación. Por tanto, en este módulo la información se encuentra dividida según su naturaleza, además de organizada por orden cronológico.

4.1.1 Funcionalidad

A parte de proporcionar al usuario una vista general de todos los ítems guardados en la aplicación, este módulo tiene tres funcionalidades principales: acceder a la consulta, edición o eliminación de un ítem, accediendo de esta manera a uno de los módulos de las cuatro categorías, acceder a la pantalla de creación de un nuevo ítem, que también enlaza con los módulos mencionados, o el filtrado y búsqueda de información para su posterior consulta.

4.1.1.1 Acceso a la creación de un nuevo ítem

La aplicación permite acceder a la pantalla de creación de un nuevo ítem a través de los dos tipos de interfaz que ofrece la aplicación: interfaz táctil e interfaz oral. La interfaz táctil permite acceder fácilmente a la pantalla de creación a través del botón “Nueva entrada”. A continuación, se debe elegir el tipo de categoría que se desea crear a través de un menú de opciones, tal y como se muestra en la Figura 4.2, para que la aplicación acceda a la pantalla de creación correspondiente.

Figura 4.2: Menú de opciones para elegir la categoría de la nueva entrada

Por otro lado, si el usuario decide utilizar la interfaz oral, debe acceder a la aplicación *Google Search* a través del botón simbolizado con un micrófono. En la opción "Ayuda" del menú (accesible desde el botón "Menú" del dispositivo) del presente módulo se explica el tipo de comando a utilizar para lanzar esta actividad: únicamente es necesario decir "Crear" seguido del tipo de categoría, siendo la aplicación bastante flexible con el vocabulario utilizado (Ej.: para la categoría *Foto*, se podrán utilizar tanto la palabra "foto" como "imagen"). En el caso de los módulos de *Eventos* y *Notas*, al activar las pantallas de creación con la voz, las siguientes interacciones también son orales, es decir, se emula un diálogo entre la aplicación y el usuario de forma que no se necesite la interfaz táctil para la creación completa del nuevo ítem. Esto no ocurre para la creación de una imagen o una ubicación ya que el manejo de estas actividades conlleva mayor complejidad. La pantalla de creación de un nuevo ítem se explicará en mayor detalle cuando se describan los módulos correspondientes.

4.1.1.2 Acceso a la consulta, edición y eliminación de un ítem

La consulta de cualquier ítem se puede hacer directamente pulsando sobre uno de ellos desde este módulo. Además, si se realiza una pulsación larga sobre un ítem, se accede a un menú con las acciones disponibles a realizar sobre un ítem: "Ver", "Editar" y "Eliminar", tal y como se observa en la Figura 4.3.

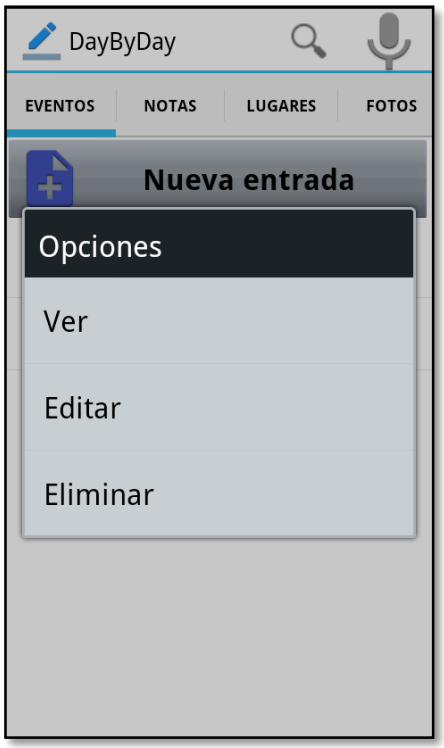


Figura 4.3: Menú de opciones tras una pulsación larga sobre un ítem

Este módulo también permite la eliminación de varios ítems a la vez a través de la opción “Eliminar(…)” del menú (accesible desde el botón “Menú” del dispositivo) que da acceso a diferentes opciones de eliminación de datos, mostradas en la Figura 4.4. En cualquier caso, la acción *Eliminar* borra permanentemente de la base de datos una o varias entradas seleccionadas.

Figura 4.4: Menú de sub-opciones de la opción “Eliminar(…)”

4.1.1.3 Filtrado y búsqueda de la información

La aplicación desarrollada permite dos tipos de búsqueda con el objetivo de hacer más accesible la información almacenada: por palabra clave o por fecha.

Búsqueda por palabra clave

La búsqueda por palabra clave se puede realizar a través de los dos tipos de interfaz: táctil y oral. La búsqueda por interfaz táctil se realiza a través de la barra de búsqueda situada en la parte superior de la pantalla. Una vez que se ha pulsado dicha barra, se abre automáticamente el teclado del móvil para permitir al usuario que escriba la palabra que quiere buscar. Esta barra de búsqueda está implementada de tal manera que se haga una consulta a la base de datos cada vez que se teclee una letra, por lo que el filtrado se va realizando dinámicamente a medida que el usuario escribe. Los resultados de la búsqueda serán aquellos ítems en los que coincida alguna letra o palabra contenida en su descripción con lo escrito en la barra de búsqueda.

Por otro lado, la búsqueda por interfaz oral se realiza a través de la aplicación *Google Search* mencionada anteriormente. En este caso, el usuario sólo tiene que dictar la palabra que desea buscar. Entonces, la palabra reconocida por la aplicación se escribe automáticamente en la barra de búsqueda, de forma que el usuario puede comprobar qué es lo que ha entendido el reconocedor de voz y editar la búsqueda manualmente si lo desea. El resultado de la búsqueda es similar al anterior.

Búsqueda por fecha

Así mismo, se pueden realizar búsquedas a partir de una fecha dada, ya que todos los ítems están caracterizados por un fecha, ya sea de creación (en el caso de imágenes y ubicaciones) o elegida por el usuario (en el caso de eventos y notas). Este tipo de búsqueda también se puede realizar a través de los dos tipos de interfaces existentes en la aplicación. Si se desea realizar a través de interfaz táctil, esta opción se encuentra en el menú del presente módulo con el nombre "Filtrado por fecha" la cual lanzará la actividad filtrando por la fecha del día actual por defecto y permitiendo, a través de un control situado en la parte superior de la pantalla, modificar manualmente la fecha de la búsqueda. Si se desea acceder por voz, solamente es necesario que el usuario dicte la fecha completa (Ej: 15 de mayo de 2011) a través de la aplicación *Google Search.* Igualmente, se abrirá una nueva pantalla con el resultado de la búsqueda pero esta vez realizando el filtrado directamente por la fecha dictada. La pantalla con el resultado de una búsqueda por fecha se muestra en la Figura 4.5.

Figura 4.5: Captura de pantalla de la actividad *Filtrado por fecha*

Como se observa en la imagen, esta pantalla es similar a la pantalla principal, con la diferencia de que a esta se le añade un control para modificar la fecha del filtro manualmente, tal y como se muestra en la Figura 4.6.

Figura 4.6: Control de modificación de fecha en la actividad *Filtrado por fecha*

De la misma forma que desde la pantalla principal, desde esta pantalla también se puede consultar, editar o eliminar un ítem. Además, también se puede acceder a las pantallas de creación de un nuevo ítem desde el botón representado con un símbolo '+'. Por último, si puede volver a la pantalla principal pulsando la opción "Ver todo".

4.1.2 Arquitectura y flujo de datos

La Figura 4.7 muestra la arquitectura y flujo de datos del módulo Principal. Tal y como se observa en la imagen, el proceso general se puede resumir en los siguientes pasos:

1) El usuario introduce una búsqueda o solicita realizar una acción (crear un ítem), ya sea por interfaz táctil o través de la aplicación de reconocimiento de voz.
2) El sistema procesa la petición del usuario y, según esta, decide la acción siguiente. El siguiente paso es el caso 3) si trata de una búsqueda o 5) si se trata de una acción.
3) La aplicación realiza la consulta a la base de datos interna SQLite con el criterio de búsqueda correspondiente.
4) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.
5) La aplicación muestra el resultado de la petición del usuario: el resultado de la búsqueda o la actividad que permite realizar la acción solicitada.

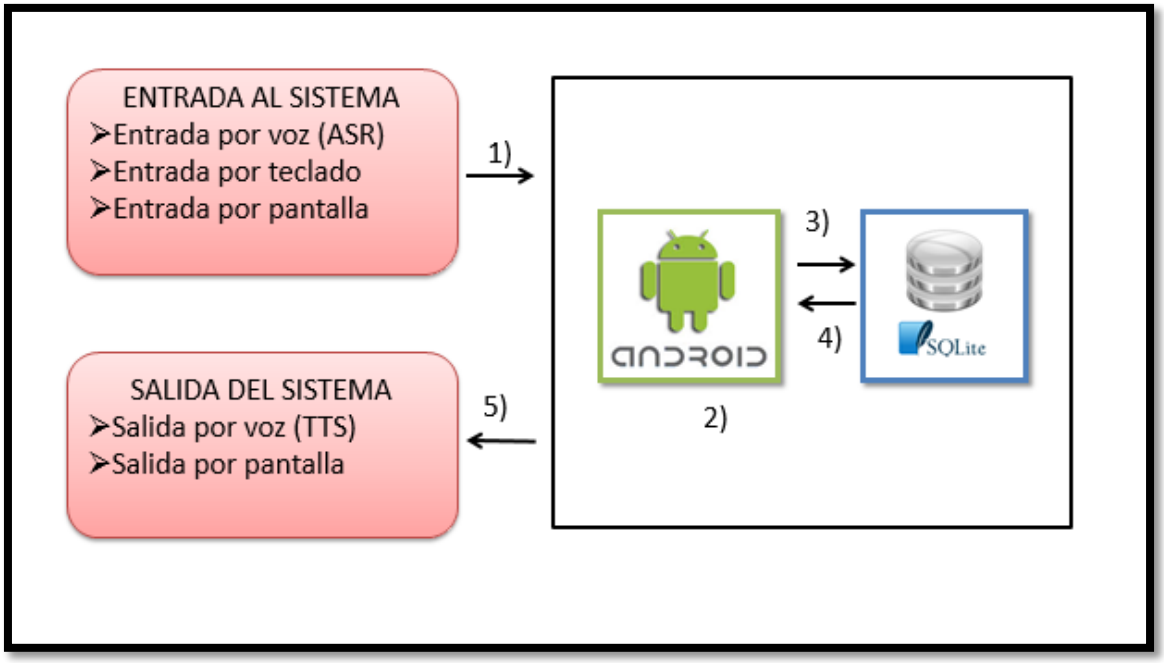


Figura 4.7: Arquitectura y flujo de datos del módulo Principal

4.1.3 Escenario de uso

La Figura 4.8 muestra las cuatro pestañas de la pantalla de inicio con las que el usuario se encuentra en la primera interacción con la aplicación. En estas pantallas se encuentran los accesos a todos los ítems guardados por el usuario hasta la fecha.

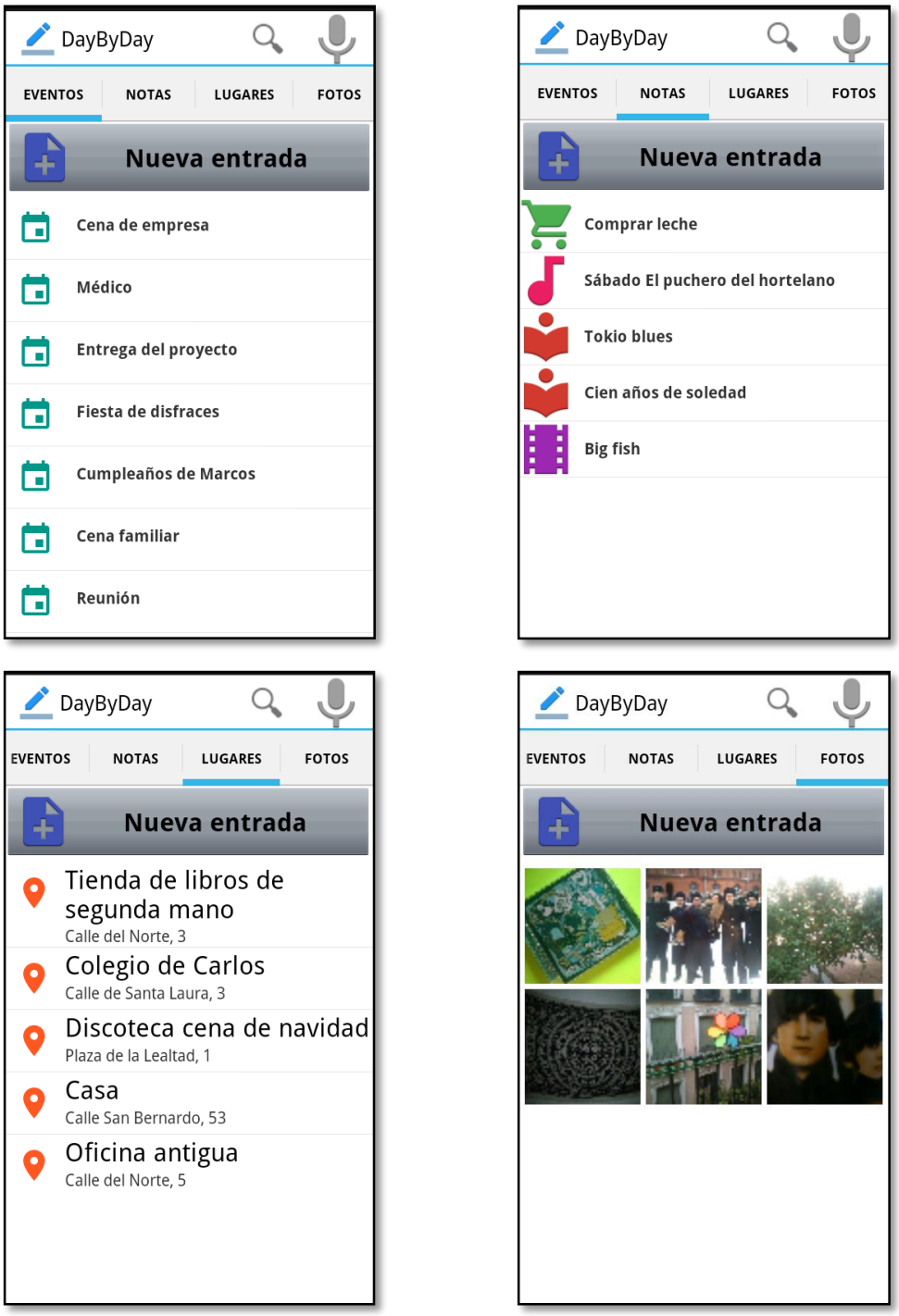


Figura 4.8: Captura de pantalla de las cuatro pestañas que forman la pantalla de inicio

A continuación, el usuario accede a la aplicación *Google Search* para dictar la búsqueda, tal y como se observa en la Figura 4.9.

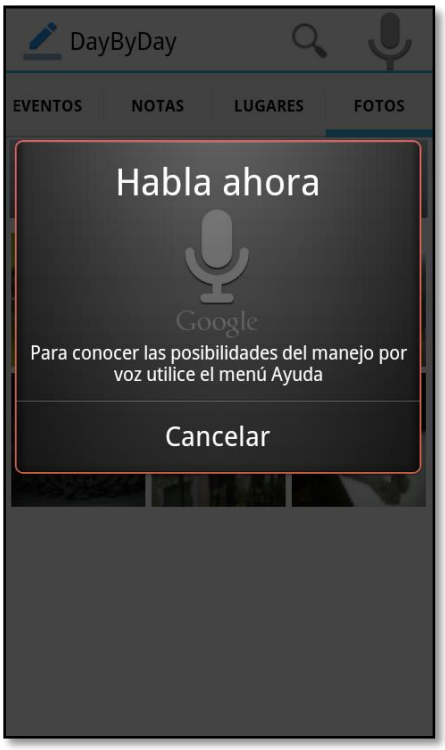


Figura 4.9: Actividad de reconocimiento de voz de la actividad Principal

El reconocedor de voz procesa la palabra recibida, cuyo resultado es utilizado por la aplicación para realizar la consulta a la base de datos SQLite. La Figura 4.10 muestra el resultado de la búsqueda.

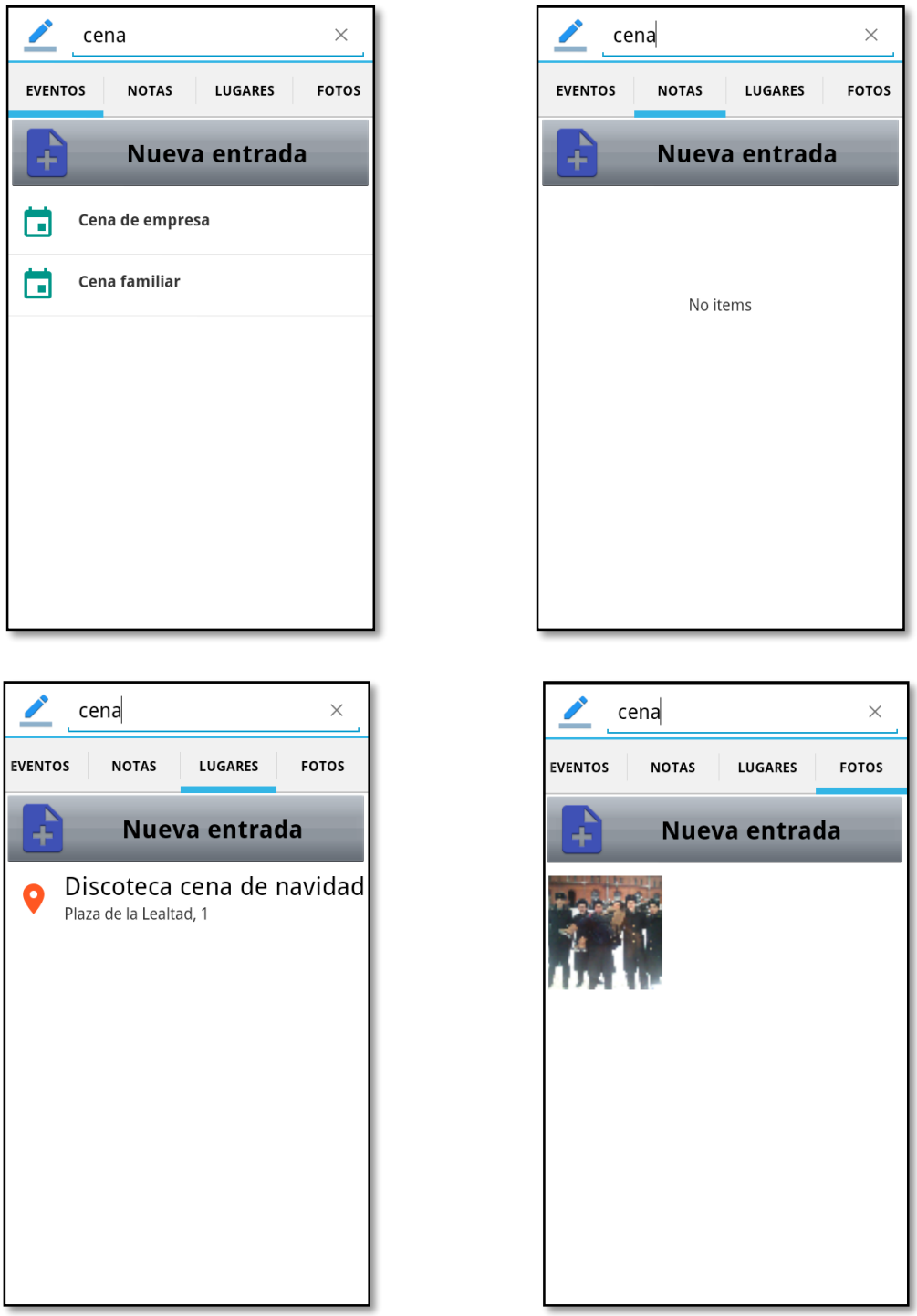


Figura 4.10: Resultado de la búsqueda por palabra clave

4.2 Módulo de Eventos

El módulo de Eventos comprende todas aquellas acciones relacionadas con la información asociada a la categoría *Evento*. Este tipo de información está caracterizada por la fecha y hora del evento, la descripción (asunto) y la ubicación del mismo. El

presente módulo puede dividirse en tres sub-módulos o actividades: ***Creación, Edición*** y ***Consulta.***

4.2.1 Actividad de Creación

4.2.3.1 Funcionalidad

La funcionalidad de la presente actividad es la creación de un nuevo ítem asociado a la categoría *Evento*. La Figura 4.11 muestra la pantalla de esta actividad.

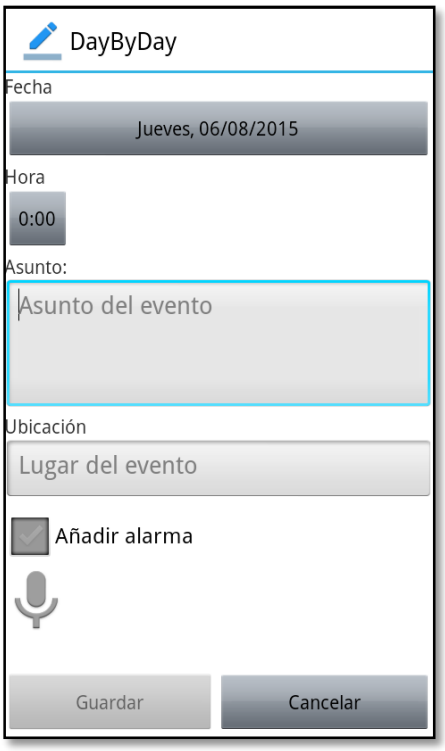


Figura 4.11: Captura de pantalla de la actividad *Creación de evento*

Como se puede observar, dicha pantalla consta de un formulario a rellenar con los datos del posible evento: la fecha, la hora, el asunto (descripción del evento) y la ubicación del evento. El campo "Asunto" se trata de un campo obligatorio ya que resulta ser el campo más descriptivo de este tipo de categoría y el que será utilizado por las base de datos para las búsquedas por palabra clave. Una vez que dicho campo se ha rellenado, la aplicación permite guardar el evento pulsando el botón "Guardar". Además, se puede descartar en cualquier momento pulsando el botón "Cancelar".

Este Actividad permite rellenar los campos "Asunto" y "Ubicación" a través de interfaz oral. La aplicación está preparada para entender el contenido de estos dos campos dictados en una misma frase de la forma natural en que una persona lo diría: por ejemplo, "Cumpleaños en mi casa", entendiendo que lo que se encuentra previamente a la preposición es el asunto y lo que le precede es la ubicación. Si se ha accedido a él por interfaz táctil, es posible acceder a la aplicación *Google Search* a través del botón

del micrófono presente en la pantalla. Por otro lado, si se ha accedido a él por voz, se lanzará automáticamente el sintetizador de texto a voz con la pregunta "¿Cuál es tu evento?".

A continuación, sin necesidad de que el usuario pulse ningún botón, se abrirá la aplicación *Google Search* preparada para recibir la locución. Además, una vez ha rellenado estos dos campos, el sintetizador volverá a preguntar si se desea guardar el evento a lo que el usuario también puede contestar por voz con "Sí" o "No", lo que se traducirán en las mismas acciones que los botones "Guardar" y "Cancelar" respectivamente. Cabe destacar que si en esta última interacción lo entendido por la aplicación de reconocimiento de voz no coincide con la respuesta esperada, se informará del error al usuario a través del sintetizador con la locución "Lo siento, no te he entendido" y activando de nuevo la aplicación de reconocimiento de voz para que el usuario repita su respuesta.

4.2.3.2 Arquitectura y flujo de datos

La Figura 4.12 muestra la arquitectura y posible flujo de datos de la actividad Creación de eventos. Los pasos correspondientes a la interacción táctil-visual se muestran en negro y los correspondientes a la interacción multimodal se muestran en rojo.

Interacción táctil-visual

1) El usuario introduce los datos del evento y pulsa el botón "Guardar".
2) La aplicación almacena la información introducida por el usuario en la base de datos interna SQLite.
3) Se muestra por pantalla un mensaje de aviso con el texto "Evento guardado" para confirmar que la operación se ha realizado correctamente.

Interacción multimodal

1) El sintetizador de voz reproduce la locución "¿Cuál es tu evento?" y, acto seguido, se lanza automáticamente la actividad de reconocimiento de voz.
2) El usuario dicta la descripción del evento a la aplicación de reconocimiento de voz.
3) Se muestra por pantalla, escrito en los campos correspondientes del formulario, lo entendido por el reconocedor de voz. Al mismo tiempo, el sintetizador de voz solicita la conformidad del usuario a través de la locución "¿Quieres guardar el evento?". Acto seguido, se lanza automáticamente la actividad de reconocimiento de voz.
4) El usuario dice "Sí" para guardar o "No" para descartar (se saltan los pasos 5 y 6).
5) En caso afirmativo, la aplicación almacena la información introducida por el usuario en la base de datos interna SQLite.

6) El sintetizador de voz avisa al usuario de que la operación se ha realizado correctamente a través de la locución “Evento guardado”.

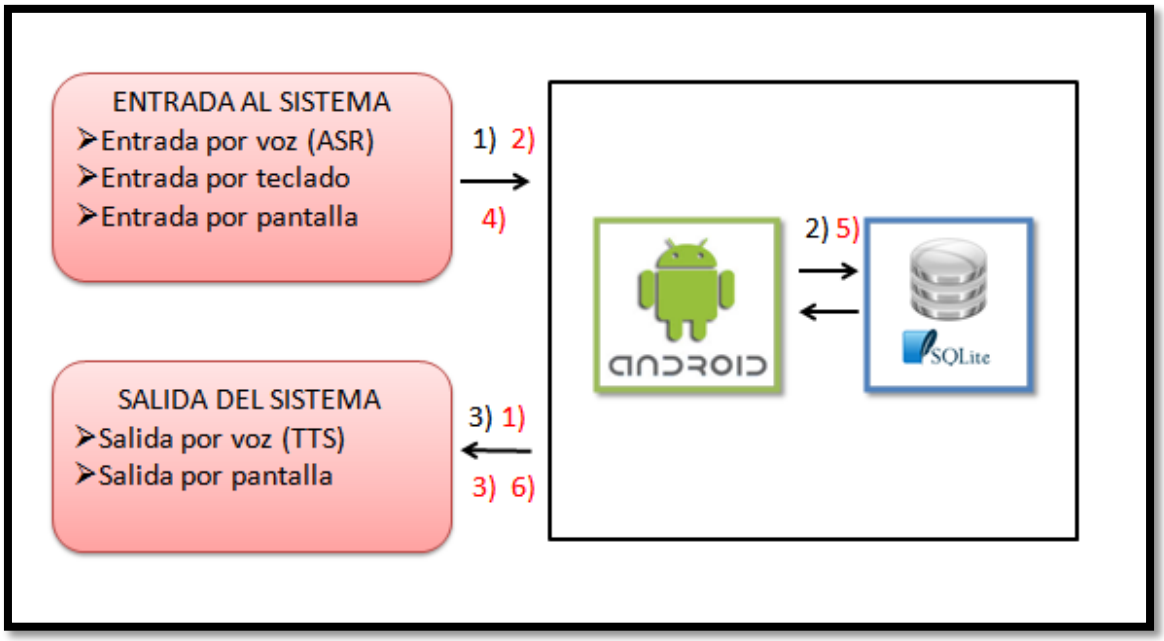


Figura 4.12: Arquitectura y flujo de datos de la actividad Creación de eventos

4.2.3.3 Escenario de uso

El usuario ha accedido a la pantalla de creación de un nuevo evento a través de la voz por lo que el sintetizador de voz se ha lanzado con la locución “¿Cuál es tu evento?”. A continuación, la aplicación *Google Search* se ejecuta automáticamente esperando el contenido del campo “Asunto” y, opcionalmente, la “Ubicación”, tal y como se muestra en la Figura 4.13.

Figura 4.13: Aplicación de reconocimiento de voz en la actividad *Creación de Eventos*

Una vez que la aplicación de reconocimiento de voz ha procesado lo dictado por el usuario, se vuelve a activar el sintetizador con la locución "¿Quieres guardar el evento?" y, seguidamente, se lanza de nuevo la aplicación *Google Search* esperando la confirmación como se observa en la Figura 4.14.

Figura 4.14: Aplicación de reconocimiento de voz en la actividad *Creación de Eventos*

Tal y como se indica en el mensaje de ayuda, el usuario debe contestar “Sí” para que dicho evento se guarde, almacenándose en la base de datos. Si esto ocurre, finalmente el sintetizador informará que la operación se ha realizado con éxito a través del mensaje “Evento guardado”.

4.2.2 Actividad de Edición

4.2.2.1 Funcionalidad

La funcionalidad de la presente actividad es la edición de un determinado evento ya almacenado en la aplicación. La Figura 4.15 muestra la pantalla de esta actividad.

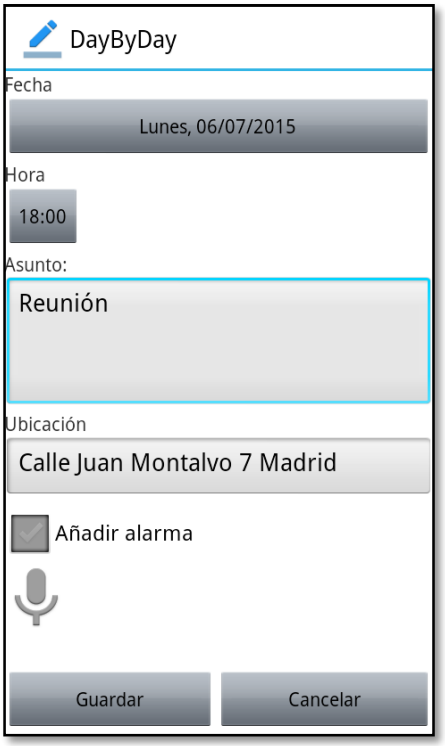


Figura 4.15: Captura de pantalla de la actividad *Edición de evento*

Como se puede observar, dicha pantalla es similar a la utilizada en la actividad *Creación de evento*, con la diferencia de que los campos del formulario aparecen rellenados por defecto con los valores antiguos introducidos. Todos los campos del formulario permiten interfaz táctil para la edición. Además, los campos “Asunto” y “Ubicación” permiten ser modificados por interfaz oral a través de la aplicación de reconocimiento de voz accesible desde el botón del micrófono. El usuario solamente tiene que dictar el asunto y, opcionalmente, la ubicación del evento tal y como se explicó en la sección 4.2.1 y la aplicación sobrescribirá automáticamente el contenido de los campos correspondientes. Una vez rellanados los campos necesarios (al menos el campo obligatorio “Asunto”), la aplicación permite guardar el evento pulsando en el

botón “Guardar”. Además, se puede descartar en cualquier momento pulsando el botón “Cancelar”.

A diferencia de la actividad Creación de Evento, no existe tal diálogo con la aplicación ya que a esta actividad se accede siempre a través de interfaz táctil.

4.2.2.2 Arquitectura y flujo de datos

La Figura 4.16 muestra la arquitectura y el flujo de datos de la actividad *Edición de evento*. Tal y como se observa en la imagen, el proceso general se puede resumir en los siguientes pasos:

1) La aplicación realiza una consulta a la base de datos interna SQLite para obtener la información del evento a editar.
2) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.
3) La aplicación muestra en la pantalla de edición los valores antiguos del evento a editar.
4) El usuario introduce los nuevos datos del evento, ya sea por interfaz táctil u oral a través de la aplicación de reconocimiento de voz, y pulsa el botón “Guardar”.
5) La aplicación almacena en la base de datos interna SQLite los nuevos valores del evento.
6) Se muestra por pantalla un mensaje de aviso con el texto “Evento guardado” para confirmar que la operación se ha realizado correctamente.

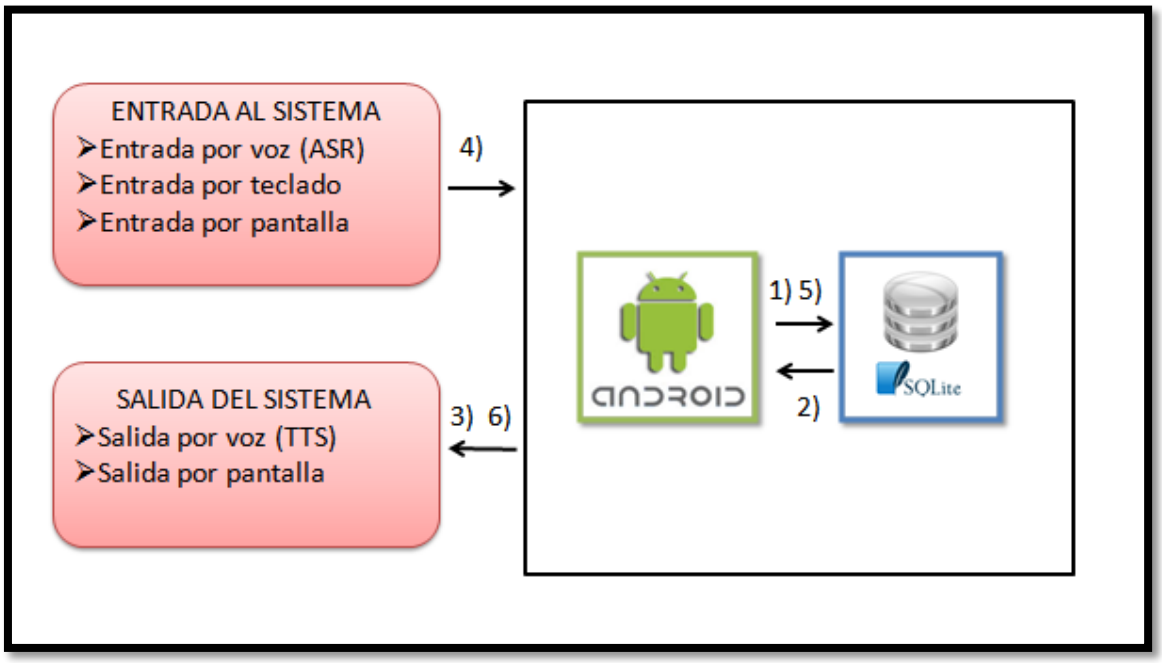
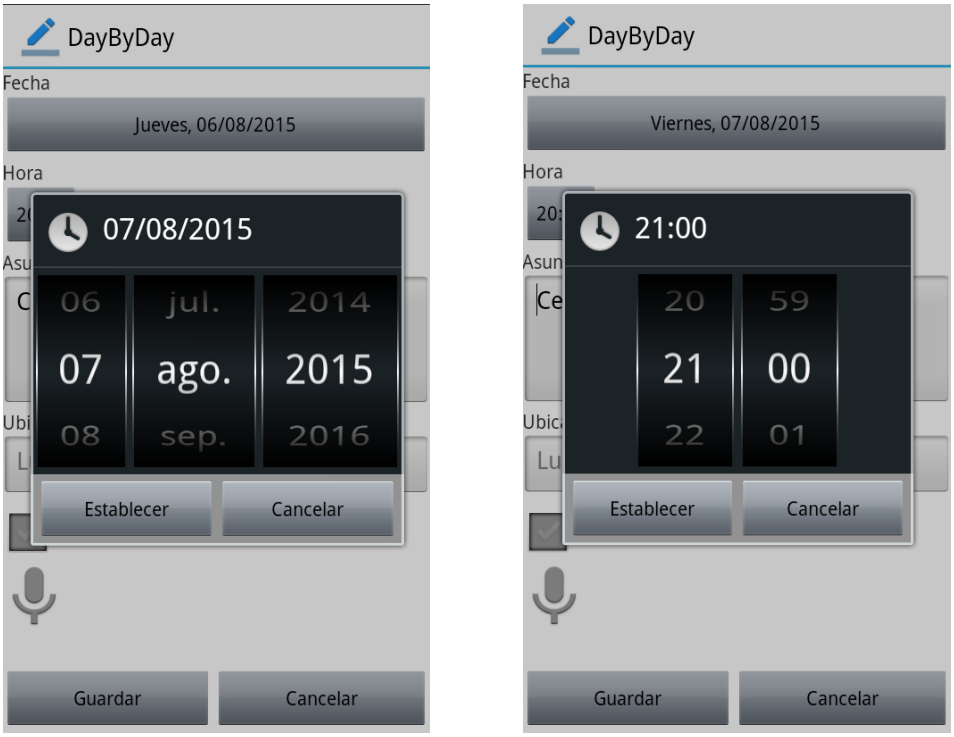


Figura 4.16: Arquitectura y flujo de datos de la actividad *Edición de evento*

4.2.2.3 Escenario de uso

El usuario accede a la pantalla de edición de un determinado evento y, en primer lugar, edita la fecha y la hora de dicho evento, tal y como se muestran en las Figura 4.17.



(a) Componente de edición de fecha (b) Componente de edición de hora

Figura 4.17: Captura de pantalla de la actividad *Edición de evento* (componentes de edición de fecha y hora)

A continuación, el usuario edita el contenido de los campos “Asunto” y “Ubicación” utilizando la aplicación de reconocimiento de voz, la cual se muestra en la Figura 4.18.

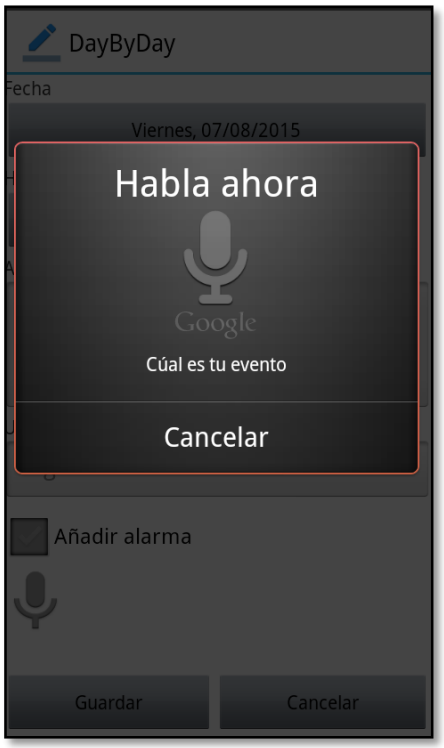


Figura 4.18: Aplicación de reconocimiento de voz en la actividad *Edición de Evento*

Finalmente, una vez que la aplicación de reconocimiento de voz ha procesado la expresión dictada por el usuario, se sobrescriben automáticamente los campos correspondientes quedando como resultado una pantalla similar a la mostrada en la Figura 4.19.

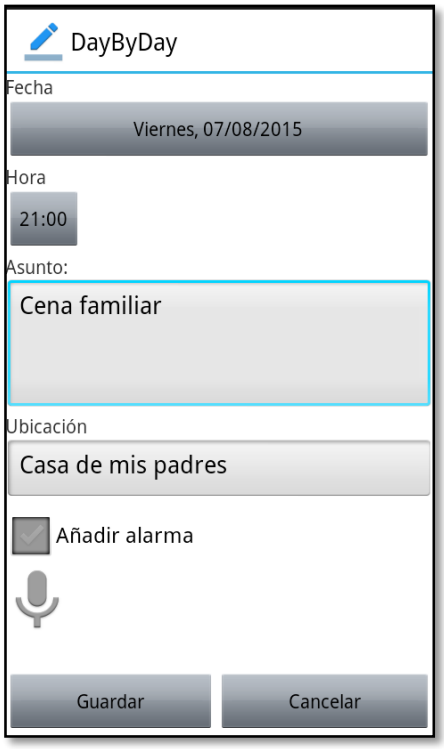


Figura 4.19: Captura de pantalla de la actividad *Edición de evento*

Llegado a este punto, el usuario debe pulsar el botón “Guardar” para que el evento se actualice en la base de datos.

4.2.3 Actividad de Consulta

4.2.3.1 Funcionalidad

Esta actividad permite al usuario consultar la información de un determinado evento existente en la aplicación. La Figura 4.20 muestra una pantalla típica de la presente actividad.

Figura 4.20: Captura de pantalla de la actividad *Consulta de evento*

Tal y como se observa, dicha pantalla muestra las datos que caracterizan al evento y que fueron introducidos por el usuario en su creación. Además, con el objetivo de enriquecer la información aportada por el usuario, en el caso de que la ubicación del evento se trate de una dirección existente, se muestra en la parte inferior de la pantalla un fragmento de un mapa indicando dicha localización.

Desde esta pantalla se puede tanto editar (accediendo a la actividad *Edición de evento*) como eliminar el presente evento a través de las opciones “Editar” y “Eliminar” respectivamente del menú de la presente pantalla (accesible desde el botón “Menú” del dispositivo).

4.2.3.2 Arquitectura y flujo de datos

La Figura 4.21 muestra la arquitectura y flujo de datos de la actividad *Consulta de evento*. Tal y como se observa en la imagen, el proceso general se puede resumir en los siguientes pasos:

1) La aplicación realiza una consulta a la base de datos interna SQLite para obtener la información del evento que se desea consultar.
2) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.
3) Se realiza una petición al servidor Google Maps con las coordenadas de la ubicación del evento para poder mostrarla en un mapa.
4) El servidor Google Maps devuelve el resultado de la petición a la aplicación.
5) Se muestra por pantalla la información del evento.

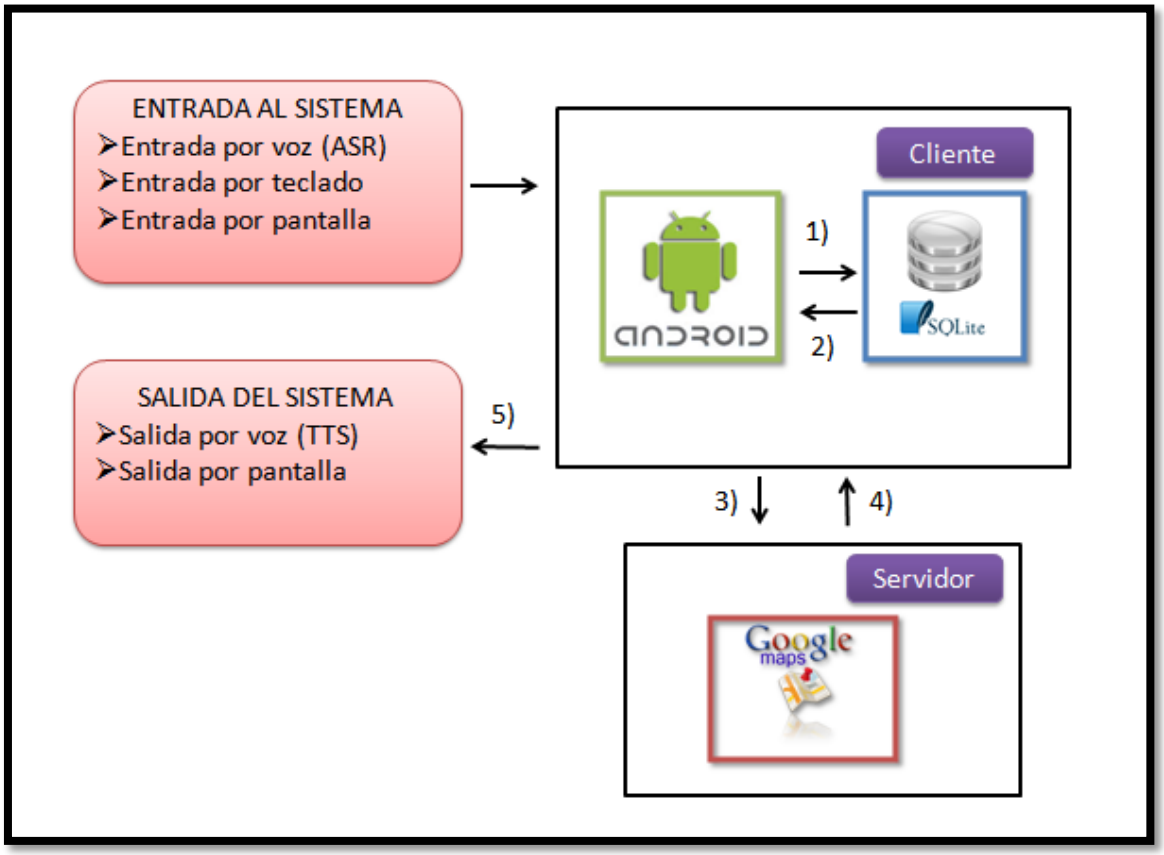


Figura 4.21: Arquitectura y flujo de datos de la actividad *Consulta de evento*

4.2.3.3 Escenario de uso

El escenario de uso de la presente actividad consta únicamente de la pantalla mostrada en la Figura 4.20, la cual ha sido anteriormente descrita en la sección 4.2.3.1.

4.3 Módulo de Notas

El módulo de Notas comprende todas aquellas acciones relacionadas con la información asociada a la categoría *Nota*. Este tipo de información está caracterizada por una fecha, el cuerpo de la nota y una categoría que describa la temática del contenido. El presente módulo puede dividirse en tres sub-módulos o actividades: ***Creación, Edición*** y ***Consulta.***

4.3.1 Actividad de Creación

4.3.1.1 Funcionalidad

La funcionalidad de la presente actividad es la creación de un nuevo ítem asociado a la categoría *Nota*. La Figura 4.22 muestra la pantalla de esta actividad.

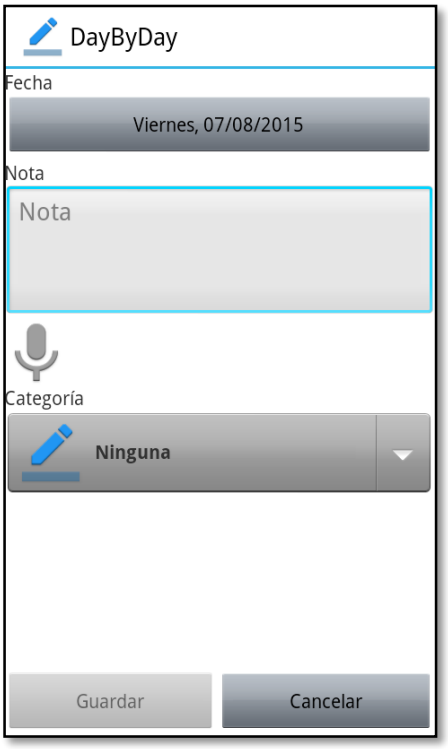


Figura 4.22: Captura de pantalla de la actividad *Creación de nota*

Como se puede observar, dicha pantalla consta de un formulario a rellenar con los datos de la posible nota: la fecha, el cuerpo de la nota y una categoría que describa la temática de la misma. El campo “Nota” se trata de un campo obligatorio ya que es el campo principal de este tipo de categoría y el que será utilizado por las base de datos para realizar búsquedas por palabra clave. Una vez que dicho campo se ha rellenado, la aplicación permite guardar la nota pulsando en el botón “Guardar”. La not0061 se puede descartar en cualquier momento pulsando el botón “Cancelar”.

Esta actividad permite rellenar el campo “Nota” a través de interfaz oral. Si se ha accedido a él por interfaz táctil, es posible acceder a la aplicación *Google Search* a través del botón del micrófono presente en la pantalla. Por otro lado, si se ha accedido a esta actividad por voz, al abrirse esta pantalla se lanzará automáticamente el sintetizador de texto a voz con la pregunta “¿Qué quieres anotar?”. A continuación, sin necesidad de que el usuario pulse ningún botón, se abrirá la aplicación *Google Search* preparada para recibir la locución.

Además, una vez se ha rellenado este campo, el sintetizador volverá a preguntar si se desea guardar la nota a lo que el usuario también puede contestar por voz con “Sí” o “No”, lo que tendrá como consecuencia las mismas acciones que los botones “Guardar” y “Cancelar” respectivamente. Cabe destacar que si en esta última interacción lo entendido por la aplicación de reconocimiento de voz no coincide con la respuesta esperada, se informará del error al usuario a través del sintetizador con la locución “Lo siento, no te he entendido” y activando de nuevo la aplicación de reconocimiento de voz para que el usuario repita su respuesta.

La categoría de la nota se puede seleccionar a través del menú desplegable que se encuentra en la presente pantalla. Este campo tiene como fin caracterizar con mayor detalle las notas guardadas en la aplicación. Además, es utilizado por la aplicación para aportar información adicional u ofrecer enlaces de interés relacionados con el contenido de la nota, como se explicará en la Sección 4.3.3. La Figura 4.23 muestra dicho menú en el que se pueden observar las posibles categorías que sugiere la aplicación.

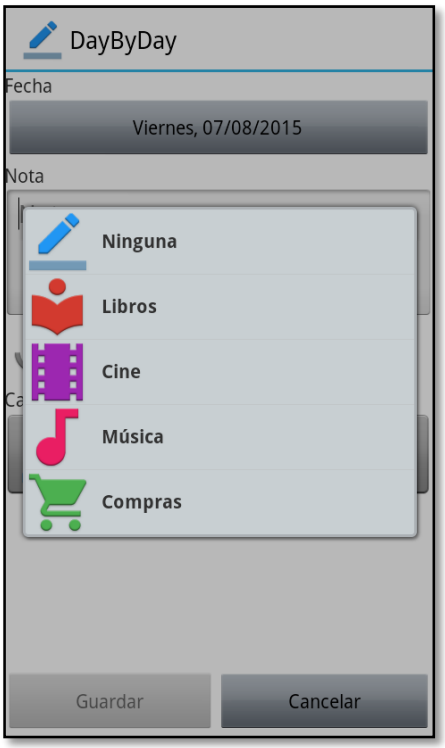


Figura 4.23: Opciones del menú “Categoría” de la actividad *Creación de nota*

4.3.1.2 Arquitectura y flujo de datos

La Figura 4.24 muestra la arquitectura y posible flujo de datos de la actividad *Creación de nota*. Los pasos correspondientes a la interacción táctil-visual se muestran en negro y los correspondientes a la interacción multimodal se muestran en rojo.

Interacción táctil-visual

1) El usuario introduce los datos de la nota y pulsa el botón “Guardar.
2) La aplicación almacena la información introducida por el usuario en la base de datos interna SQLite.
3) Se muestra por pantalla un mensaje de aviso con el texto “Nota guardada” para confirmar que la operación se ha realizado correctamente.

Interacción multimodal

1) El sintetizador de voz reproduce la locución “¿Qué quieres anotar?” y, acto seguido, se lanza automáticamente la actividad de reconocimiento de voz.
2) El usuario dicta el cuerpo de la nota a la aplicación de reconocimiento de voz.
3) Se muestra por pantalla, escrito en el campo correspondiente del formulario, lo entendido por el reconocedor de voz. Al mismo tiempo, el sintetizador de voz solicita la conformidad del usuario a través de la locución “¿Quieres guardar la nota?”. A continuación, se lanza de nuevo la actividad de reconocimiento de voz.
4) El usuario dice “Sí” para guardar o “No” para descartar (se saltan los pasos 5 y 6).
5) En caso afirmativo, la aplicación almacena la información introducida por el usuario en la base de datos interna SQLite.
6) El sintetizador de voz avisa al usuario de que la operación se ha realizado correctamente a través de la locución “Nota guardada”.

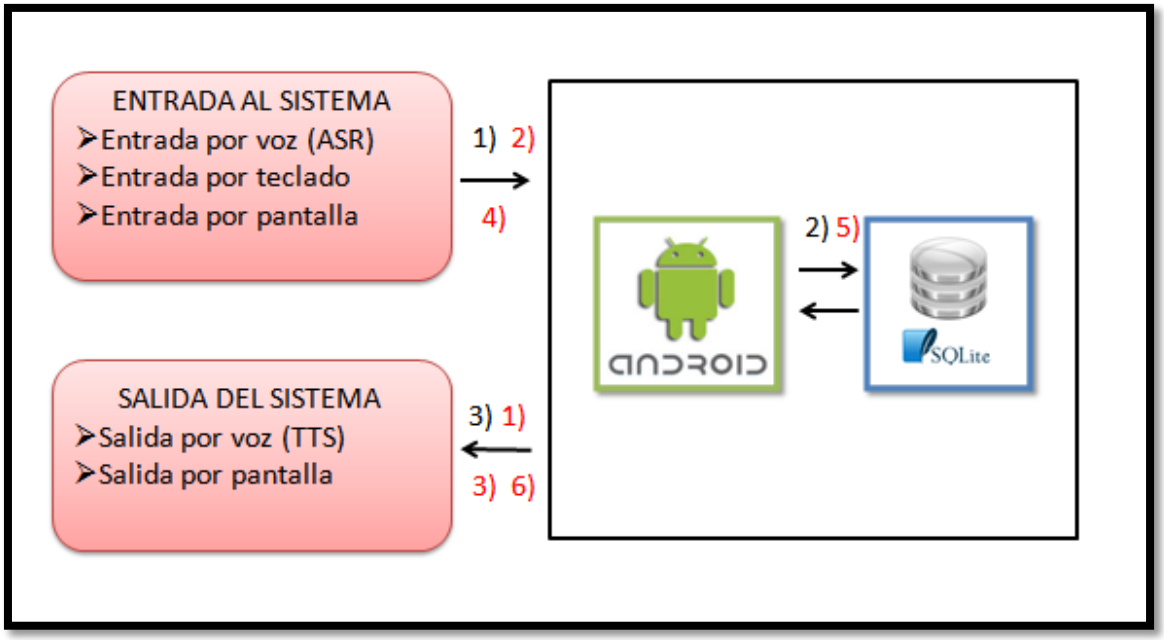


Figura 4.24: Arquitectura y flujo de datos de la actividad *Creación de nota*

4.3.1.3 Escenario de uso

El usuario ha accedido a la pantalla de creación de una nueva nota a través de la voz por lo que el sintetizador de voz se ha lanzado con la locución "¿Qué quieres anotar?". A continuación, la aplicación *Google Search* se abre automáticamente esperando el contenido del campo "Nota", tal y como se muestra en la Figura 4.25.

Figura 4.25: Aplicación de reconocimiento de voz en la actividad *Creación de nota*

Una vez que la aplicación de reconocimiento de voz ha procesado lo dictado por el usuario, se vuelve a activar el sintetizador con la locución "¿Quieres guardar la nota?" y, seguidamente, se lanza de nuevo la aplicación *Google Search* esperando la confirmación como se observa en la Figura 4.26.

Figura 4.26: Aplicación de reconocimiento de voz en la actividad *Creación de nota*

Tal y como se indica en el mensaje de ayuda, el usuario debe contestar "Sí" para que dicha nota se guarde, almacenándose en la base de datos. Si esto ocurre, finalmente el sintetizador informará de que la operación se ha realizado con éxito a través de la expresión "Nota guardada".

4.3.2 Actividad de Edición

4.3.2.1 Funcionalidad

La funcionalidad de la presente actividad es la edición de una determinada nota ya almacenada en la aplicación. La Figura 4.27 muestra la pantalla de esta actividad.

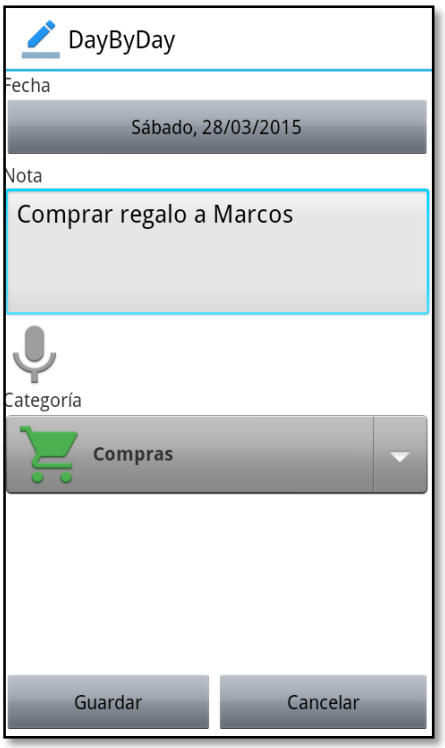


Figura 4.27: Captura de pantalla de la actividad *Edición de nota*

Como se puede observar, dicha pantalla es similar a la utilizada en la actividad *Creación de nota*, con la diferencia de que los campos del formulario aparecen rellenados por defecto con los valores antiguos introducidos. Todos los campos del formulario permiten interfaz táctil para la edición. Además, el campo "Nota" permite ser modificado por interfaz oral a través de la aplicación de reconocimiento de voz accesible desde el botón del micrófono. El usuario solamente tiene que dictar la nota y la aplicación sobrescribirá automáticamente el contenido de dicho campo. Una vez rellanados los campos necesarios (al menos el campo obligatorio "Nota"), la aplicación permite guardar la nota pulsando en el botón "Guardar". Además, se puede descartar en cualquier momento pulsando el botón "Cancelar".

A diferencia de la actividad *Creación de nota*, no existe tal diálogo con la aplicación ya que a esta actividad se accede siempre a través de interfaz táctil.

4.3.2.2 Arquitectura y flujo de datos

La Figura 4.28 muestra la arquitectura y el flujo de datos de la actividad *Edición de nota*. Tal y como se observa en la imagen, el proceso general se puede resumir en los siguientes pasos:

1) La aplicación realiza una consulta a la base de datos interna SQLite para obtener la información de la nota a editar.
2) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.

3) La aplicación muestra en la pantalla de edición los valores antiguos de la nota a editar.
4) El usuario introduce los nuevos datos de la nota, ya sea por interfaz táctil u oral a través de la aplicación de reconocimiento de voz, y pulsa el botón “Guardar”.
5) La aplicación almacena en la base de datos interna SQLite los nuevos valores de la nota.
6) Se muestra por pantalla un mensaje de aviso con el texto “Nota guardada” para confirmar que la operación se ha realizado correctamente.

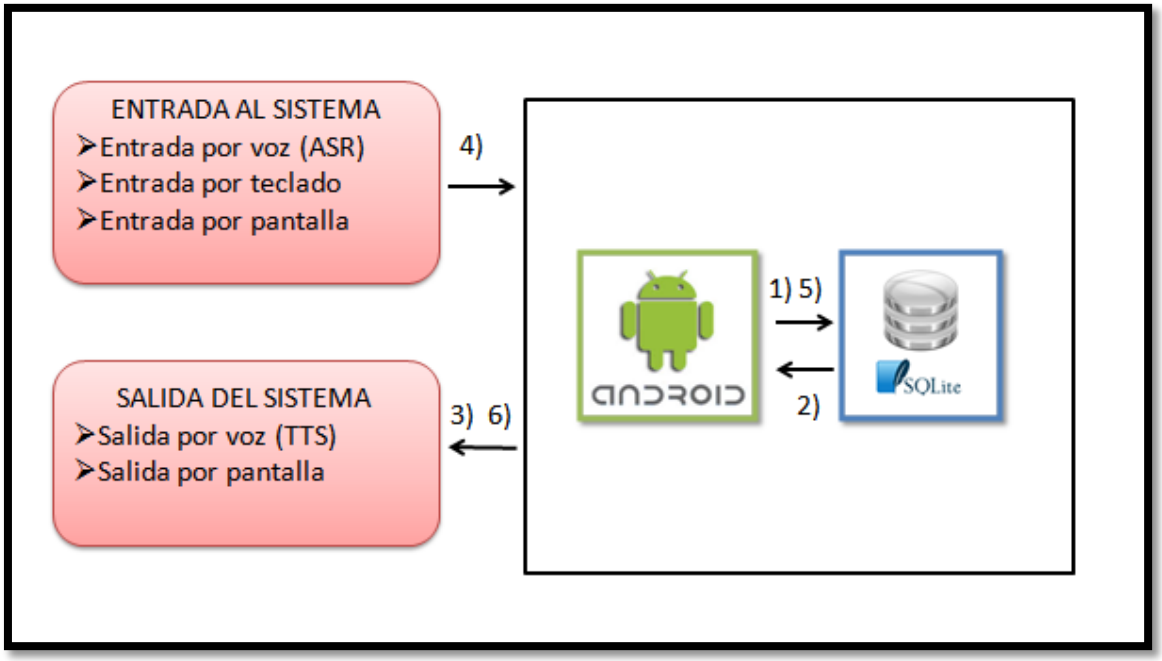


Figura 4.28: Arquitectura y flujo de datos de la actividad *Edición de nota*

4.3.2.3 Escenario de uso

El usuario accede a la pantalla de edición de una determinada nota y, en primer lugar, edita la fecha de la misma, tal y como se muestran en las Figura 4.29.

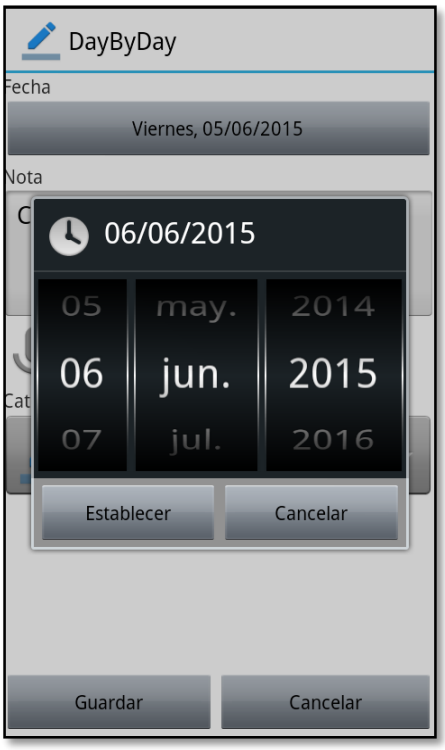


Figura 4.29: Captura de pantalla de la actividad *Edición de nota* (componente de edición de fecha)

A continuación, el usuario edita el contenido del campo “Nota” utilizando la actividad de reconocimiento de voz, la cual se muestra en la Figura 4.30.

Figura 4.30: Actividad de reconocimiento de voz en la actividad *Edición de nota*

Finalmente, una vez que la actividad de reconocimiento ha procesado la expresión dictada por el usuario, se sobrescriben automáticamente el campo correspondiente quedando como resultado una pantalla similar a la mostrada en la Figura 4.31.

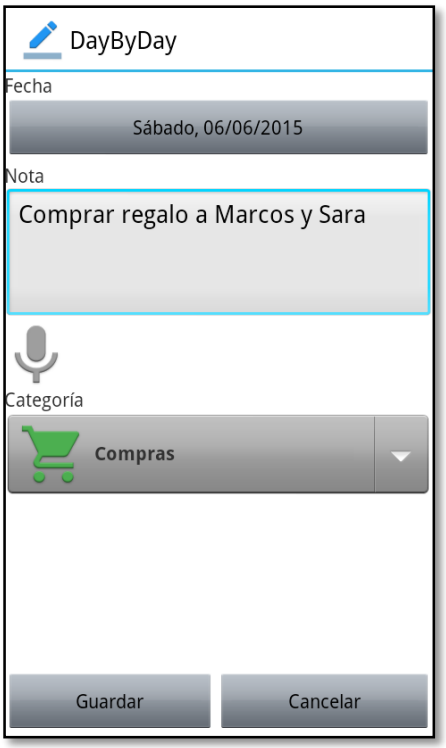


Figura 4.31: Captura de pantalla de la actividad *Edición de evento*

Llegado a este punto, el usuario debe pulsar el botón “Guardar” para que la nota se actualice en la base de datos.

4.3.3 Actividad de Consulta

4.3.3.1 Funcionalidad

Esta actividad permite al usuario consultar la información de una determinada nota existente en la aplicación. La Figura 4.32 muestra una pantalla típica de la presente actividad.

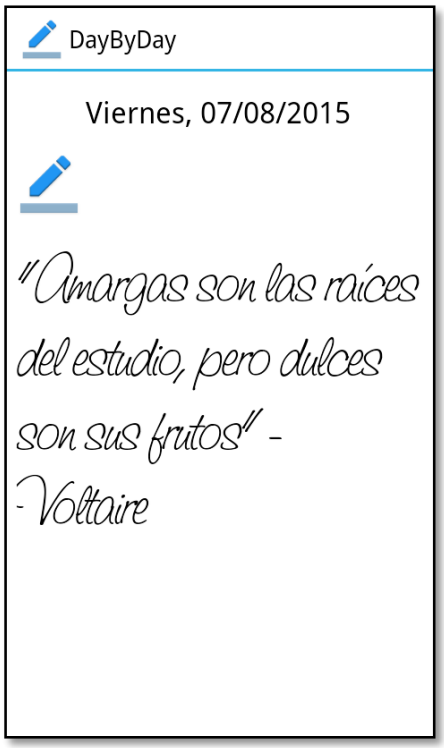


Figura 4.32: Captura de pantalla de la actividad *Consulta de nota*

Tal y como se observa, dicha pantalla muestra las datos que caracterizan la nota y que fueron introducidos por el usuario en su creación.

Desde esta pantalla se puede tanto editar la nota (accediendo, de esta manera, a la actividad *Edición de nota*) como eliminarla a través de las opciones "Editar" y "Eliminar" respectivamente del menú de la presente actividad (accesible desde el botón "Menú" del dispositivo).

Como se ha introducido en la sección 4.3.1, el campo "Categoría" del módulo *Notas* es utilizado por la aplicación para aportar información adicional u ofrecer enlaces de interés relacionados con el contenido de la nota. En concreto, esta funcionalidad está disponible para las categorías "Música", "Libros" y "Cine".

Consulta de nota asociada a la categoría *Música*

Cuando se consulta una nota cuya categoría es "Música" la aplicación ofrece, automáticamente, un enlace a la aplicación *Youtube*, instalada en la mayoría de los dispositivos, con una búsqueda lanzada con el texto de la nota. La Figura 4.33 muestra la pantalla de consulta de una nota para esta categoría y el resultado de pulsar el botón "Buscar".

Figura 4.33: Actividad *Consulta de nota* para la categoría música

Consulta de nota asociada a la categoría Libros

Cuando se consulta una nota cuya categoría es "Libros" la aplicación ofrece, automáticamente, un enlace a la página web de Casa del Libro (http://www.casadellibro.com/) con una búsqueda lanzada con el texto de la nota. La Figura 4.34 muestra la pantalla de consulta de una nota para esta categoría y el resultado de pulsar el botón "Buscar", respectivamente. Dado que la búsqueda se realiza a través de Internet, esta actividad devolverá un resultado siempre y cuando se disponga de conexión a Internet.

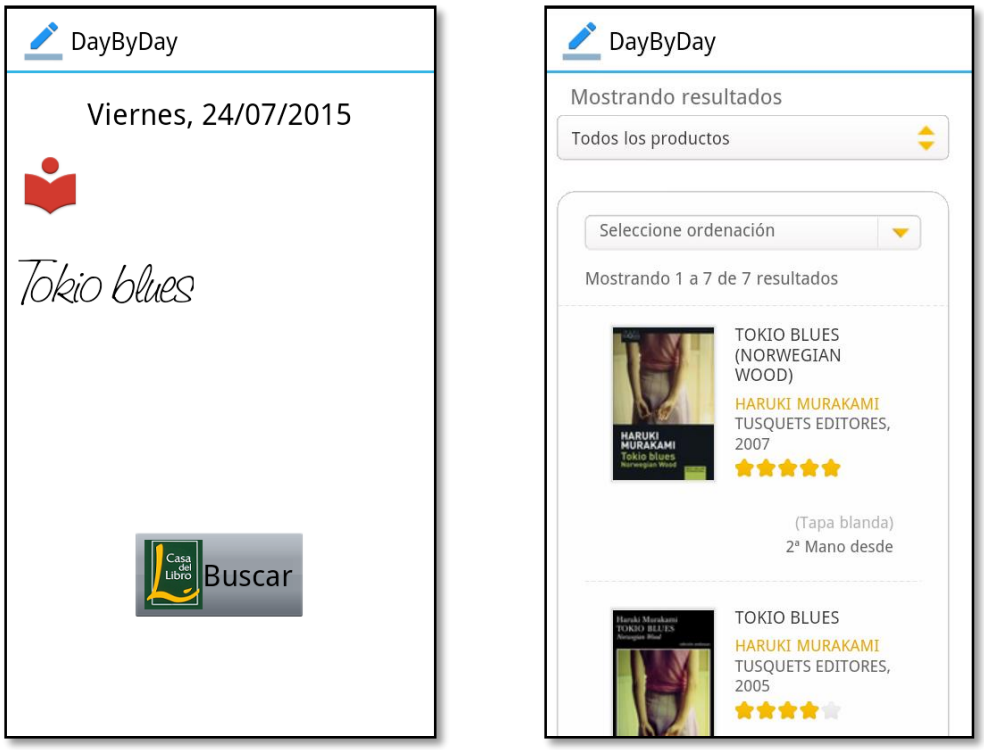
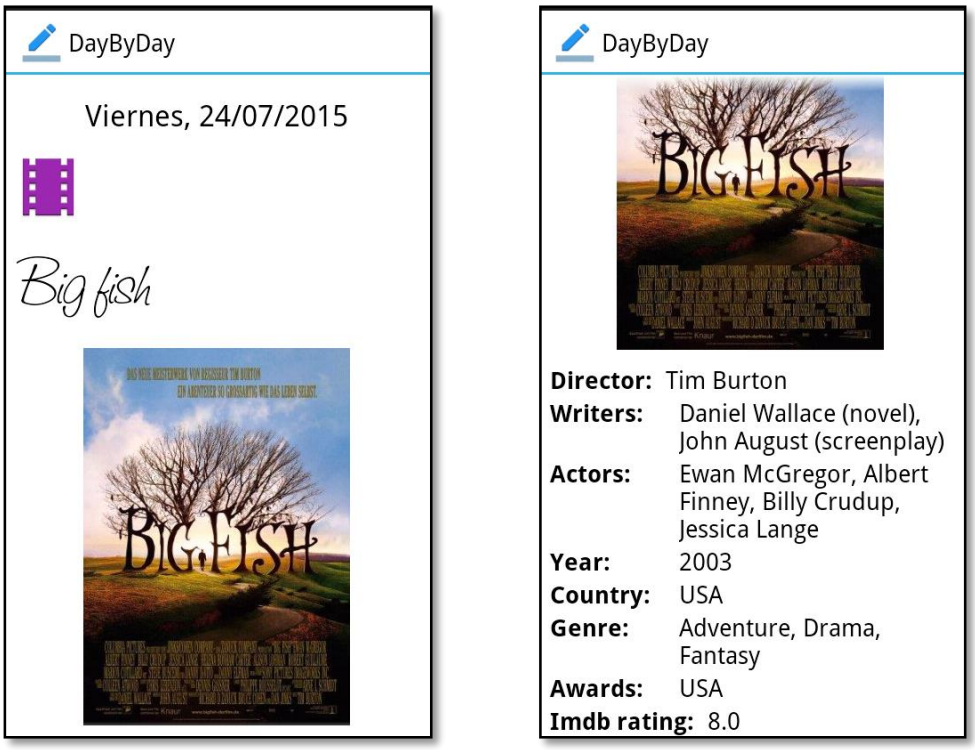


Figura 4.34: Actividad *Consulta de nota* para la categoría "Libros"

Consulta de nota asociada a la categoría Cine

Cuando se consulta una nota cuya categoría es "Cine" la aplicación realiza automáticamente una petición al servidor IMDb con el contenido de la nota. Si dicho contenido se trata del título de una película y está disponible en la base de datos del servidor, se muestra en pantalla información adicional sobre dicha película así como una imagen del póster de la misma. De esta forma, se ofrece información de interés al usuario al mismo tiempo que se decora la presente pantalla. La Figura 4.35 muestra la pantalla típica de una consulta a una nota asociada a la presente categoría.



(a) Parte superior de la pantalla (b) Parte inferior de la pantalla

Figura 4.35: Captura de pantalla de la actividad *Consulta de nota* para la categoría "Cine"

4.3.3.1 Arquitectura y flujo de datos

La Figura 4.36 muestra la arquitectura y posible flujo de datos de la actividad *Consulta de nota*. Dependiendo de la categoría de nota elegida, los casos que componen el flujo de datos difiere. Los casos correspondientes a las categorías "Libros" y "Cine" están marcados en rojo y azul respectivamente, los casos correspondientes al resto de categorías están marcados en negro.

Flujo de datos correspondiente a la categoría "Cine"

1) La aplicación realiza una consulta a la base de datos interna SQLite para obtener la información de la nota que se desea consultar.
2) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.
3) La aplicación realiza una petición al servidor web IMDb con el contenido de la nota.
4) El servidor devuelve el objeto JSON con la información solicitada.
5) La aplicación muestra por pantalla la información de la nota, tanto la introducida por el usuario como la recibida por el servidor.

Flujo de datos correspondiente a la categoría “Libros”

1) La aplicación realiza una consulta a la base de datos interna SQLite para obtener la información de la nota que se desea consultar.
2) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.
3) Se muestra por pantalla la información de la nota.
4) El usuario acciona la opción “Buscar” a través de interfaz táctil.
5) La aplicación realiza una petición al servidor web Casa del Libro con el contenido de la nota.
6) El servidor web Casa del Libro devuelve el resultado de la petición a la aplicación.
7) Se lanza una nueva actividad mostrando en pantalla la información recibida por el servidor.

Flujo de datos correspondiente al resto de categorías

1) La aplicación realiza una consulta a la base de datos interna SQLite para obtener la información de la nota que se desea consultar.
2) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.
3) Se muestra por pantalla la información de la nota.

Sólo para la categoría “Música”:

4) El usuario acciona la opción “Buscar” a través de interfaz táctil.
5) Se lanza la aplicación *Youtube* con la búsqueda correspondiente y se muestra al usuario.

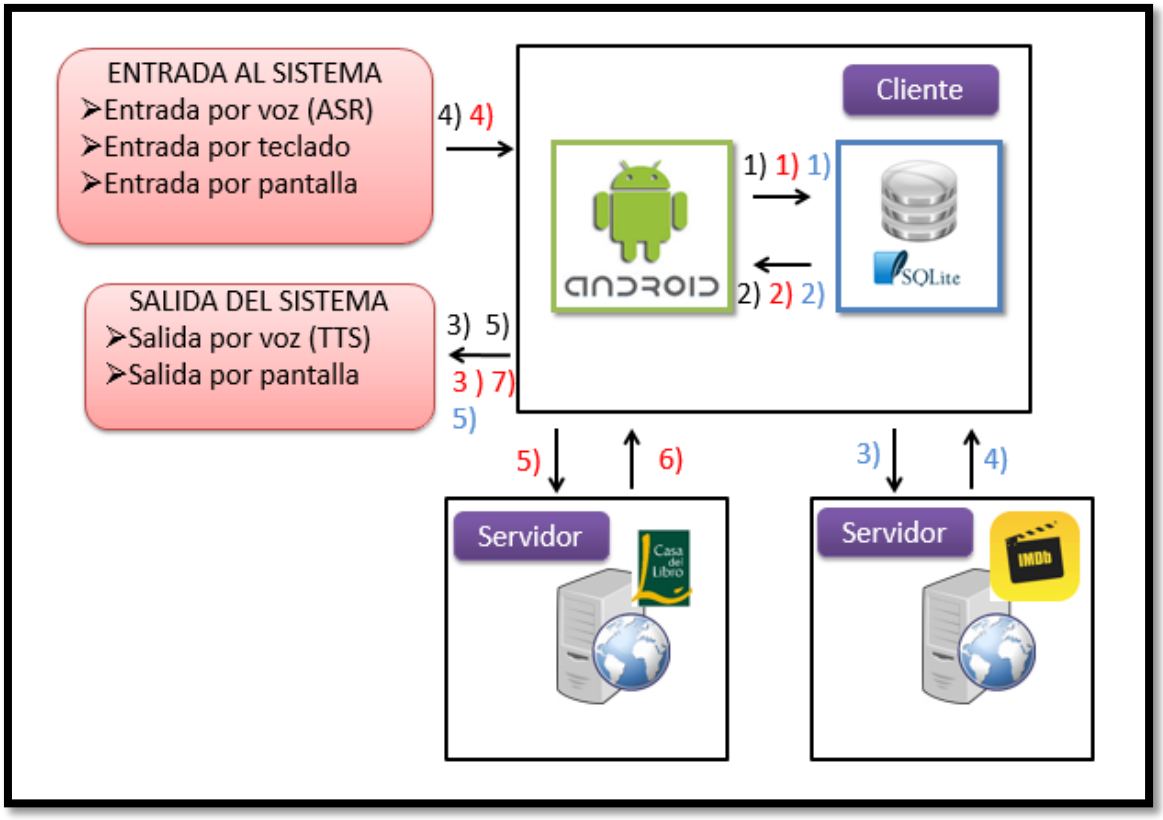


Figura 4.36: Arquitectura y flujo de datos de la actividad *Consulta de notas*

4.3.3.2 Escenario de uso

El usuario consulta una nota guardada en la aplicación cuya categoría es “Libros”. La Figura 4.37 muestra la pantalla que ve el usuario en esta primera interacción.

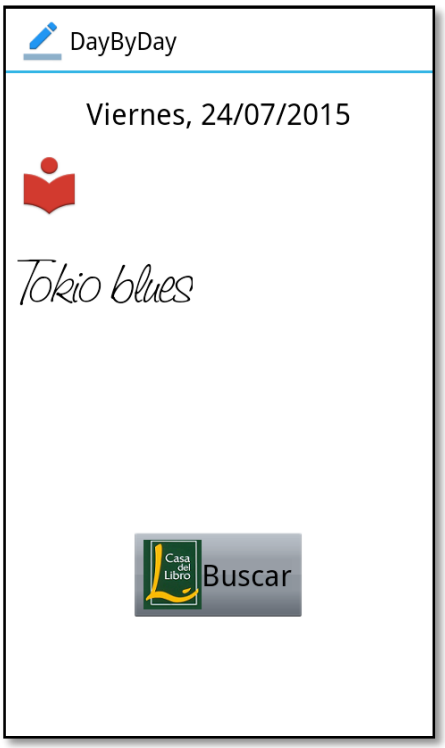


Figura 4.37: Captura de pantalla de la actividad *Consulta de nota* de la categoría "Libros"

El usuario pulsa el botón "Buscar" para obtener más información sobre el libro anotado. En la Figura 4.38 se muestra la actividad lanzada tras esta acción.

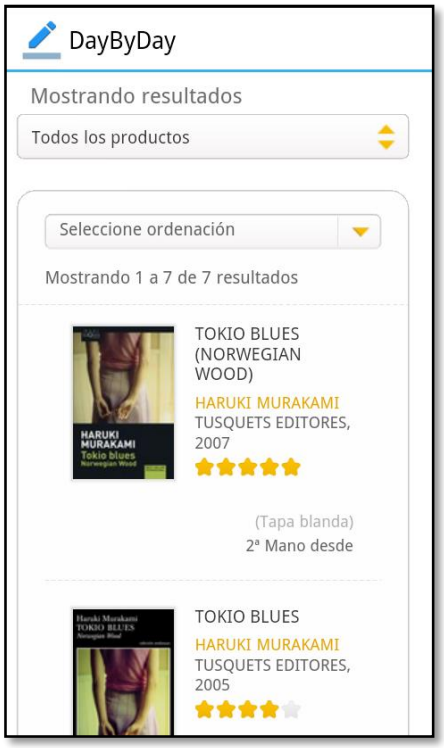


Figura 4.38: Actividad de búsqueda en la Casa del Libro

Además, si el usuario sigue navegando por esta actividad puede encontrar información de interés como datos y resumen del libro, recomendaciones, valoración según los usuarios, etc., tal y como se muestra en la Figura 4.39.

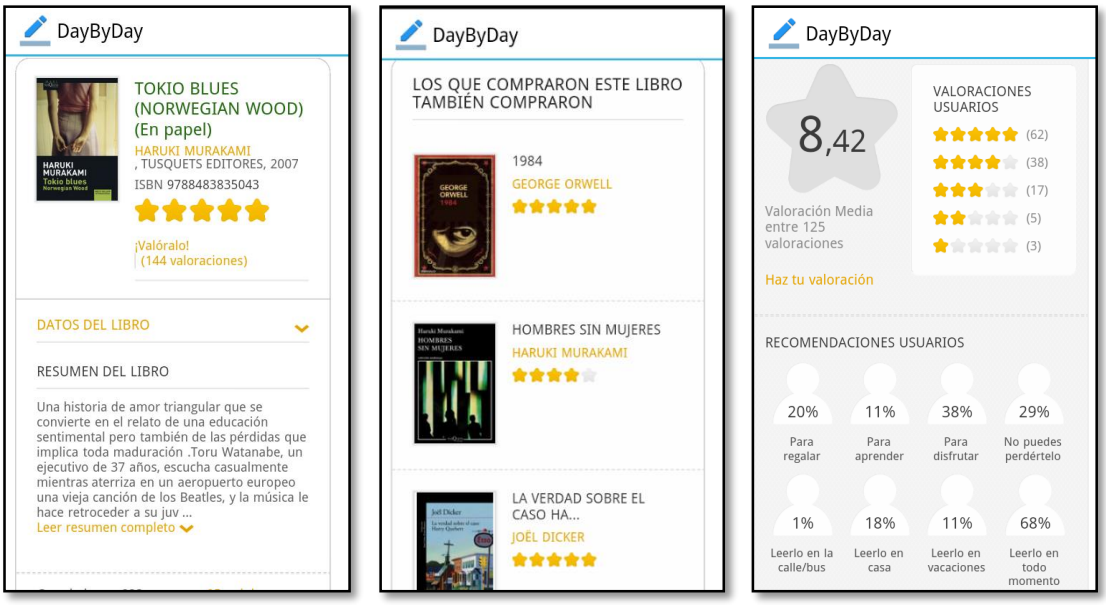


Figura 4.39: Pantallas ofrecidas por la actividad de búsqueda de libro.

4.4 Módulo de Lugares

El módulo de Lugares comprende todas aquellas acciones relacionadas con la información asociada a la categoría *Lugar*. Este tipo de información está caracterizada por la geolocalización del usuario, la fecha de creación del ítem y, opcionalmente, una breve descripción del lugar en forma de texto y/o audio. El presente módulo puede dividirse en tres sub-módulos o actividades: ***Creación, Edición*** y ***Consulta.***

4.4.1 Actividad de Creación

4.4.1.1 Funcionalidad

La funcionalidad de la presente actividad es la creación de un nuevo ítem asociado a la categoría *Lugar*. La Figura 4.40 muestra la pantalla de esta actividad.

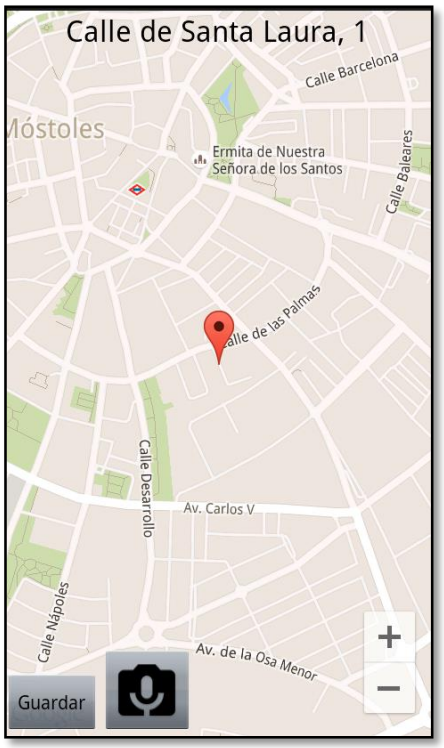


Figura 4.40: Captura de pantalla de la actividad *Creación de lugares*

Como se observa en la imagen, la aplicación muestra en un mapa la localización actual del usuario mostrando, además, la dirección completa (calle) de la misma. Esta actividad ofrece la posibilidad de adjuntar una grabación de audio que se asociará al ítem, a modo de explicación o recordatorio. Esta acción se realiza a través del botón simbolizado con un micrófono el cual abre una pantalla que permite realizar la grabación y escucharla al finalizar.

Si el usuario decide guardar la localización obtenida, debe pulsar el botón “Guardar” el cual lanza un diálogo para permitir al usuario añadir una nota descriptiva al lugar antes de ser almacenado y cuyo texto será utilizado por la aplicación para realizar búsquedas por palabra clave. Por otro lado, el usuario puede descartar en cualquier momento la creación del nuevo ítem a través del botón “Atrás” del dispositivo.

Cabe destacar que, dado que el proveedor de localización utilizado en esta actividad es el dispositivo GPS, este debe encontrarse previamente activado para obtener dicha la localización. En caso contrario, la aplicación avisa al usuario, a través de un mensaje en pantalla, que el GPS no se encuentra disponible.

4.4.1.2 Arquitectura y flujo de datos

La Figura 4.41 muestra la arquitectura y flujo de datos de la actividad *Creación de lugar*. Tal y como se observa en la imagen, el proceso general se puede resumir en los siguientes pasos:

1) Se realiza una petición al servidor Google Maps con las coordenadas actuales del dispositivo, obtenidas a partir del proveedor de localización (GPS).
2) El servidor Google Maps devuelve el resultado de la petición a la aplicación.
3) Se muestra por pantalla el mapa con la localización del usuario.
4) El usuario adjunta la grabación de audio (opcional) y pulsa el botón guardar para añadir la nota descriptiva.
5) Se almacena el fichero de audio obtenido en el paso 4) en la memoria externa del dispositivo.
6) La aplicación almacena la información introducida por el usuario en la base de datos interna SQLite.
7) Se muestra por pantalla un mensaje de aviso con el texto "Ubicación guardada" para confirmar que la operación se ha realizado correctamente.

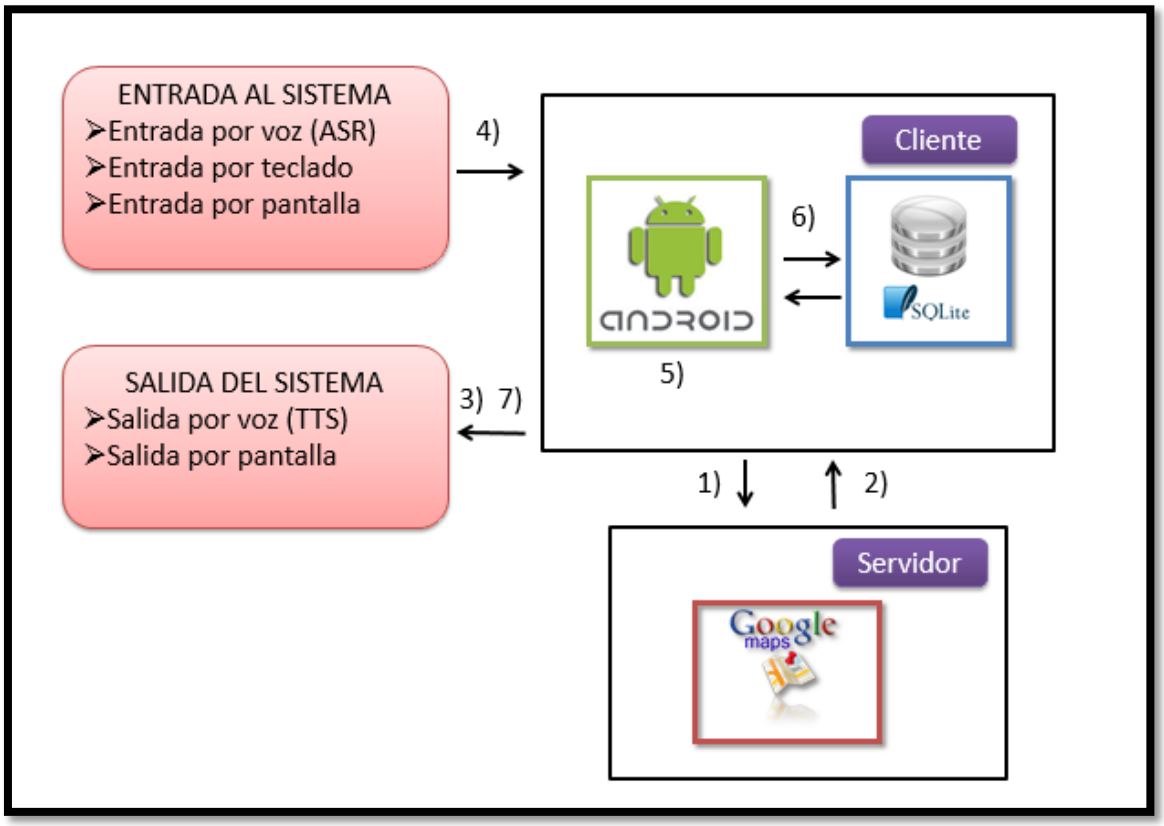


Figura 4.41: Arquitectura y flujo de datos de la actividad *Creación de lugar*

4.4.1.3 Escenario de uso

El usuario accede a la pantalla de creación de un nuevo lugar a través de interfaz táctil u oral desde el módulo principal. Previamente, ha activado el GPS del dispositivo desde el menú de configuración del mismo. La primera pantalla que el usuario se encuentra en la presente actividad se muestra en la Figura 4.42.

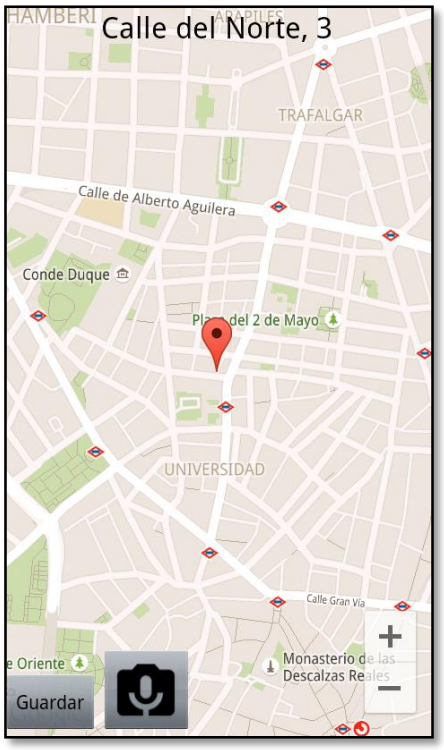


Figura 4.42: Pantalla principal de la actividad *Creación de lugar*

A continuación, el usuario pulsa el botón que abre la actividad de grabación de audio. Esta pantalla, la cual se muestra en la Figura 4.43, consta de dos botones: uno para iniciar/parar la grabación y otro para iniciar/parar la escucha del fichero de audio una vez grabado.

(a) Controles antes de iniciar la grabación

(b) Controles después de iniciar la grabación

Figura 4.43: Pantalla de la actividad para la grabación de audio en el Actividad *Creación de lugar*

Una vez grabado el audio, el usuario pulsa el botón “Guardar” para adjuntar el fichero de audio a la nueva ubicación, volviendo a la pantalla principal de esta actividad.

Finalmente, el usuario pulsa la opción “Guardar” para guardar el nuevo ítem. Antes de que el ítem sea almacenado, la aplicación abre una nueva pantalla la cual permite introducir una nota descriptiva del lugar que se asocie al nuevo ítem, tal y como se muestra en la Figura 4.44.

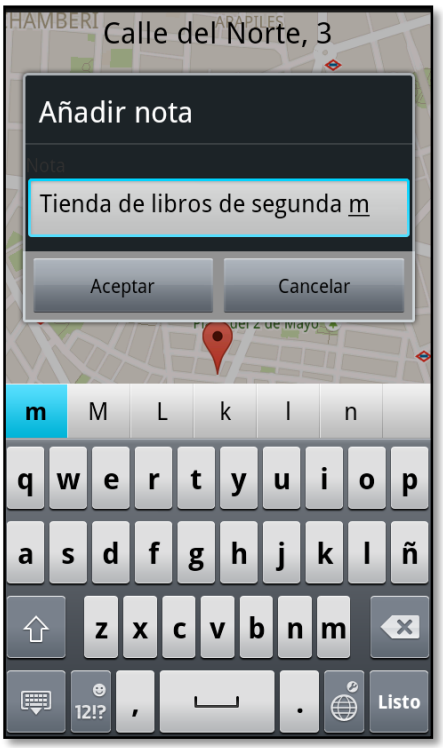


Figura 4.44: Pantalla de la actividad para adjuntar una nota en el Actividad *Creación de lugar*

A continuación, el usuario pulsa la opción “Aceptar” y, finalmente, el nuevo ítem es almacenado en la base de datos de la aplicación.

4.4.2 Actividad de Edición

4.4.2.1 Funcionalidad

La funcionalidad de la presente actividad es la edición de un determinado ítem de la categoría “Lugar” ya almacenada en la aplicación. De este tipo de ítem, lo único que puede ser modificado una vez creado y guardado es la nota descriptiva por lo que la pantalla de edición se trata de una pequeña ventana con un solo cuadro de texto, similar a la utilizada en la actividad *Creación de lugar.* La Figura 4.45 muestra la pantalla de la presente actividad.

Figura 4.45: Pantalla de la actividad *Edición de lugar*

El cuadro de texto de la presente pantalla muestra por defecto la nota antigua asociada al ítem. Como se observa en la imagen, se accede a esta actividad a través de la pantalla del módulo principal, realizando una pulsación larga en el ítem que se desea editar.

4.4.2.2 Arquitectura y flujo de datos

La Figura 4.46 muestra la arquitectura y el flujo de datos de la actividad *Edición de lugar*. Tal y como se observa en la imagen, el proceso general se puede resumir en los siguientes pasos:

1) La aplicación realiza una consulta a la base de datos interna SQLite para obtener la nota del ítem de tipo “Lugar” a editar.
2) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.
3) La aplicación muestra en la pantalla de edición el contenido antiguo de la nota a editar.
4) El usuario introduce el nuevo texto y pulsa el botón “Aceptar”.
5) La aplicación almacena en la base de datos interna SQLite la nueva nota del ítem.

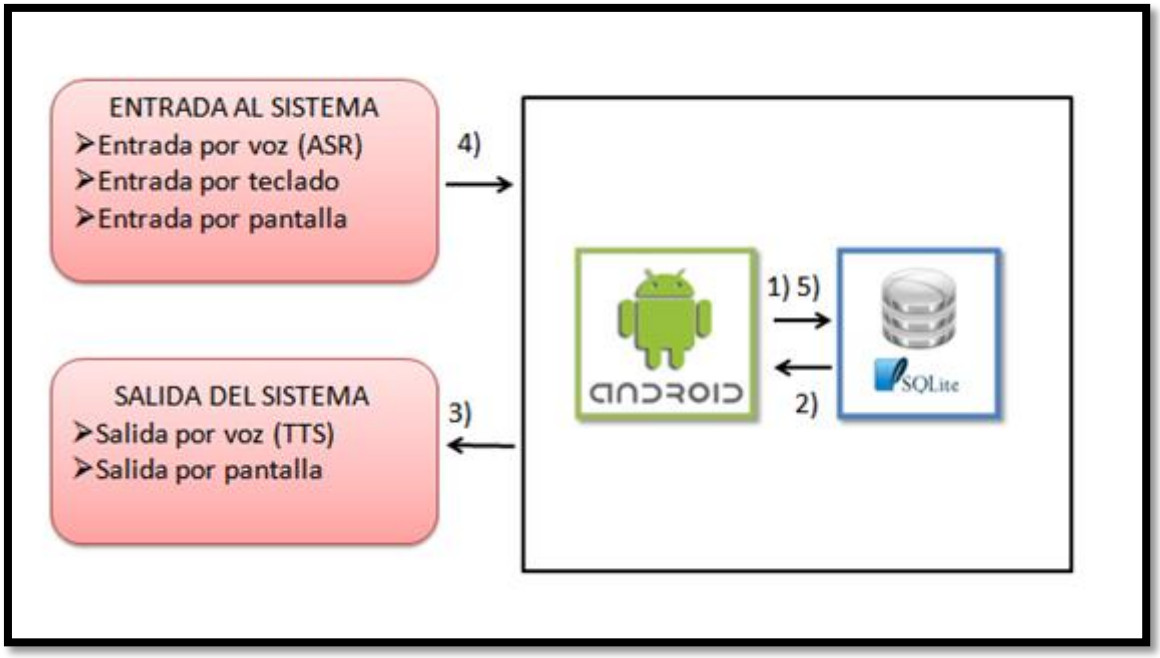


Figura 4.46: Arquitectura y flujo de datos de la actividad *Edición de lugar*

4.4.2.1 Escenario de uso

El usuario accede a la pantalla de edición de una de la ubicaciones guardadas en la aplicación a través de un una pulsación larga sobre el ítem a editar y eligiendo la opción “Editar” en el menú que aparece a continuación. Dicha pantalla de edición se muestra en la Figura 4.47.

Figura 4.47: Capturas de pantalla de la actividad *Edición de lugar*

Una vez que el usuario ha escrito el nuevo texto de la nota debe pulsar “Aceptar” para guardar la nueva información del ítem. Cuando se realiza dicha acción, la aplicación cierra esta actividad y se actualiza automáticamente el texto descriptivo del ítem en la lista de la pantalla principal, tal y como se muestra en la Figura 4.48.

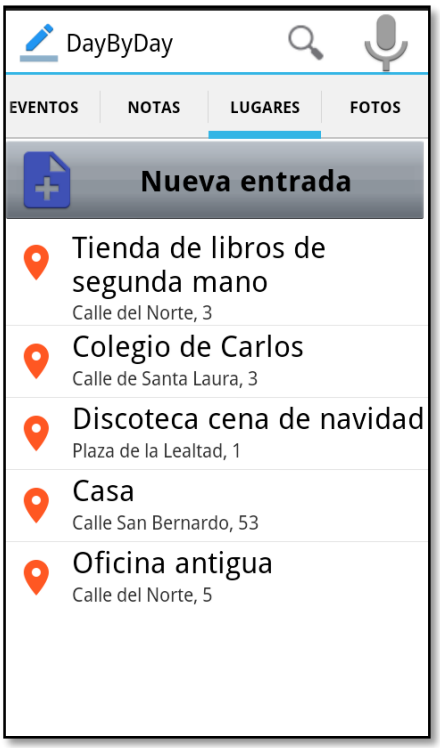


Figura 4.48: Pantalla principal de la aplicación una vez que se ha editado el ítem seleccionado

4.4.3 Actividad de Consulta

4.4.3.1 Funcionalidad

Esta actividad permite al usuario consultar la información de un determinado ítem de tipo *Lugar* existente en la aplicación. La Figura 4.49 muestra una pantalla típica de la presente actividad.

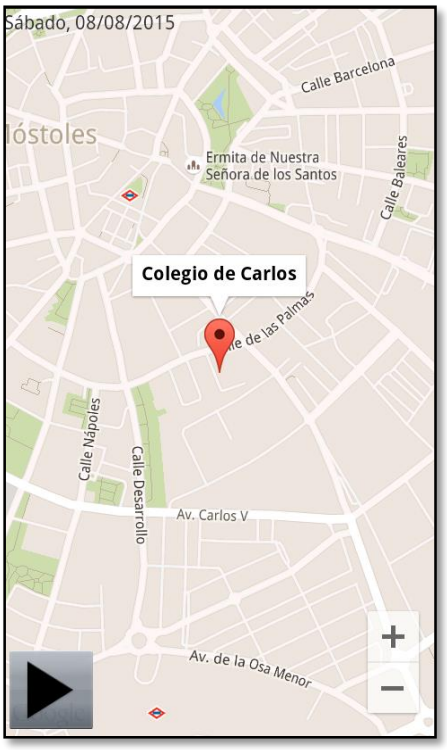


Figura 4.49: Captura de pantalla de la actividad *Consulta de lugar*

Tal y como se observa en la figura, se muestra dibujado en un mapa la geolocalización asociada al ítem. Además, se muestra la fecha de creación en la esquina superior izquierda y la nota descriptiva, que aparece al pulsar la marca en el mapa. Si el ítem contiene algún fichero de audio adjunto, se muestra un botón en la parte inferior de la pantalla que permite escuchar dicho audio.

4.4.3.2 Arquitectura y flujo de datos

La Figura 4.50 muestra la arquitectura y el flujo de datos de la actividad *Consulta de lugar*. Tal y como se observa en la imagen, el proceso general se puede resumir en los siguientes pasos:

1) La aplicación realiza una consulta a la base de datos interna SQLite para obtener la información del ítem *Lugar* que se desea consultar.
2) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.
3) Se realiza una petición al servidor Google Maps con las coordenadas de la ubicación guardada para poder mostrarla en un mapa.
4) El servidor Google Maps devuelve el resultado de la petición a la aplicación.
5) Se muestra por pantalla la ubicación pintada en un mapa, así como la demás información asociada al ítem.

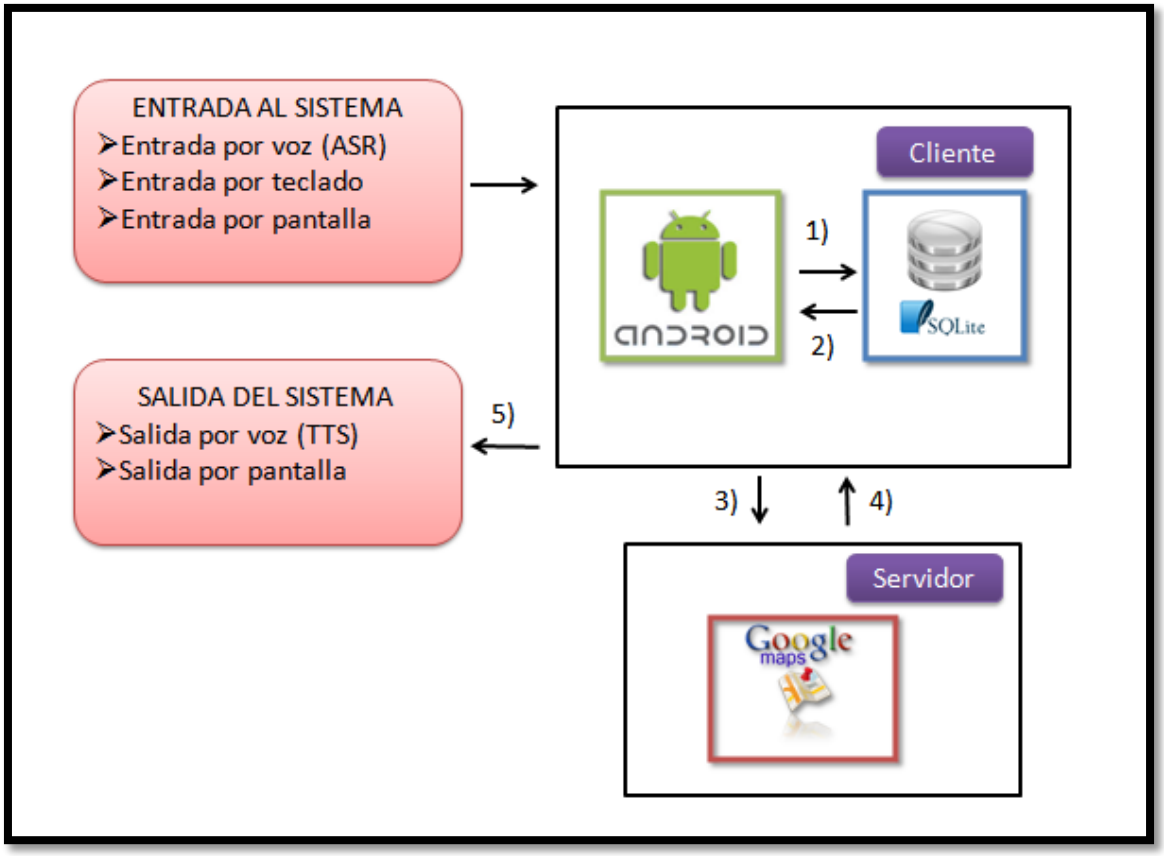


Figura 4.50: Arquitectura y flujo de datos de la actividad *Consulta de lugar*

4.4.3.3 Escenario de uso

El escenario de uso de la presente actividad consta únicamente de la pantalla mostrada en la Figura 4.49, la cual ha sido anteriormente descrita en la Sección 4.4.3.1.

4.5 Módulo de Fotos

El módulo de Fotos comprende todas aquellas acciones relacionadas con la información asociada a la categoría *Foto*. Este tipo de información está caracterizada por la foto creada por el usuario, la fecha actual y una breve descripción de la imagen a modo de nota. El presente módulo puede dividirse en tres sub-módulos o actividades: ***Creación, Edición*** y ***Consulta.***

4.5.1 Actividad de Creación

4.5.1.1 Funcionalidad

La funcionalidad de la presente actividad es la creación de un nuevo ítem asociada a la categoría *Foto*. En primer lugar, se accede a la aplicación de la cámara del dispositivo desde la pantalla principal (Nueva entrada -> Imagen o por voz con el comando "Crear foto") para tomar la foto. A continuación, la aplicación abre automáticamente una pantalla en la que se puede ver la imagen que se acaba de capturar, además de ofrecer la opción de añadir un texto descriptivo a la imagen antes de ser guardada. La Figura 4.51 muestra dicha pantalla.

Figura 4.51: Captura de pantalla de la actividad *Creación de foto*

El texto del campo "Descripción" es el utilizado por la aplicación para realizar búsquedas por palabra clave de este tipo de ítems. El cuadro de texto para dicho campo puede ser rellenado tanto por interfaz táctil (teclado) como oral. A esta última opción se accede desde el botón simbolizado con un micrófono el cual es el encargado de lanzar la actividad de reconocimiento de voz.

Por último, si el usuario desea guardar la imagen en la base de datos de la aplicación, debe pulsar el botón "Guardar". En caso contrario, se puede descartar pulsando la opción "Cancelar".

Cabe destacar que todas las imágenes realizadas por la aplicación se guardan en un mismo directorio propio de la aplicación, de forma que estén más organizadas y

accesibles para el usuario. Esta carpeta está ubicada en la galería del dispositivo y tiene el mismo nombre que la presente aplicación.

4.5.1.2 Arquitectura y flujo de datos

La Figura 4.52 muestra la arquitectura y el flujo de datos de la actividad *Creación de foto*. Tal y como se observa en la imagen, el proceso general se puede resumir en los siguientes pasos:

1) La aplicación lanza la actividad de la cámara del dispositivo.
2) El usuario toma la fotografía con la interfaz habitual del dispositivo para realizar esta acción.
3) La aplicación muestra en la pantalla de creación la fotografía tomada.
4) El usuario rellena el campo "Descripción", ya sea por interfaz táctil (teclado) como oral a través de la actividad de reconocimiento de voz. A continuación, pulsa el botón guardar.
5) La aplicación almacena la información introducida por el usuario así como la ruta de la fotografía en la memoria del teléfono en la base de datos interna SQLite.
6) Se muestra por pantalla un mensaje de aviso con el texto "Imagen guardada" para confirmar que la operación se ha realizado correctamente.

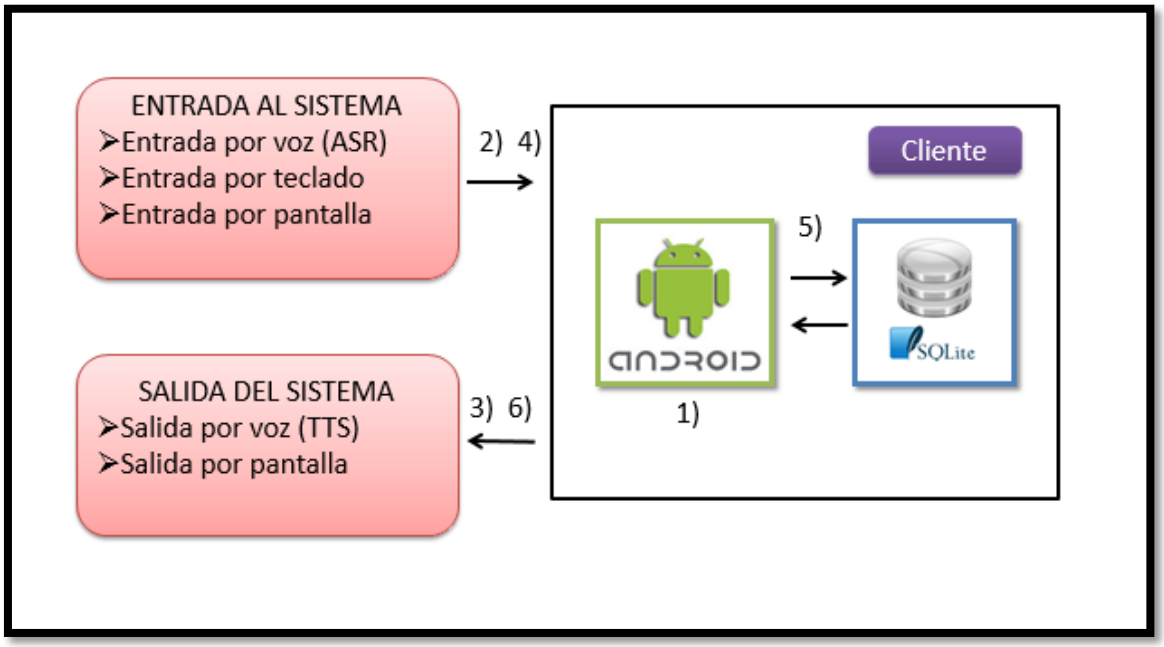


Figura 4.52: Arquitectura y flujo de datos de la actividad *Creación de foto*

4.5.1.3 Escenario de uso

El usuario accede a la actividad de la cámara del dispositivo desde la pantalla principal (Nueva entrada -> Imagen) para tomar la foto. Una vez capturada, accede automáticamente a la pantalla de creación, la cual se muestra en la Figura 4.53.

Figura 4.53: Captura de pantalla de la actividad *Creación de foto*

En esta pantalla, el usuario añade una descripción a la imagen a través de interfaz oral. Para ello, accede a la actividad de reconocimiento de voz a través del botón correspondiente. Dicha actividad y el resultado se muestran en la Figura 4.54.

Figura 4.54: Actividad de reconocimiento de voz y resultado del reconocimiento en la actividad *Creación de foto*

Una vez que el usuario ha añadido la descripción, pulsa el botón “Guardar” para almacenar el nuevo ítem en la base de datos de la aplicación.

4.5.2 Actividad de Edición

4.5.2.1 Funcionalidad

La funcionalidad de la presente actividad es la edición de un determinado ítem *Foto* ya almacenado en la aplicación. En concreto, el dato editable de este tipo de categoría es la descripción. La Figura 4.55 muestra la pantalla de esta actividad.

Figura 4.55: Captura de pantalla de la actividad *Edición de foto*

Como se puede observar, dicha pantalla es similar a la utilizada en la actividad *Creación de foto*, con la diferencia de que el campo "Descripción" aparece rellenado por defecto con el valor antiguo introducido. Dicho campo se puede editar a través de interfaz táctil (teclado) o a través de la actividad de reconocimiento de voz, accesible desde el botón representado con un micrófono.

Finalmente, si el usuario decide actualizar los cambios en el ítem seleccionado debe pulsar el botón "Guardar". Por el contrario, puede descartar los cambios seleccionando la opción "Cancelar".

4.5.2.2 Arquitectura y flujo de datos

La Figura 4.56 muestra la arquitectura y el flujo de datos de la actividad *Edición de foto*. Tal y como se observa en la imagen, el proceso general se puede resumir en los siguientes pasos:

1) La aplicación realiza una consulta a la base de datos interna SQLite para obtener la información de la foto a editar.
2) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.
3) La aplicación muestra en la pantalla de edición la imagen seleccionada así como los valores antiguos de la misma.

4) El usuario introduce la nueva descripción de la imagen, ya sea por interfaz táctil u oral a través de la aplicación de reconocimiento de voz, y pulsa el botón "Guardar".
5) La aplicación almacena en la base de datos interna SQLite la nueva descripción de la foto.
6) Se muestra por pantalla un mensaje de aviso con el texto "Imagen guardada" para confirmar que la operación se ha realizado correctamente.

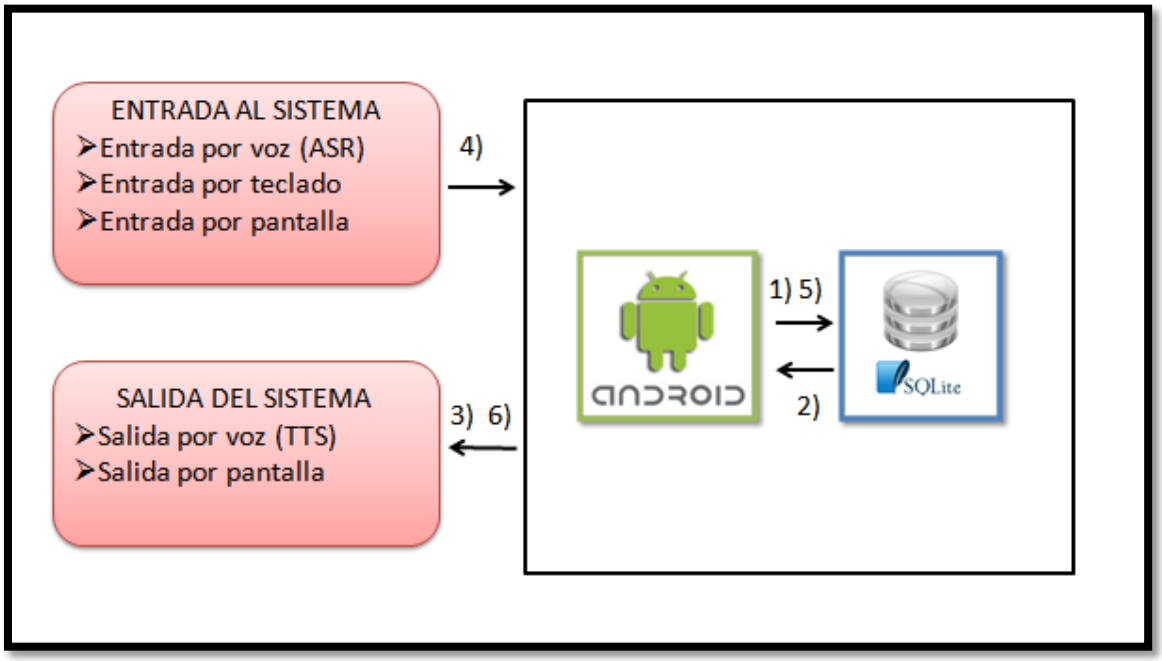


Figura 4.56: Arquitectura y flujo de datos de la actividad *Edición de foto*

4.5.2.3 Escenario de uso

El usuario accede a la pantalla de edición de una determinada imagen guardada en la aplicación. Dicha pantalla se muestra en la sección anterior, en la Figura 4.55.

A continuación, accede a la actividad de reconocimiento de voz para dictar la nueva descripción de la imagen. Una vez que el reconocimiento de voz ha procesado la locución del usuario, el resultado se escribe automáticamente en el campo "Descripción". Dicha aplicación y el resultado se muestran en la Figura 4.57.

Figura 4.57: Actividad de reconocimiento de voz y resultado del reconocimiento en la actividad *Creación de foto*

Finalmente, se pulsa la opción “Guardar” de forma que la aplicación actualiza el valor correspondiente en la base de datos interna.

4.5.3 Actividad de Consulta

4.5.3.1 Funcionalidad

Esta actividad permite al usuario consultar la información de un determinado ítem de tipo *Foto* existente en la aplicación. La Figura 4.58 muestra una pantalla típica de la presente actividad.

Figura 4.58: Captura de pantalla de la actividad *Consulta de foto*

Desde esta pantalla se puede tanto editar la descripción de la imagen (accediendo, de esta manera, a la actividad *Edición de foto*) como eliminarla a través de las opciones "Editar" y "Eliminar" respectivamente del menú de la presente actividad (accesible desde el botón "Menú" del dispositivo).

Además, pulsando sobre la imagen se accede a la ubicación de ésta en la galería. Como se ha explicado en la Sección 4.5.1.1, todas las imágenes creadas por la presente aplicación se encuentran en un mismo directorio de forma que si se accede a una de ellas a través de esta actividad, es únicamente necesario deslizar el dedo para ver las demás fotografías creadas por la aplicación.

4.5.3.2 Arquitectura y flujo de datos

La Figura 4.59 muestra la arquitectura y el flujo de datos de la actividad *Consulta de foto*. Tal y como se observa en la imagen, el proceso general se puede resumir en los siguientes pasos:

1) La aplicación realiza una consulta a la base de datos interna SQLite para obtener la información del ítem *Foto* que se desea consultar.
2) La base de datos SQLite devuelve el resultado de la consulta a la aplicación.
3) Se muestra por pantalla la imagen junto con su información asociada.

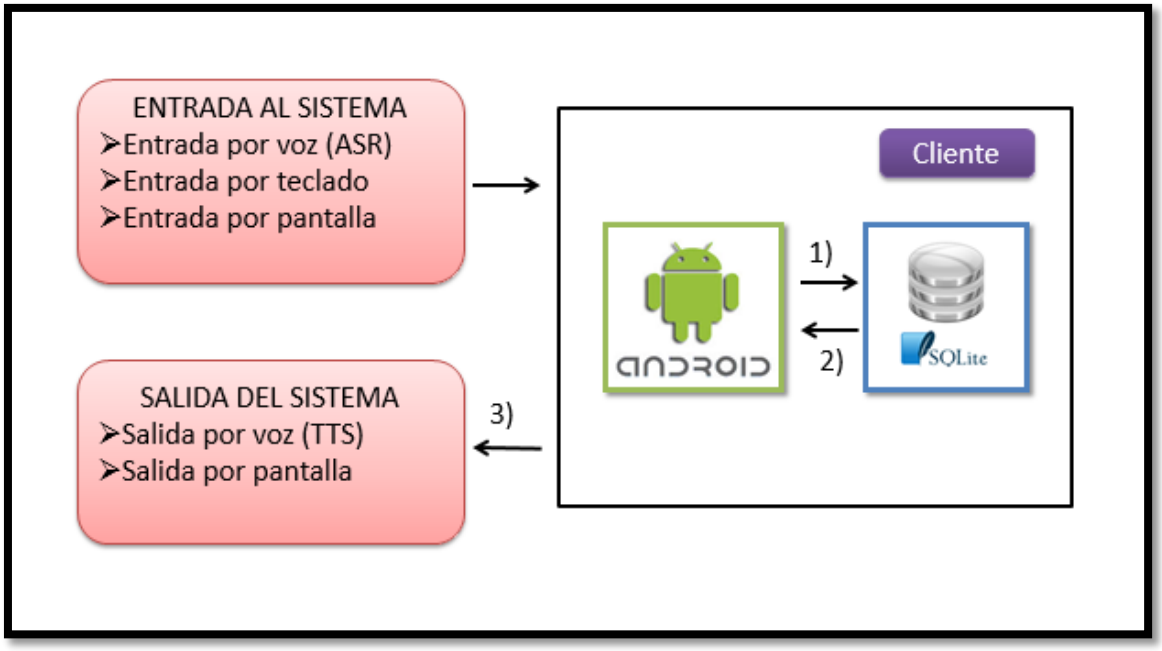


Figura 4.59: Arquitectura y flujo de datos de la actividad *Consulta de foto*

4.5.3.3 Escenario de uso

El escenario de uso de la presente actividad consta únicamente de la pantalla mostrada en la Figura 4.58, la cual ha sido anteriormente descrita en la sección 4.5.3.1.

Capítulo 5

Evaluación del sistema

En el presente capítulo de describe la metodología utilizada en la evaluación de la aplicación multimodal para dispositivos móviles desarrollada en el presente Trabajo de Fin de Grado. Posteriormente, se analizan los resultados obtenidos en la evaluación y las principales conclusiones del estudio.

5.1 Metodología de evaluación

La metodología utilizada para evaluar la aplicación multimodal desarrollada en el presente Trabajo de Fin de Grado consta de una valoración de calidad por parte de los usuarios. Para ello, se ha realizado un cuestionario que recoge la opinión subjetiva y el grado de satisfacción de los usuarios, obteniendo así una evaluación cualitativa del sistema por parte de los mismos.

Los aspectos que se han querido analizar en dicho cuestionario son: el nivel de dificultad a la hora de interactuar con el sistema, la presencia de errores, la velocidad percibida durante la interacción, la seguridad de lo que se debe hacer en cada momento, el grado en el cual el usuario valora que es entendido y entiende los mensajes del mismo y el nivel de satisfacción con el sistema en general. Además, se ha analizado el perfil de cada usuario con respecto a su conocimiento o experiencia previa en el manejo de sistemas multimodales, ya que se asume que la experiencia en el uso de interfaces táctiles es por lo general alto en los usuarios que han realizado la encuesta.

La Figura 5.1 muestra el cuestionario realizado para la evaluación. Como se observa, consta de 13 preguntas con 5 respuestas posibles cada una, de las que sólo se puede seleccionar una. El cuestionario está dividido en dos bloques, uno con preguntas dirigidas sólo al uso del sistema a través de interfaz táctil y otro dirigido al uso del sistema a través de interfaz multimodal. De esta manera, se puede realizar una comparación de la percepción del sistema por parte de los usuarios dependiendo del tipo de interfaz que se esté utilizando.

1. Puntúe en una escala del 1 al 5 su experiencia previa usando interfaces de voz o multimodales (1= “Bajo”, 5=”Alto”)
    - 1
    - 2
    - 3
    - 4
    - 5

    En lo que respecta únicamente a la interacción táctil con la aplicación:

2. Establezca el nivel de dificultad del sistema en modo táctil para usted.
    - Muy difícil
    - Difícil
    - Normal
    - Fácil
    - Muy fácil

3. ¿Ha percibido errores durante su interacción táctil con el sistema?
    - Sí, he percibido muchos errores lo que ha imposibilitado la interacción con el sistema.
    - Sí, he percibido bastantes errores lo que ha causado gran dificultad en la interacción con el sistema.
    - Sí, he percibido algunos errores lo que ha causado cierta dificultad en la interacción con el sistema.
    - Sí, he percibido algunos errores aunque la interacción con el sistema no se ha visto afectada en absoluto.
    - No he percibido errores.

4. En su opinión la interacción fue…
    - Muy lenta
    - Lenta
    - Aceptable
    - Rápida
    - Muy rápida

5. ¿Le ha sido fácil decidir qué hacer en cada momento para realizar la acción que usted solicitaba durante la interacción táctil?
    - No, ha sido imposible.
    - No, me ha supuesto gran dificultad.
    - Sí, pero con ciertas dificultades.
    - Sí, ha sido fácil.
    - Sí, ha sido muy fácil.

6. En términos generales, ¿está usted satisfecho con el sistema en modo táctil?
    - No, nada.
    - Poco satisfecho.
    - Satisfecho.
    - Bastante satisfecho.
    - Muy satisfecho.

    En lo que respecta a la interacción multimodal con la aplicación:

7. ¿Qué tal ha entendido el sistema sus peticiones?
    - Muy mal
    - Mal
    - Regular
    - Bien
    - Muy bien

8. ¿Cómo ha entendido los mensajes generados por el sistema?
    - Muy mal
    - Mal
    - Regular
    - Bien
    - Muy bien

9. Establezca el nivel de dificultad del sistema en modo oral para usted.
    - Muy difícil
    - Difícil
    - Normal
    - Fácil
    - Muy fácil

10. ¿Ha percibido errores durante su interacción oral con el sistema?
    - Sí, he percibido muchos errores lo que ha imposibilitado la interacción con el sistema.
    - Sí, he percibido bastantes errores lo que ha causado gran dificultad en la interacción con el sistema
    - Sí, he percibido algunos errores lo que ha causado cierta dificultad en la interacción con el sistema.
    - Sí, he percibido algunos errores aunque la interacción con el sistema no se ha visto afectada en absoluto.
    - No he percibido errores.

11. En su opinión la interacción fue…
    - Muy lenta
    - Lenta
    - Aceptable

- Rápida
- Muy rápida

12. ¿Le ha sido fácil decidir qué hacer en cada momento para realizar la acción que usted solicitaba durante la interacción oral?
    - No, ha sido imposible.
    - No, me ha supuesto gran dificultad.
    - Sí, pero con ciertas dificultades.
    - Sí, ha sido fácil.
    - Sí, ha sido muy fácil.

13. En términos generales, ¿está usted satisfecho con el sistema multimodal?
    - No, nada.
    - Poco satisfecho.
    - Satisfecho.
    - Bastante satisfecho.
    - Muy satisfecho.

Figura 5.1: Cuestionario de evaluación de la aplicación DayByDay

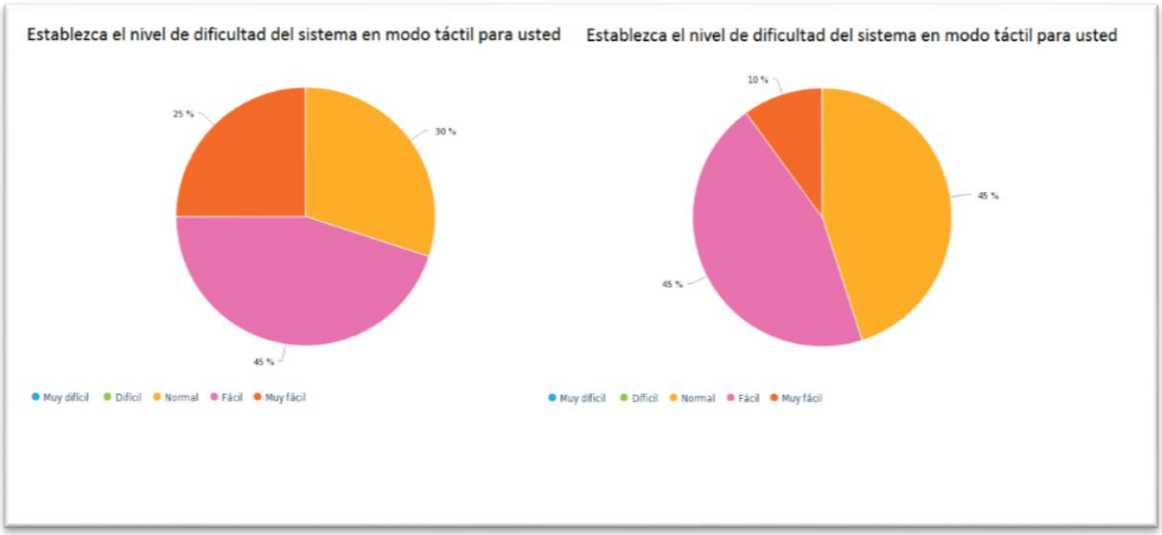
Para facilitar el registro de las opiniones de los usuarios y la recopilación de respuestas, se ha creado un cuestionario online a través del sitio web *Survio* [35], disponible en el enlace https://www.survio.com/survey/d/L7N2I3P2X3N8A5N6U.

5.1 Resultados de la evaluación

El cuestionario recoge el grado de satisfacción de 20 usuarios, 15 hispanoparlantes y 5 angloparlantes, a los que previamente se les había indicado que hicieran uso de todas las funcionalidades del sistema con todas la interfaces permitidas.

En la Figura 5.2 se muestra las estadísticas de los resultados para cada una de las cuestiones que se plantean en el cuestionario, agrupando aquellas que analicen la misma característica para cada tipo de interfaz con el objetivo de facilitar la comparación entre las dos modalidades (apareciendo a la derecha la correspondiente a la interfaz táctil y a la izquierda la correspondiente a la interfaz multimodal).

Valor mínimo	Valor máximo	Valor medio
1	5	2,7

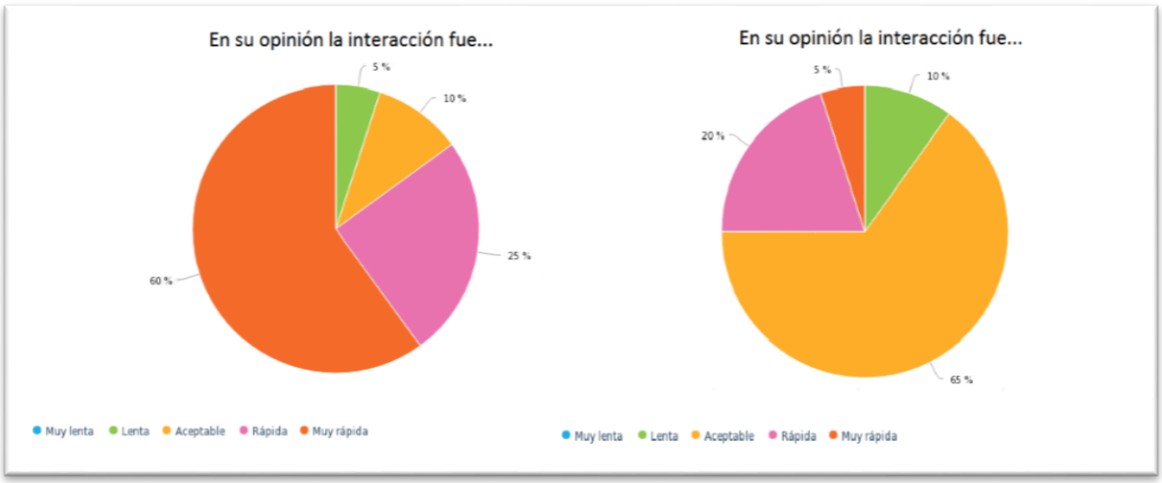


Valor mínimo	Valor máximo	Valor medio
3	5	3,6

Valor mínimo	Valor máximo	Valor medio
3	5	3,65

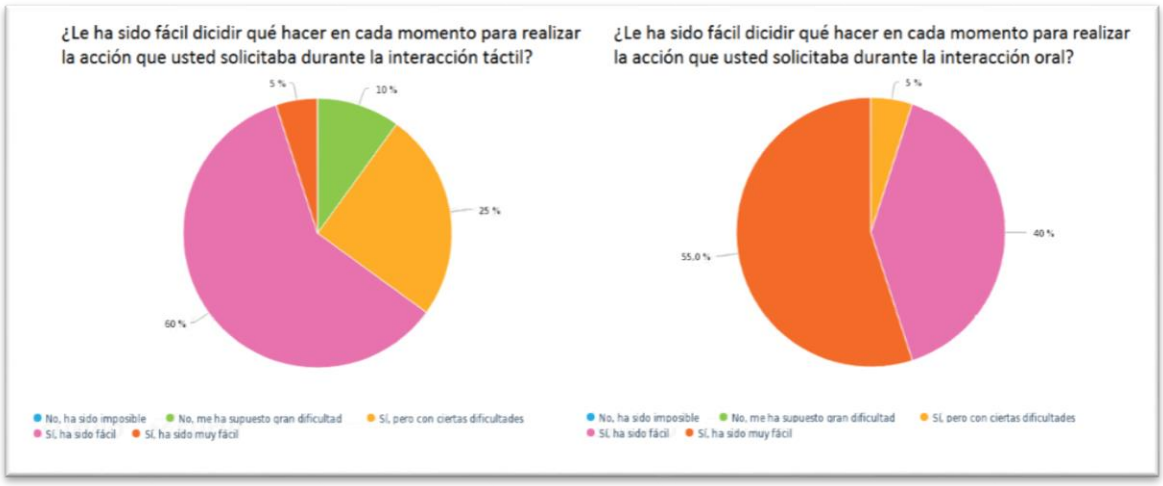
Valor mínimo	Valor máximo	Valor medio
3	5	4,9

Valor mínimo	Valor máximo	Valor medio
3	5	4,4



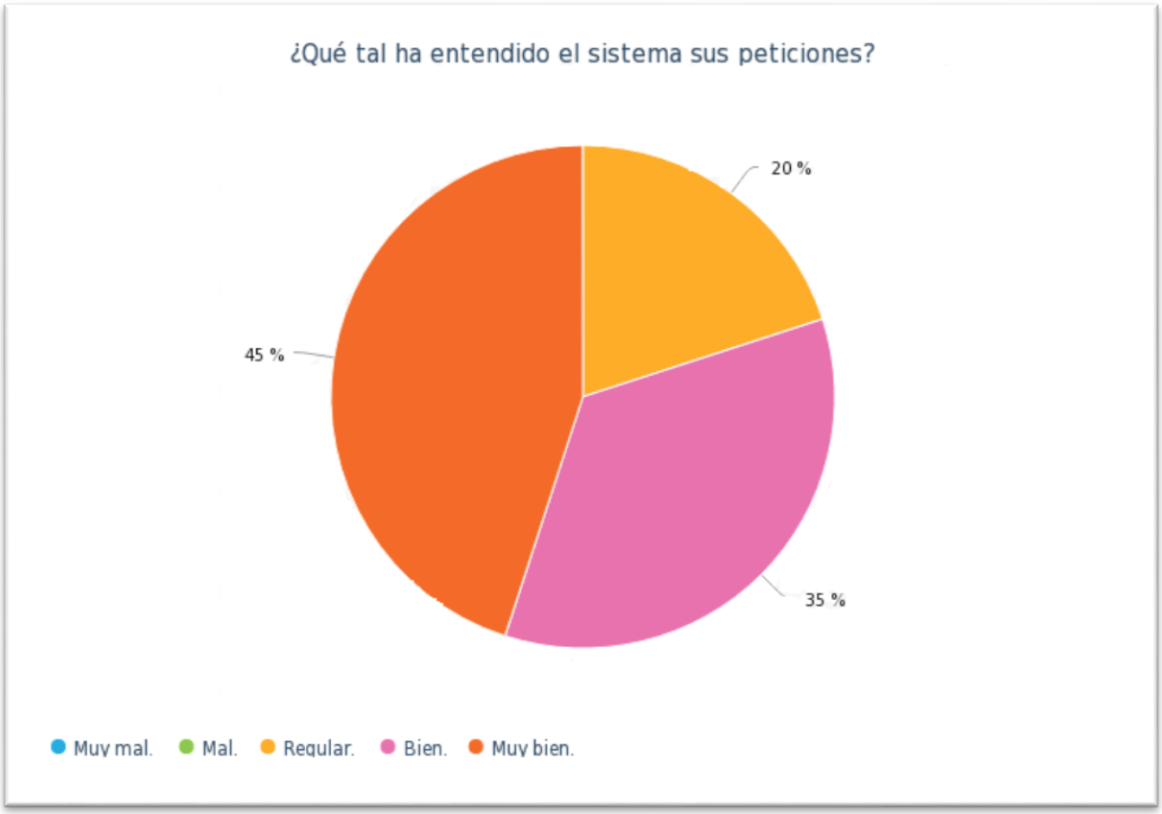
Valor mínimo	Valor máximo	Valor medio
2	5	4,4

Valor mínimo	Valor máximo	Valor medio
2	5	3,2

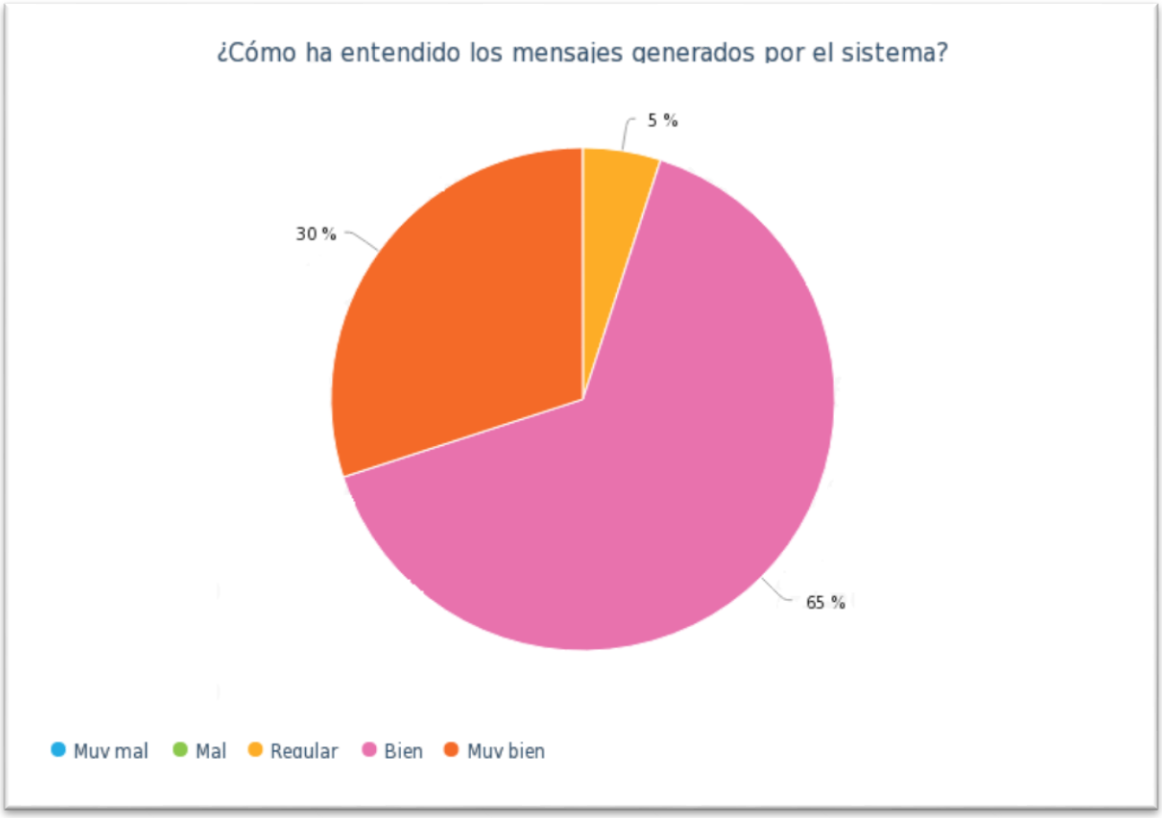


Valor mínimo	Valor máximo	Valor medio
2	5	3,6

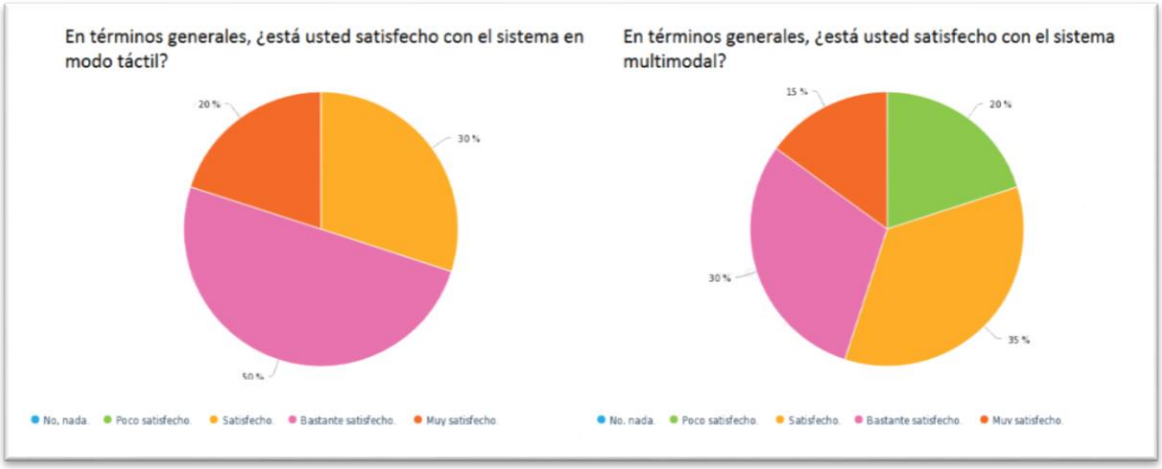
Valor mínimo	Valor máximo	Valor medio
3	5	4,5



Valor mínimo	Valor máximo	Valor medio
3	5	4,25



Valor mínimo	Valor máximo	Valor medio
3	5	4,25



Valor mínimo	Valor máximo	Valor medio
3	5	3,9

Valor mínimo	Valor máximo	Valor medio
2	5	3,4

Figura 5.2: Estadísticas de los resultados de la evaluación de la aplicación

Analizando los resultados obtenidos, se han obtenido las siguientes conclusiones:

1. **Nivel de experiencia previa en el uso de interfaces orales o multimodales:** se ha observado que el nivel de conocimiento de sistemas de interfaz oral o multimodal en los usuarios que han participado en la evaluación es variado, predominando un nivel medio-bajo, ya que se ha intentado seleccionar tanto a usuarios familiarizados con este tipo de sistemas como a usuarios sin experiencia previa con el objetivo de extraer conclusiones más fiables
2. **Nivel de dificultad del sistema:** los usuarios han valorado, para ambos tipos de interfaz, que el nivel de dificultad del manejo del sistema se encuentra en normal y muy fácil, encontrando algo más sencillo el manejo de la aplicación a través de interfaz multimodal.
3. **Detección de errores durante la interacción:** los usuarios no han percibido en general errores en su interacción con la aplicación en ninguna de las modalidades de interfaz. Se aprecia que el modo multimodal se ha visto algo más afectado ante la presencia de errores ya que el usuario puede sentirse más inseguro a la hora de tomar una decisión después de la ocurrencia de un error durante la interacción oral frente a las interfaces habituales.
4. **Velocidad percibida durante la interacción:** los usuarios han valorado que la velocidad de la interacción en el modo táctil ha sido muy rápida y en el modo multimodal ha sido aceptable. La velocidad durante la interacción en el modo multimodal se ha podido ver afectada debido al tiempo empleado en el procesamiento de la señal de voz emitida por el usuario, así como el tiempo que dura la locución del sintetizador de voz, durante el diálogo entre el usuario y la aplicación.
5. **Seguridad del usuario frente a las acciones a realizar en cada momento:** los usuarios han valorado que ha sido fácil tomar decisiones sobre qué hacer en cada momento en los dos modos de interfaz, aunque se han sentido con mayor al hacer uso de la interacción multimodal. El motivo principal por el que el modo oral aporta ventaja reside en que la aplicación en este modo emite instrucciones de utilización durante el diálogo. Además, esto refleja que las fórmulas utilizadas para realizar peticiones a la aplicación a través de la voz han resultado muy intuitivas para los usuarios.
6. **Grado en el cual el usuario valora que es entendido por el sistema:** el usuario ha valorado como muy alto el grado en el que el sistema ha entendido las peticiones a través de interfaz oral, lo que quiere decir que, por lo general, las acciones realizadas por el sistema se ajustaban a las peticiones dictadas por el usuario. Este resultado es muy significativo para la evaluación de la aplicación ya que dota de una alta robustez al sistema.
7. **Grado en el cual el usuario valora que entiende los mensajes del sistema:** en lo que respecta al grado de en el que el usuario ha entendido las locuciones del sintetizador se han obtenido buenos resultados. Hay que tener en cuenta que esto depende del motor de síntesis que el usuario haya programado como

principal en el dispositivo, cuyas voces pueden ser más o menos naturales pero, por lo general, suelen ser bastante nítidas.

Capítulo 6

Conclusiones y trabajo futuro

En el presente capítulo se realiza un balance general del trabajo realizado en el presente Trabajo de Fin de Grado. Se exponen las conclusiones finales extraídas realizando una valoración de los objetivos cumplidos frente a los marcados inicialmente. Finalmente, se presentan los posibles trabajos futuros que se podrían desarrollar a partir de este proyecto con el fin de mejorar la aplicación desarrollada.

6.1 Conclusiones

En el presente Trabajo de Fin Grado se ha desarrollado una aplicación multimodal para dispositivos móviles Android lo que constituye el objetivo principal del proyecto. Se trata de la aplicación denominada DayByDay cuya función es la de servir como agenda personal avanzada para el usuario. Dicha aplicación permite al usuario almacenar todo tipo de información referente a sus actividades diarias, a modo de recordatorio o diario, de forma organizada y permitiendo una acceso sencillo y rápido a la misma. Además, aprovecha los servicios y posibilidades que ofrece la plataforma Android así como los sensores disponibles en los teléfonos móviles inteligentes para enriquecer la forma de generar los contenidos. De esta forma, el usuario puede guardar en ella desde sencillas notas escritas hasta imágenes, grabaciones de voz o localizaciones.

Asimismo, se ha desarrollado un sistema de interactuación con la aplicación los más fácil e intuitiva posible. Esto se ha conseguido a través del uso de una interfaz multimodal que comprende las interfaces táctiles tradicionales y el uso de interfaces orales que, a través de frases del lenguaje natural, emulan un diálogo con el usuario. Realizado una revisión de la funcionalidad y el manejo de la aplicación desarrollada, se puede concluir que se ha alcanzado el objetivo principal del presente Trabajo de Fin de Grado.

Con el fin de alcanzar el objetivo principal, se han marcado inicialmente unos objetivos parciales a seguir. A continuación se realiza un balance de los resultados obtenidos a partir de estos:

- **Estudio de los sistemas de diálogo.**
  En primer lugar, se ha llevado a cabo un estudio completo de los sistemas de diálogo en cuanto a funcionalidad y arquitectura modular. Además, se ha realizado un análisis de los requisitos que debe cumplir un sistema de diálogo

ideal y de las limitaciones a las que están sujetos, así como un estudio de aplicaciones y ejemplos existentes en la actualidad. Este estudio ha sido necesario para poder incorporar a la aplicación desarrollada las características de mayor interés de los sistemas de diálogo, aprovechando de este modo todo el potencial que ofrecen.

- **Estudio de la plataforma Android**
  Se ha realizado un estudio detallado de la plataforma Android en cuanto a características, arquitectura, componentes y funcionalidad. Además, se ha analizado en profundidad el entorno de desarrollo Android así como las herramientas y recursos que ofrece. Este estudio es indispensable antes de realizar el desarrollo de cualquier aplicación para dispositivos móviles Android y ha supuesto una gran fuente de conocimiento y experiencia en este tipo de sistemas.

- **Análisis de las posibilidades que ofrece Android para el desarrollo de aplicaciones que permitan la interacción oral con el usuario.**
  Antes de comenzar con el desarrollo de la aplicación multimodal para dispositivos móviles Android, ha sido necesario realizar un estudio completo sobre las posibilidades que ofrece Android para integrar el reconocimiento automático del habla y la síntesis de texto a voz en sus aplicaciones.
  Como se ha visto en este estudio, los desarrolladores de Android junto con Google han trabajado desde los inicios por integrar estos sistemas en los dispositivos móviles Android con el objetivo de buscar la comodidad en el uso de estos dispositivos. Es por ello que Android ofrece una multitud de posibilidades para el desarrollo de aplicaciones que permitan la interacción oral con el usuario.

  En lo que respecta al reconocimiento de voz, se ha concluido que la opción más simple consiste en enviar un objeto de la clase `android.speech.RecognizerIntent` a la aplicación de búsqueda por voz de Google, por lo que se han analizado las principales características del paquete `android.speech` del API de Android.

  En cuanto a la síntesis de texto a voz, se ha visto que a partir de la versión 1.6 de Android se incorpora en la mayoría de los dispositivos un motor de síntesis de voz denominado Pico TTS, aunque se han estudiado otros motores alternativos que el usuario puede instalar y configurar en el dispositivo. Este motor de síntesis permite integrar de forma sencilla la síntesis de voz mediante la clase `TextToSpeech` del paquete `android.speech.tts`.

  Además, como ayuda para perfilar el diseño de la aplicación a desarrollar, se han consultado ejemplos de asistentes virtuales y otras aplicaciones para

dispositivos móviles Android de interés que integran el reconocimiento automático del habla y la síntesis de texto a voz.

- **Estudio de otras tecnologías necesarias en el desarrollo de la aplicación para dispositivos móviles Android.**
  Además del estudio realizado de la plataforma Android y de la metodología de integración de sistemas de reconocimiento de voz y síntesis de texto a voz en una aplicación Android, ha sido necesario el estudio sobre la integración de otras tecnologías necesarias en el desarrollo de la presente aplicación:
    - La creación, gestión y manipulación de una base de datos interna SQLite en Android, necesaria para almacenar y manejar la información introducida por el usuario en la aplicación. Las bases de datos SQLite, debido a su pequeño tamaño, son altamente adecuadas para ser utilizadas en dispositivos con poca memoria como teléfonos móviles o, en general, cualquier dispositivo móvil Android. Además, SQLite proporciona funcionalidad a través de una librería y no como un proceso separado por lo que se reducen las dependencias externas, simplifica las operaciones y aumenta la portabilidad. Por tanto se ha considerado que, en base a las características estudiadas, SQLite alcanza los requisitos de almacenamiento de la aplicación desarrollada.

    - El estudio de proveedores y gestores de localización que ofrece la plataforma Android para la construcción de servicios basados en localización (*location based services)* así como el uso de la API v2 de Google Maps para Android para la implementación de actividades basadas en mapas. En concreto, para la obtención de la localización del dispositivo se ha hecho uso de los métodos y contantes de la clase `android.location.LocationManager` disponible en el API de Android. También se ha implementado la clase android.location.Geocoder, utilizada para la traducción entre localizaciones (expresadas en latitud y longitud) y direcciones de calles. Este estudio de estas tecnologías ha sido necesario para los módulos ***Lugares*** y ***Eventos.***

    - La integración de un servicio web de tipo cliente/servidor para la extracción de contenido de páginas web en el módulo ***Notas***. En concreto, se ha implementado un servicio web RESTful, por su alta escalabilidad y rendimiento del sistema. Para ello, se han hecho uso de las librerías `http-request,` para realizar la petición, y la librería Gson para parsear la respuesta del servidor. Además, la implementación de este servicio ha implicado la ejecución de tareas en segundo plano, necesario para que aquellas operaciones largas o costosas no bloquen o ralenticen la ejecución del resto de componentes de la aplicación. Para evitar este efecto, se ha hecho uso de la clase auxiliar AyncTask, sobrescribiendo los métodos necesarios.

El balance obtenido del proyecto es, por lo tanto, muy positivo ya que se han alcanzado todos los objetivos parciales marcados inicialmente y, con ello, el objetivo principal, adquiriendo una gran experiencia en el desarrollo de aplicaciones Android, así como altos conocimientos en tecnologías desconocidas anteriormente a la realización del presente trabajo.

6.2 Trabajo futuro

A continuación, se describen las líneas futuras de trabajo que se proponen como mejora de las prestaciones de la aplicación desarrollada:

- **Mejoras en el reconocimiento automático del habla**

Una importante mejora consiste en la implementación de un mecanismo de "Siempre a la escucha", de forma que la aplicación se quede a la espera de que el usuario diga una palabra clave para comenzar a procesar las peticiones del usuario. De esta forma, se conseguiría un manejo completo de la aplicación sin utilizar el sistema táctil, lo que aumentaría en funcionalidad en contextos en los que resulta difícil o poco aconsejable el uso de interfaces tradicionales, o para usuarios con problemas de visión o discapacidades motoras. Aunque estos mecanismos se están empezando a emplear en ciertas aplicaciones y dispositivos, aún no existe ninguna API disponible para su implementación o documentación que haga referencia a esta funcionalidad.

Por otro lado, como se ha visto, la creación de los ítems *Eventos* y *Notas* no permite la entrada por voz de los campos fecha y hora. Aunque estos campos pueden ser rellenados más tarde en la edición de los ítems cuando el usuario pueda utilizar las interfaces táctiles, sería interesante permitir al usuario la creación completa de estos ítems a partir de la voz, implementando un mecanismo de procesamiento de la locución recibida para reconocer en ella cada campo del formulario.

Una última línea de trabajo con respecto al reconocimiento automático del habla consiste en permitir que funcione el reconocimiento *offline,* lo que permite utilizar el reconocimiento de voz sin la necesidad de tener acceso a Internet. Por el momento, no existe ninguna API disponible para implementar esta funcionalidad y se puede activar sólo en algunos dispositivos desde el menú de ajustes, por lo que no se ha incorporado por el momento en la aplicación desarrollada.

- **Aprender de los gustos del usuario**

Una significativa mejora consiste en la implementación de un mecanismo de "aprendizaje" por parte de la aplicación a partir de la información aportada por el usuario. De esta manera, se sugiere que la aplicación pueda realizar recomendaciones

y ofrecer información personalizada sin necesidad de que el usuario haga ninguna petición. Por ejemplo, la aplicación sería capaz de sugerir centros comerciales u ofertas a partir de las notas con la categoría *Compras* o informar sobre conciertos próximos a partir de las notas con la categoría *Música.* Esto se realizaría a través de una recopilación de los gustos y preferencias (géneros musicales, autores de libros, ciudades que suele visitar, etc) almacenándolos en la base de datos interna SQLite y realizando a través de las entradas más significativas un servicio web que ofrezca información de interés para el usuario.

- **Idiomas**

Como se ha visto, la aplicación desarrollada está implementada en dos idiomas: español e inglés. Esto implica que tanto el vocabulario de la aplicación como las locuciones generadas por el sistema y la metodología seguida para el reconocimiento de voz funcionen acorde al idioma seleccionado en el dispositivo. Sería beneficioso para aumentar el número de posibles usuarios extender el uso de la aplicación a otros idiomas.

Capítulo 7

Gestión del proyecto

En el presente capítulo se presenta la planificación temporal de las tareas en las que se divide el proyecto, así como un desglose de los costes del proyecto.

7.1 Planificación temporal

En base a la división del proyecto en fases descrita en la Sección 1.3, se ha realizado la planificación temporal a través de un diagrama de Gantt, el cual expone el tiempo de dedicación para cada tarea a lo largo del tiempo total empleado en el desarrollo del proyecto. Para la creación del diagrama se ha utilizado la herramienta *OpenProj.*

La Figura 7.1 muestra la tabla realizada con la herramienta *OpenProj* que indica las fechas de inicio y fin y duración estimada de cada tarea. Como se observa, contiene las tres fases principales del proyecto (planificación, ejecución y cierre) que contienen el resto de sub-tareas que se enumeraron en la Sección 1.3.

		Name	Duration	Start	Finish
1		Inicio del proyecto	0 days	10/1/14 8:00 AM	10/1/14 8:00 AM
2		**Fase de planificación**	**71 days**	**10/1/14 8:00 AM**	**1/7/15 5:00 PM**
3		Estudio de los sistemas d...	10 days	10/1/14 8:00 AM	10/14/14 5:00 PM
4		Estudio de la plataforma ...	5 days	10/16/14 11:00 AM	10/23/14 11:00 AM
5		Estudio de los sistemas d...	20 days	10/24/14 8:00 AM	11/20/14 5:00 PM
6		Estudio de aplicaciones si...	7 days	11/21/14 8:00 AM	12/1/14 5:00 PM
7		Definición de la funcionali...	10 days	12/2/14 8:00 AM	12/15/14 5:00 PM
8		Estudio de la tecnologías ...	17 days	12/16/14 8:00 AM	1/7/15 5:00 PM
9		**Fase de ejecución**	**115.75 days**	**1/8/15 8:00 AM**	**6/18/15 3:00 PM**
10		Diseño detallado	5.875 days	1/8/15 8:00 AM	1/15/15 4:00 PM
11		Programación de la aplica...	78.875 days	1/15/15 4:00 PM	5/6/15 3:00 PM
12		Integración y pruebas	20 days	5/6/15 3:00 PM	6/3/15 3:00 PM
13		Evaluación de la aplicación	11 days	6/3/15 3:00 PM	6/18/15 3:00 PM
14		**Fase de cierre**	**189.875 days**	**1/8/15 8:00 AM**	**9/30/15 4:00 PM**
15		Redacción de la memoria	167.875 days	1/8/15 8:00 AM	8/31/15 4:00 PM
16		Preparación de la presen...	22 days	8/31/15 4:00 PM	9/30/15 4:00 PM
17		Fin del proyecto	0 days	10/1/15 8:00 AM	10/1/15 8:00 AM

Figura 7.1: Tabla realizada con *OpenProj* con la duración estimada de cada tarea

La Figura 7.2 muestra el diagrama de Gantt resultante de aplicar la planificación temporal de la tabla de la Figura 7.1. En dicho diagrama se pueden observar las

relaciones de precedencia de cada tarea y cuáles se solapan en el tiempo, como las fases de ejecución y de redacción de memoria.

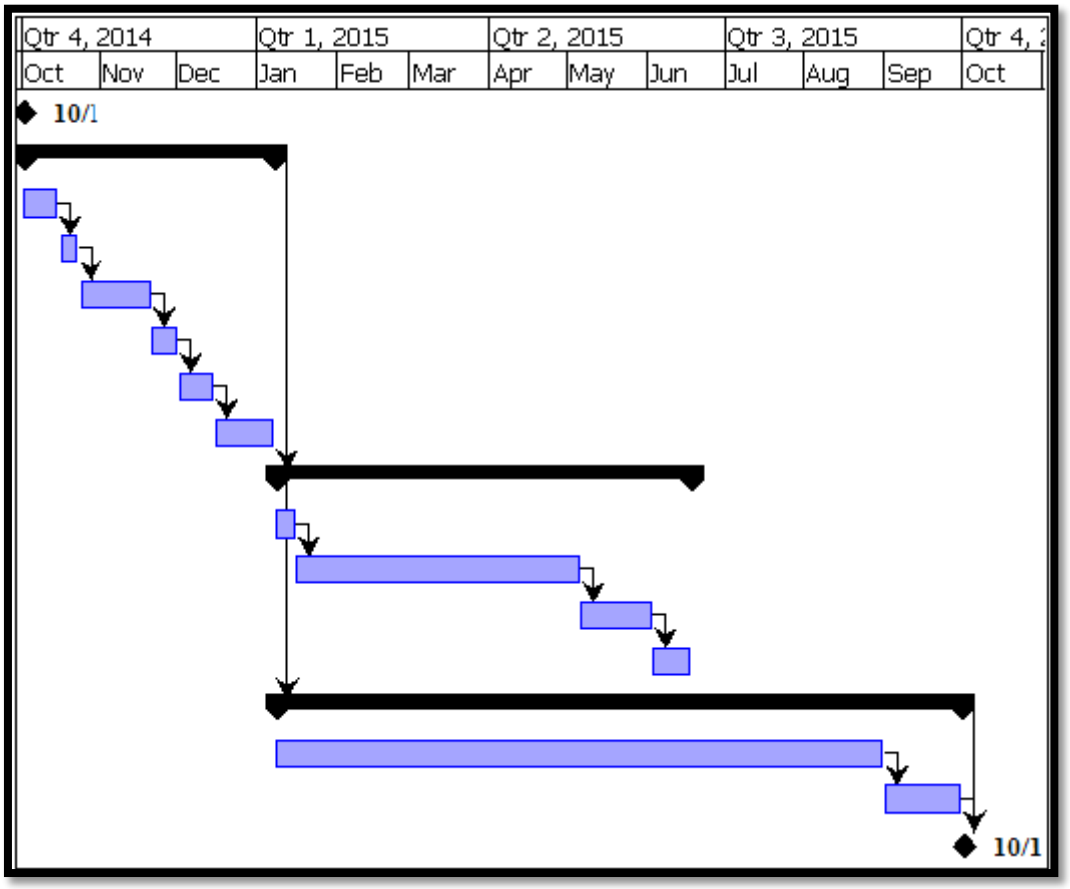


Figura 7.2: Diagrama de Gantt de la planificación temporal del Trabajo de Fin de Grado

Por lo tanto, se ha estimado una duración de 261 días para la realización del trabajo, sin incluir aquellos días que no se ha ejecutado ninguna tarea.

7.2 Presupuesto

En base a la planificación y duración que se han establecido en la sección anterior, a continuación se presenta el desglose presupuestario del presente Trabajo de Fin de Grado que incluye los costes de personal y de equipo.

Costes de Personal

A partir de la fórmula:

Coste = ((duración días * horas diarias)/dedicación persona mes) * coste persona mes

Se calcula el coste de personal, teniendo en cuenta que para el presente trabajo:

- Duración en días = 261
- Horas diarias = 4
- Dedicación persona mes = 131,25
- Coste persona mes = 2694,39 (ingeniero)

Dando como resultado unos gastos de personal de 21.431,95 €.

Costes de Equipo

Los recursos empleados en el presente Trabajo de Fin de Grado son los siguientes:

- Recursos Hardware

  - Ordenador portátil: 900 €
  - Smartphone Sony Ericsson Xperia V: 200 €
  - Cable USB: 5 €

- Recursos Software:

  - Entorno de desarrollo de código abierto multiplataforma Eclipse: 0 €
  - SDK *(Software Development Kit)* de Android: 0 €
  - JDK *(Java debelopment kit)*: 0 €
  - Plug-in ADT (Android Development Tools) para Eclipse: 0 €
  - Sintetizador de voz PICO TTS: 0 €
  - Aplicación de búsqueda por voz de Google: 0 €
  - API versión 2 de Google Maps para Android: 0 €
  - Librería *Android-support* (v4 y v7)*:* 0 €
  - Editor de programación Notepad++: 0 €
  - Herramienta *OpenProj* de administración de proyectos: 0 €

La Tabla 7.1 muestra la amortización de los equipos:

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Ordenador portátil	900	100	12	60	180
Smartphone Sony Ericsson Xperia V	200	100	12	60	40
Cable USB	5	100	12	60	1
				Total	221

Tabla 7.1: Amortización de equipos

La fórmula utilizada para el cálculo de la amortización es:

$$\frac{A}{B} \times C \times D$$

Donde,

- A = nº de meses desde la fecha de facturación en que el equipo es utilizado
- B = periodo de depreciación (60 meses)
- C = costes del equipo (sin IVA)
- D = % del uso que se dedica al proyecto (habitualmente 100%)

Resumen de costes

La Tabla 7.2 muestra el resumen de costes del presente Trabajo de Fin de Grado.

Fuente	Presupuesto Costes Totales
Personal	21.431,95
Amortización	221
Subcontratación de tareas	0
Costes de funcionamiento	0
Total sin IVA	21.652,95
Total con IVA (21 %)	26.200,07

Tabla 7.2: Resumen de costes

El presupuesto total del presente proyecto asciende a la cantidad de VEINTISEIS MIL DOSCIENTOS EUROS.

Bibliografía

- **[1]** http://es.wikipedia.org/wiki/Almacenamiento_vital_digital, Accedido en Enero de 2015

- **[2]** López-Cózar, R., and Araki, M.: 'Spoken, Multilingual and Multimodal Dialogue Systems: Development and Assessment' (Wiley, 2005).

- **[3]** Griol, D., Callejas, Z., and López-Cózar, R.: 'Utilización de los sistemas de diálogo hablado para el acceso a la información en diferentes dominios', II Conferencia Internacional sobre Brecha Digital e Inclusión Social, Madrid, España, del 28-30 de obtubre, 2009.

- **[4]** Llisterri, J. (2006): 'Introducción a los sistemas de diálogo' In J. Listerri & M.J. Machuca (Eds.), 'Los sistemas de diálogo.' (pp. 11-21). Bellaterra, Soria: Universitat Autònoma de Barcelona, Fundación Duques de Soria.

- **[5]** Griol, D.: 'Desarrollo y evaluación de diferentes metodologías para la gestión automática del diálogo.' Tesis Doctoral, Universidad Politécnica de Valencia, 2007

- **[6]** Marquet, I. 'Desarrollo de un asistente multimodal para proporcionar información sobre la televisión y el cine en dispositivos Android' Proyecto Fin de Carrera, Universidad Carlos III de Madrid, 2014.

- **[7]** http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/146-las-versiones-de-android-y-niveles-de-api, Accedido en Enero de 2015

- **[8]** http://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android, Accedido en Enero de 2015

- **[9]** https://support.google.com/nexus/answer/2781851?hl=es, Accedido en Enero de 2015

- **[10]** http://www.elandroidelibre.com/2011/09/controla-tu-android-con-la-voz-con-google-voice-actions.html, Accedido en Enero de 2015

- **[11]** http://www.geek.com/apple/google-is-close-to-releasing-siri-rival-called-majel-1449289/, Accedido en Febrero de 2015

- **[12]** http://andro4all.com/2014/12/google-now-historia-evolucion-uso, Accedido en Febrero de 2015

- **[13]** http://www.desarrolloweb.com/actualidad/llega-busqueda-inteligente-google-6952.html, Accedido en Febrero de 2015

- **[14]** Hashimi, S., Komatineni, S., y MacLean, D. (2010). Exploring Text to Speech and Translate APIs. En S. Hashimi, S. Komatineni, y D. MacLean (Eds.), *Pro Android 2* (pp. 583–589). United States of America: Apress.

- **[15]** http://processors.wiki.ti.com/index.php/Text-To-Speech-and-Speech-Recognition-on-Android, Accedido en Febrero de 2015

- **[16]** http://developer.android.com/reference/packages.html, Accedido en Octubre de 2014

- **[17]** https://software.intel.com/es-es/articles/developing-android-applications-with-voice-recognition-features, Accedido en Febrero de 2015

- **[18]** http://www.ispeech.org/text.to.speech, Accedido en Febrero de 2015

- **[19]** http://android-developers.blogspot.com.es/2009/09/introduction-to-text-to-speech-in.html, Accedido en Febrero de 2015

- **[20]** http://www.emezeta.com/articulos/10-sintetizadores-de-voz-tts-para-android#axzz2GjM77pDz, Accedido en Febrero de 2015

- **[21]** https://www.ivona.com/, Accedido en Febrero de 2015

- **[22]** http://espeak.sourceforge.net/, Accedido en Febrero de 2015

- **[23]** http://www.nuance.com/for-business/by-solution/customer-service-solutions/solutions-services/inbound-solutions/loquendo-small-business-bundle/text-to-speech/index.htm, Accedido en Febrero de 2015

- **[24]** http://www.eguidedog.net/, Accedido en Febrero de 2015

- **[25]** http://www.vajatts.com, Accedido en Febrero de 2015

- **[26]** https://svoxmobilevoices.wordpress.com/, Accedido en Febrero de 2015

- **[27]** http://androidzone.org/2012/12/mejores-asistentes-de-voz-para-android, Accedido en Febrero de 2015.

- **[28]** http://www.sgoliver.net/blog/mapas-en-android-google-maps-android-api-v2-i/, Accedido en Marzo de 2015

- **[29]** https://columna80.wordpress.com/2011/02/17/arquitectura-de-android/, Accedido en Marzo de 2015

- **[30]** Meier, R.: "Professional Android 2 Application Development", Wiley Publishing Inc. 2010

- **[31]** http://www.arquitecturajava.com/servicios-rest/, Accedido en Febrero de 2015

- **[32]** http://www.omdbapi.com, Accedido en Febrero de 2015

- **[33]** http://www.imdb.com, Accedido en Febrero de 2015

- **[34]** http://kevinsawicki.github.io/http-request/, Accedido en Febrero de 2015

- **[35]** http://www.survio.com/es/, Accedido en Julio de 2015

- **[36]** http://www.verbio.com/es/, Accedido en Enero de 2015

- **[37]** Turunen M., Salonen E-P. y Hartikainen, M. (2004): "AthosMail – a Multilingual Adaptative Spoken Dialogue System for E-mail Domain" Proceedings of the COLING Workshop Robust and Adaptive Information Processing for Mobile Speech Interfaces, Geneva, Switzerland.

- **[38]** http://groups.csail.mit.edu/sls/research/jupiter.shtml, Accedido en Enero de 2015

- **[39]** http://groups.csail.mit.edu/sls/research/pegasus.shtml, Accedido en Enero de 2015

- **[40]** http://groups.csail.mit.edu/sls/research/mercury.shtml, Accedido en Enero de 2015

- **[41]** http://www.ydilo.com/es/, Accedido en Enero de 2015

- **[42]** Péres, G., Amores, G. y Manchón, P. (2006): "A multimodal achitecture for home control by disabled users" En *Spoken Language Technology Workshop* (pp. 134–137). Palm Beach, Aruba.

- **[43]** http://www.naturalvox.com/, Accedido en Enero de 2015

- **[44]** Mostow, J. (2008): "Experience from a reading tutor that listens: Evaluation purposes, excuses, and methods" En C. Kinzer y L. Verhoeven (Eds.), *Interactive Literacy Education: Facilitating Literacy Environments Through Technology* (pp. 117–148). New York: Erlbaum Publishers.

- **[45]** Vaquero, C., Saz, O., Lleida, E., Marcos, J. y Canalís, C. (2006): "VOCALIZA: An application for computer-aided speech therapy in Spanish language" En *IV Jornadas en Tecnología del Habla* (pp. 321–326). Zaragoza, España.

- **[46]** Litman, D. y Silliman, S. (2004): "ITSPOKE: an intelligent tutoring spoken dialogue system." En *Demonstration papers at HLT-NAACL* (Inf. Téc., pp. 5–8). Stroudsburg, PA, USA: Association for Computational Linguistics.

- **[47]** http://www.aisoy.com/, Accedido en Enero de 2015

- **[48]** http://www.renfe.com/, Accedido en Enero de 2015

- **[49]** http://www.ispeech.org, Accedido en Febrero de 2015

- **[50]** http://cmusphinx.sourceforge.net/, Accedido en Febrero de 2015

- **[51]** https://maps.google.es/, Accedido en Febrero de 2015

- **[52]** http://www.alicoid.com/, Accedido en Febrero de 2015

- **[53]** http://sherpaassitant.blogspot.com.es/, Accedido en Febrero de 2015

- **[54]** http://www.skyviapp.com/, Accedido en Febrero de 2015

- **[55]** https://assistant.ai/, Accedido en Febrero de 2015

- **[56]** https://play.google.com/store, Accedido en Enero de 2015

- **[57]** http://developer.android.com/sdk/installing/index.html

ANEXO A

Summary

The mobile devices Android application developed in this project is a personal multimodal advanced diary. This application allows the storage of all kinds of information about the daily activities in an organized way and with a quick and easy access using diverse research methods. This information is divided into categories depending on the nature of each item: a picture, a note, a place visited (geolocation) or a future event. Moreover, the application takes advantage of services and possibilities offered by the Android platform as well as smart phones sensors in order to enhance the way of generating content. Thus, the user can store simple notes as well as pictures, voice records or locations.

Since it is a multimodal application, the system allows the user to manage the application by using the usual tactile interfaces (screen or keyboard) or by voice. Also, the responses generated by the application can be represented both visually an orally. This functionality makes interacting with the application more interesting and intuitive. Figure 8.1 shows a general diagram of the designed system. When the user makes a request by voice, automatic speech recognition module processes the voice signal emitted by the user and recognizes the information contained in it. The response of this operation is the written representation of the information received, which is the input of the application. Likewise, the system output in text form is processed by the Text to Speech synthesizer (TTS) generating oral representation.

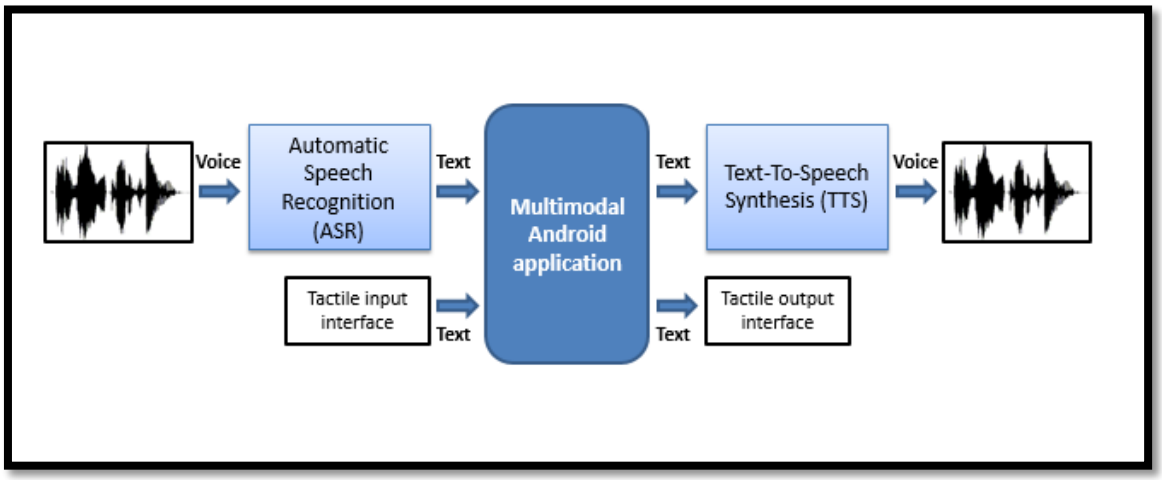


Figure 8.1: Modular diagram of the designed system

In order to integrate the voice recognition functionality, the recognition service used is *Google Voice Search,* as it is one of the best voice recognition systems for Android, supported in several languages and which is installed by default on most devices. This

application consists on an initial screen with the message “Speak now” indicating to the user that the device is “listening”. It is therefore necessary that the application is preinstalled in the device. Figure 8.2 shows the main screen of this application.

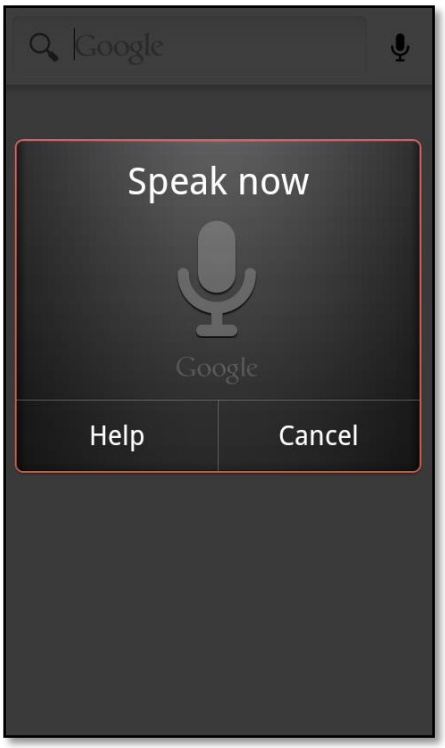


Figure 8.2: Google Voice Search application

This application works in a way that the received audio is transferred to Google servers in order to perform the recognition. Finally, the voice recognition results are returned to the application in the form of written representation so they can be processed by any Android application.

Regarding the Text-To-Speech Synthesis, from Android version 1.6 a synthesis engine called Pico TTS is incorporated in most devices. These synthesis engine allows the integration of the Text-To-Speech Synthesis on any Android application. However, the Android platform provides many alternative synthesis engines that can be downloaded and selected for its use in the device. The one developed in this project, like most existing application in the market, uses the TTS engine selected in the device to perform the synthesis. Thus it is ensured that the application will run correctly in any device, without having to download any additional applications.

On the other hand, Figure 8.3 shows the modular diagram of the application developed. It is composed of a main module and four modules that represent each type of category in which the stored information is divided. The following sections summarize the features and functionality of each module.

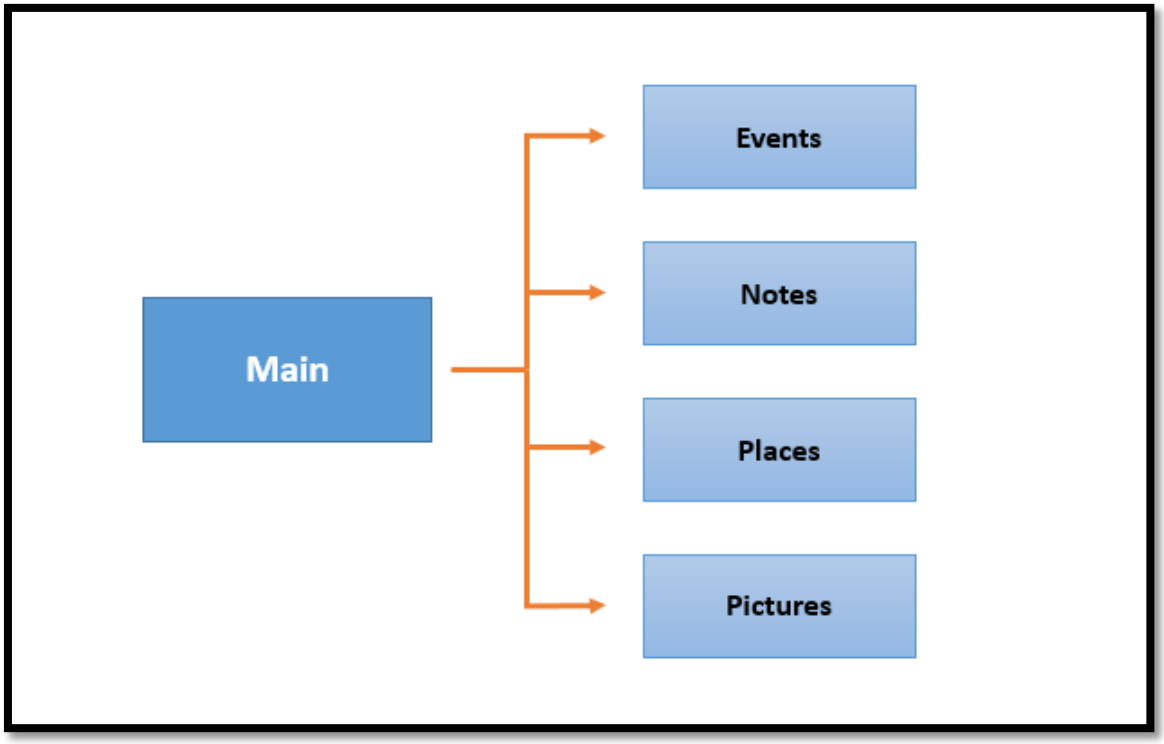


Figure 8.3: Modular diagram of the application developed

Main module

It is the application initial screen and it provides access to all other modules in the application. In this screen, the user can access to the information of any stored item, which is shown divided into categories and chronologically ordered. Such information is stored in an internal SQLite database. The Figure 8.4 shows an example of the four tabs that make this screen.

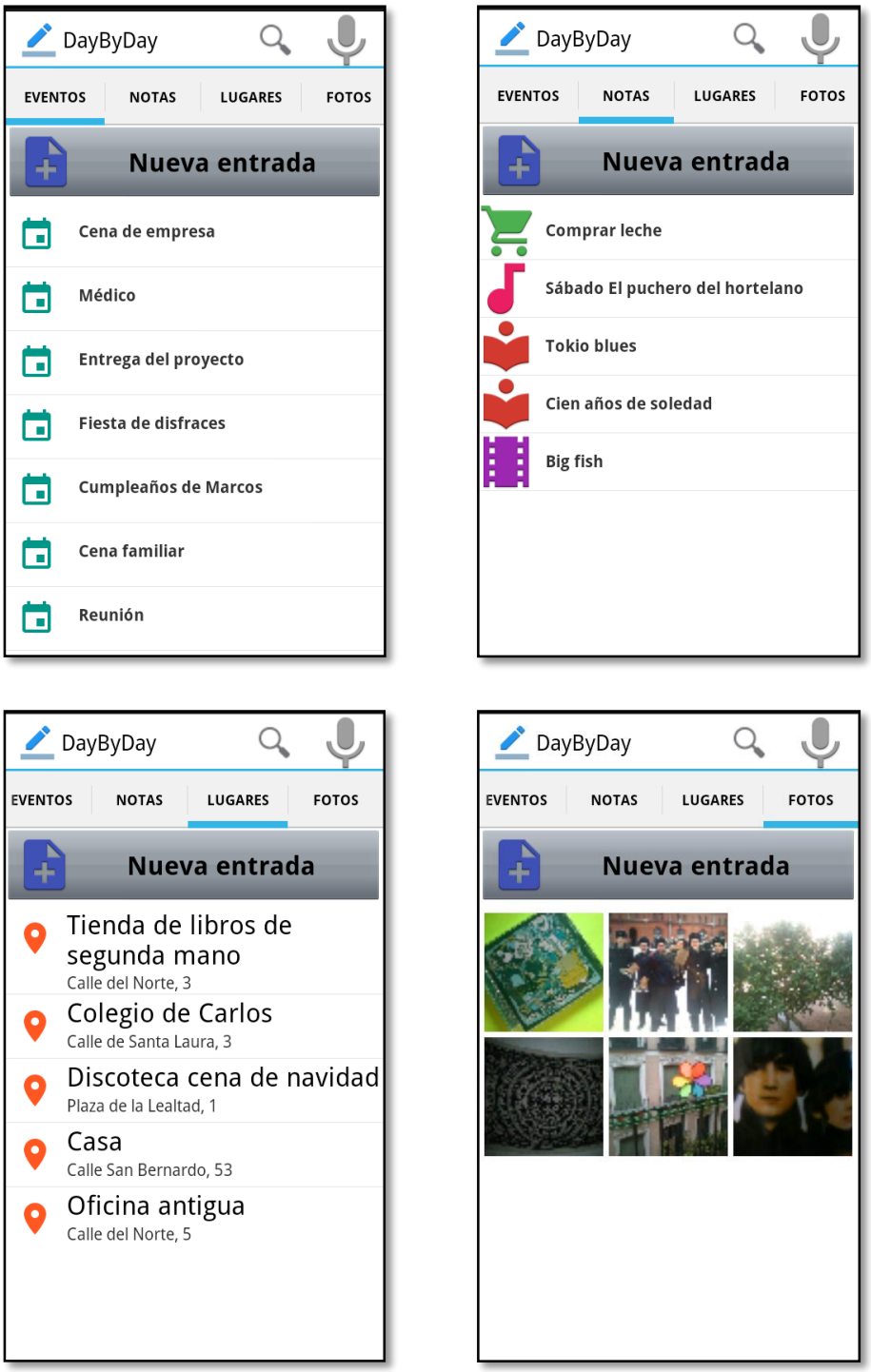


Figure 8.4: Application initial screen

Furthermore, through this activity the information can be searched using one of the two types of interface existing in the application: tactile or oral interface. Using the tactile interface, the user can search an item writing a keyword in the search bar. Also, the items can be filtered by a specific date, by accessing to the “Filter by date” option located in the menu on this screen. On the other hand, user can search an item using the *Google Voice Search* application that is activated via the microphone icon. Then

the user just have to dictate the keyword or the full date to automatically obtain the search result.

Events module

It includes all actions related to the information associated with the *Event* category. This type of item is characterized by the date and time of the event, its description and its location.

Coming up next, the actions of this module are presented, all of them accessible from the initial screen.

- **Create:** With this option, the user can access the screen to create and save a new event in which the characteristic information thereof is filled. This action also allows its management by the oral interface using Google Voice Search application, so that the user can dictate the subject and location of the event and save it without touching the screen.
- **View:** The **'**View' action allows the user to view the information of an event. In order to enrich the information provided by the user, if the event location is an existing location, a fragment of a map indicating the location is shown at the bottom of the screen.
- **Edit:** This option allows to edit the information of an event. The screen of this option is similar to the creation screen but the input components of the data show the old event data by default. It can also be managed by the oral interface as in the creation screen.
- **Delete:** It allows to permanently delete an event from the database.

Notes module

It includes all the actions related to the information associated with the *Note* category. This type of item is characterized by a date, a free text and a category to choose, which characterizes the content of the note as additional information to help the search of the item. The application suggests the following categories: Books, Cinema, Music, Shopping or None (freer category). This type of item is designed for those ideas, phrases, thoughts or simple reminders that the user wants to store, emulating a notebook.

The actions of this module are presented below, all of them accessible from the initial screen:

- **Create:** With this option, the user can access the screen to create and save a new event in which the characteristic information thereof is filled. This action can also be managed by the oral interface using the Google Voice Search application, so that the user can dictate the text of the note and save it without touching the screen.

- **View:** The 'View' action allows the user to view the text of a note, as well as its date and its category. In order to enrich the information provided by the user, the application added additional information in this screen in the case that the category is *Cinema, Music* or *Books.* If the category is *Cinema,* the movie poster and information about the film written in the note (director, writers, actors, etc.) are shown. This information is retrieved through a request to the IMDb server (Internet Movie Database). If the category is *Music,* the screen displays a link to Youtube application that provides automatically a search with the text of the note. If on the contrary the category is *Books*, a link to a search with the annotated text on the website of *La Casa del Libro* is displayed.
- **Edit:** This option allows the edition of the information of a note. The screen of this option is similar to the creation screen but the input components of the data show the old data of the note by default. It can also be managed by the oral interface as in the creation screen.
- **Delete:** It allows to permanently delete a note from the database.

Places module

It includes all actions related to the information associated with the *Place* category. This type of item is characterized by the geolocation of the user, the current date and, optionally, a brief description of the place in the form of text and/or audio. In order to obtain the current geolocation of the device, the GPS location provider has been used due to its being one of the most commonly used and the one that gives best features in terms of accuracy. To represent the map and the geolocation of the device, the Google Maps Android V2 API [28] has been implemented.

The actions of this module are presented below, all of them accessible from the initial screen:

- **Create:** With this option, the user can access the screen to create and save a new location in which the current geolocation of the user appears drawn on a map. The creation activity also allows to record an audio file to associate with the place. Once the user has pressed the *Save* button, the application allows to add a text note as a description of the place before saving the location. After that, the current geolocation and its information is stored in the SQLite database.
- **View:** The 'View' action allows the user to consult the information that characterize a *Place* item. This activity includes a fragment of a map with a mark pointing to the saved location, as well as the address of the place in the top of the screen. Also, the user can view the text note associated to the item pressing on the mark of the map. The audio file associated can be accessed through the button at the bottom of the screen. This functionality is implemented by using Google Maps Android V2 [28].

- **Edit:** This option allows to edit the descriptive note of a *Place* item. This functionality is performed through a pop-up screen (*Dialog*) which allows the insertion of a new text and save it, making the item is automatically associated with the new note.
- **Delete:** It allows to permanently delete a *Place* item from the database.

Pictures module

It includes all actions related to the information associated with the *Picture* category. This type of item is characterized by a photograph taken with the device, the date on which the image is taken and, optionally, a note that describes the image.

The actions of this module are presented below, all of them accessible from the initial screen:

- **Create:** With this option, the user can directly access to the camera application of the device. Once the picture is taken, an activity which allows to add a descriptive text, as well as to save or discard the image, is launched. After pressing the *Save* button, the location of the photo in the device's gallery as well as its date and descriptive text is stored in the SQLite database. This action can also be managed by the oral interface accessing to the Google Voice Search application through a microphone icon on this screen, so that the user can dictate the note without touching the screen.
- **View:** The 'View' action allows the user to look up a picture and its associated information. Also, the user can access the image from the gallery of the device by clicking on the picture in this screen.
- **Edit:** This option allows the edition of the descriptive note of a *Picture* item. The screen in this option is similar to the creation screen but the input components of the data show by default the old data of the note. It can also be managed by the oral interface as in the creation screen.
- **Delete:** It allows to permanently delete a *Picture* item from the database.

The creation functionality of *Events, Notes, Places* and *Pictures* modules explained above can be accessed from the main module using the oral interface through the Google Voice Search application icon located on the main screen. This requires the use of the command "Create" followed by the name of the respective category (eg. "Create photo"). Moreover, if screens creation of a note or an event are activated by the voice, the application detects that the user prefers to use the oral interface. Therefore, automatically, a dialog with the application starts, allowing the user to activate the screen creation, dictate the text of the item and save it without touching the screen. In this process, the user is assisted by the locutions of the application to always know the answer to give in each interaction. The dialog functionality is not

implemented for *Pictures* and *Places* modules since the creation of these items involves greater complexity.

In addition, the developed application is implemented in two languages: Spanish and English. This implies that the vocabulary, the locutions generated by the systems and the methodology for voice recognition operate according the selected language on the device.