Universidad
Carlos III de Madrid

Ingeniería Informática

PROYECTO FIN DE CARRERA

Desarrollo de una App para la recomendación de actividades de acuerdo con el reconocimiento de emociones del usuario

Autor: Cristina Valverde Martín

Tutor: Dr. David Griol Barres

Leganés, septiembre 2017

Título: Desarrollo de una App para la recomendación de actividades de acuerdo con el reconocimiento de emociones del usuario

Autor:

Director:

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____Septiembre_de 2017__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Agradezco a mi tutor David Griol, por ayudarme en este último paso para terminar la carrera. Tenía este trabajo pendiente desde hace años y finalmente con su dedicación y apoyo he podido llevarlo a cabo. Hoy en día me dedico al sector bancario, mundo de tarjetas, sistemas integrados, certificaciones. Sin embargo, gracias a David por acercarme un poquito a otro sector muy diferente de sistemas de recomendación, análisis de sentimientos y con mucha proyección desde mi humilde punto de vista.

Echando vista atrás, recuerdo con cariño, ilusión y esfuerzo los años de universidad. Hago mención a todos los profesores del 2001- 2008 que impartieron clase y me enseñaron los conocimientos necesarios para desarrollarme profesionalmente y poder disfrutar de lo que hago hoy en día. A la mayoría de compañeros que llenaron estos momentos de risas y afecto.

Por último, agradecer a mi familia que siempre me ha estado apoyando y animando a escoger la carrera y últimamente a poder finalizarla. Gracias por estar ahí siempre.

Resumen

En este Proyecto Fin de Carrera se ha desarrollado un sistema de recomendación multimodal para Android que consiste en un asistente virtual que reconoce las emociones, identifica el estado de ánimo del usuario y sus preferencias de ocio para recomendar actividades acordes a sus necesidades.

Gracias a la aplicación desarrollada, denominada PersonalOcio, el usuario de la aplicación puede visualizar como se encuentra a través de reconocimiento facial y reconocimiento de voz del dispositivo móvil. La aplicación realiza propuestas de ocio que el usuario no tenía contempladas e incluso terapéuticas para mejorar el estado anímico de la persona.

El presente proyecto también incluye un estudio completo de los sistemas de diálogo, sistemas de recomendación, reconocimiento facial y análisis de sentimientos con el objetivo principal de incorporar a la aplicación para Android las características típicas de estos sistemas y de este modo aprovechar toda la potencia que ofrecen.

Además, se realiza un análisis exhaustivo de la plataforma Android y del entorno de desarrollo de sus aplicaciones, centrándose en las posibilidades que ofrece dicha plataforma para el desarrollo de aplicaciones que permitan tanto la interacción oral con el usuario, es decir, que integren el reconocimiento automático del habla para la entrada, como la interacción visual para el reconocimiento facial.

Posteriormente, se determina el potencial de ambas interacciones oral y visual como punto de partida para el análisis de sentimientos y el sistema de recomendación, su importancia en el sector y las posibilidades que ofrece al incluir esta tecnología en el desarrollo de aplicaciones.

Palabras clave: Sistemas de recomendación, análisis de sentimientos, reconocimiento facial, gramáticas, ocio, interacción multimodal, y reconocimiento de voz

Abstract

In this Final Degree Project, a recommender system for Android has developed. It is intended to be used as a virtual assistant that recognizes the emotions, identifies the mood of the user and their leisure preferences to recommend activities according to their needs.

Thanks to the application developed, called PersonalOcio, the user of the application can visualize as it is through facial recognition and voice recognition of the mobile device. The application makes leisure proposals that the user had not contemplated and even therapeutic to improve the mood of the person.

The present project also includes a complete study of the recommender systems, facial recognition, sentiment analysis and dialogue systems, with the main objective of including to the Android application the typical characteristics of these systems and in this way to take advantage of all the power these systems offer.

In addition, a comprehensive analysis of the Android platform and the development environment of its applications is done, focusing on the possibilities offered by this platform for the development of applications that allow both oral interaction with the user, integrate the recognition automatic speech for input, such as visual interaction for facial recognition.

Moreover, the potential of both oral and visual interactions is determined as the starting point for the sentiment analysis and the recommendation system, its importance in the sector and the possibilities offered by including this technology in the development of applications.

Keywords: Recommender systems, sentiment analysis, facial recognition, grammars, leisure, multimodal interaction, and voice recognition

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

1. Introducción y objetivos

Este primer capítulo comienza con una introducción del presente Proyecto Final de Carrera, que incluye una breve descripción de los sistemas de diálogo, de sus limitaciones y antecedentes, con el objetivo de justificar la motivación de dicho proyecto de incluir análisis de sentimientos y reconocimiento facial a la hora de desarrollar una aplicación de recomendación multimodal para Android.

Describe la influencia que ejercen las tecnologías móviles en la sociedad, representadas a través de los dispositivos móviles, así como los principales sistemas operativos que surgen para interactuar con ellos y dotarles de funcionalidad.

Posteriormente, se describen los objetivos del proyecto y se establecen las fases de desarrollo. Por último, se resume el contenido de cada capítulo

1.1. Introducción

La facilidad de acceso a la información que hoy en día proporcionan los ordenadores ha hecho que se conviertan en valiosas herramientas para el ser humano. Si además se tiene en cuenta la llegada de los dispositivos móviles, casi ordenadores reducidos al tamaño de la palma de la mano, entonces podemos hablar no de herramientas si no casi de extremidades.

Cada día que pasa los sistemas informáticos tienen más posibilidades, integran más dispositivos y cada vez son más pequeños. Estos periféricos no solo permiten acceder a información de cualquier naturaleza y casi en cualquier lugar, sino que además permiten que la interacción con ellos sea de forma más natural. El micrófono y el altavoz, la pantalla y su capacidad para reconocer gestos sobre ella, han permitido a los ingenieros de software aprovechar sus capacidades para desarrollar aplicaciones más cercanas y más intuitivas para el ser humano.

Hoy en día el mundo de internet y las comunicaciones nos llevan a manejar mucha información. Analizar, extraer, modificar…cantidades de datos a veces estructurados y otras veces no, es lo que se llama Big Data. Los datos se pueden combinar con técnicas de Data Mining, Web Mining and Text Mining. Pero la información específica de foros, blogs, redes sociales, e-commerce, plataformas, etc donde los usuarios/consumidores expresan opiniones de todo tipo, es difícil de analizar de forma automática.

Con el objetivo de evitar las limitaciones de la interacción basada exclusivamente en el habla surgen los sistemas de diálogo multimodales. En una interacción multimodal el usuario no está restringido a utilizar el habla como único canal de comunicación, sino que puede utilizar varios dispositivos de entrada, como por ejemplo un teclado, un ratón, un micrófono, una cámara, una pantalla sensible al tacto, etc. Asimismo, un sistema multimodal puede utilizar diversos canales de salida para proporcionar información al usuario como, por ejemplo, voz, texto, gráficos o imágenes, con el objeto de estimular varios sentidos del usuario de forma simultánea.

No obstante, a pesar de que la investigación ha avanzado notablemente en las últimas décadas, construir sistemas de diálogo ideales constituye un reto ya que están sujetos a las limitaciones del reconocimiento automático del habla. Su comprensión y respuesta están restringidas a dominios específicos, se encuentran condicionados por la naturalidad del habla sintetizada y aún sigue existiendo la necesidad de que el interlocutor humano guíe al sistema con el fin de confirmar que la interpretación de sus preguntas es la correcta. Todo esto es debido a problemas del diálogo espontáneo.

De ahí la importancia de dar un paso más al reconocimiento de voz y analizar los sentimientos (Sentimiento Análisis -SA), emociones, opiniones y actitudes a través de texto de forma rápida. El análisis de sentimiento, también conocido como minería de opinión (opinión mining), es un término muy discutido en estos últimos años. El lenguaje humano es complejo. Enseñar a una máquina a analizar los diferentes matices gramaticales, variaciones culturales, jergas y faltas de ortografía de las menciones online es un proceso difícil. Y enseñar a una máquina a entender cómo el contexto puede afectar al tono, es aún más difícil. ¿Se puede medir el escepticismo, la esperanza, la ansiedad, la emoción o la falta de ella? Hasta que esto ocurra el análisis del sentimiento es unidimensional.

De momento el análisis automático de sentimiento aporta la capacidad para procesar altos volúmenes de datos con un mínimo de retardo y consistencia a bajo coste, lo que permite complementar el análisis humano en multitud de escenarios. Las tareas de detección, extracción y clasificaciones de opiniones, sentimientos y actitudes relacionas con diferentes temas de conversación son complicadas de

realizar. Incluso se puede aplicar a otros ámbitos más profesionales. Llegado el caso puede detectar un fraude, escrutar a pasajeros aéreos y ayudar a un técnico en un call center a lidiar mejor con un cliente enojado. Se pueden usar para monitorear empleados, encuestas en incluso en entrevistas de trabajo. Frente a todo este potencial preocupa la privacidad. Incluso al ser una tecnología con poco recorrido, su precisión es difusa por ahora.

En los últimos años, psicólogos e investigadores han analizado y proporcionado evidencias concretas sobre una cuestión dejada con anterioridad a los filósofos: ¿qué nos hace felices? Descubrimientos en el campo de la psicología positiva sugieren que las acciones de un individuo pueden tener un efecto significativo en su felicidad y satisfacción con la vida. En particular, el cultivo de emociones positivas mediante la realización de ciertas actividades puede ayudar a una persona a desarrollar áreas del cerebro asociadas al bienestar y la felicidad.

Un paso más en el análisis de sentimientos es el reconocimiento facial. Se ha convertido en los últimos años en un área de investigación activa que abarca diversas disciplinas, como el procesado de imágenes, el reconocimiento de patrones, la visión por ordenador y las redes neuronales. Involucra tanto a investigadores del área de informática como a neurocientíficos y psicólogos. Se podría considerar también dentro del campo de reconocimiento de objetos, donde la cara es un objeto tridimensional sujeto a variaciones de iluminación, pose, etc., y ha de ser identificada basada en su proyección 2D (excepto cuando se utilizan técnicas 3D).

Los sistemas de recomendación (RS) puede ser un buen escenario para aplicar este conocimiento extra del usuario. Además, hoy en día con la facilidad de acceso a la información y la multitud de opciones que nos ofrece el mercado, los sistemas de filtrado y recomendación personalizada se están viendo potenciados por esta necesidad.

1.2. Objetivos

El objetivo fundamental del proyecto es desarrollar una aplicación multimodal de recomendación de actividades de acuerdo con el reconocimiento de emociones del usuario para dispositivos móviles basado en el sistema operativo Android, que facilite opciones de ocio según el estado de ánimo del usuario y de sus preferencias. Para ello es necesario captar mediante video los rasgos faciales y posturas del usuario para determinar el estado anímico. Junto con el análisis de la voz y de su contenido se podrá ajustar las propuestas de ocio que se va a ofrecer. Incluso con su ubicación y las preferencias configuradas en su perfil se puede dar información más localizada. Esta aplicación tiene el nombre de PersonalOcio, se utilizará a lo largo del documento para hacer referencia a la misma.

En base a ese objetivo principal, la aplicación hace uso del reconocimiento automático del habla y del tratamiento de su contenido para obtener palabras clave e identificar el estado de ánimo (sentiment análisis -SA).

La aplicación se apoya en el reconocimiento del habla (Automatic Speech Recognition) desarrollada por Google para dispositivos móviles.

Además, necesita soporte multimedia para realizar capturas faciales e interpretar los gestos humanos. Por lo tanto, necesita de librerías de reconocimiento facial, análisis de sentimientos y generación de una gramática propia para capturar las palabras clave del texto obtenido tras procesar la voz y proporcionar de un listado de actividades la que más se ajuste al usuario en ese momento.

La principal ventaja que tiene la aplicación es el ahorro del tiempo debido al uso de la voz, lo que facilita el uso de la misma en personas mayores y/o personas con discapacidades motoras o visuales. Además, el desarrollo de la aplicación para dispositivos Android, tendrá un mayor rango de terminales en los que se pueda instalar, ya que este sistema operativo es la que más uso está teniendo en los últimos años dentro de los dispositivos móviles [17].

La aplicación desarrollada necesita conexión a Internet para acceder a fuentes de páginas web sobre ocio, restauración y eventos cercanos a la posición del usuario.

Una vez iniciada la sesión, el usuario podrá configurar su perfil respondiendo a unas preguntas básicas sobre sus preferencias y aficiones.

Como fin último, se encuentra el de abrir el camino hacia nuevas funcionalidades o desarrollos basados en el proyecto implementado como ampliar el conocimiento del usuario a través de preguntas diarias, ejercicios básicos o situaciones para conocer mejor al usuario y poder ajustar el sistema de recomendación. Además de ampliar la aplicación a otros idiomas y configurar el tiempo disponible para realizar estas actividades entre otras propuestas de mejora.

En definitiva, para alcanzar este objetivo principal, se definen los siguientes objetivos parciales:

- Estudio de plataforma Android, librerías que ofrecen reconocimiento facial, análisis de sentimientos y procesamiento de voz a texto y sistemas de dialogo

- Búsqueda de aplicaciones similares en el mercado de sistemas de recomendación, reconocimiento facial, análisis de sentimientos

- Tecnología que se va a integrar gestión de usuarios, procesamiento de texto, análisis de sentimientos, gestión de palabras clave (gramáticas)

- Gestión del sistema de recomendación a través de la selección opciones de ocio y filtro por las preferencias del usuario.

1.3. Fases del desarrollo

La realización de este proyecto se ha estructurado siguiendo la metodología MÉTRICA Versión 3 [52]. Esta metodología sirve de base para el ciclo de vida de un producto software Las fases de desarrollo resultantes son las siguientes:

1.3.1. Fase 1: Análisis /Planificación

- Estudio de los sistemas de diálogo: definición de los sistemas de diálogo y de su arquitectura, estudio de los requisitos que debe cumplir un sistema de diálogo ideal y de las limitaciones a las que están sujetos.

- Estudio de análisis de sentimientos: definición de sistemas que realizan análisis de sentimientos y de su arquitectura, estudio de los requisitos que debe cumplir para realizar un análisis de sentimientos y de las limitaciones a las que están sujetos.

- Estudio de reconocimiento facial: definición de los sistemas con reconocimiento facial y de su arquitectura, estudio de los requisitos que debe cumplir un sistema de reconocimiento ideal y de las limitaciones a las que están sujetos, y presentación de aplicaciones existentes en la actualidad.

- Estudio de sistemas de recomendación: definición de los sistemas de recomendación y de su arquitectura, estudio de los requisitos que debe cumplir un sistema de recomendación y de las limitaciones a las que están sujetos.

- Estudio de la plataforma Android: realizar un estudio exhaustivo de la plataforma Android y del entorno de desarrollo de sus aplicaciones. Análisis de alternativas para el desarrollo de aplicaciones para Android basadas en sistemas de diálogo: análisis de las alternativas que ofrece Android para integrar el reconocimiento de voz, la síntesis de voz en una aplicación, análisis de sentimientos, análisis facial.

- Estudio de aplicaciones para Android basadas en sistemas de diálogo, reconocimiento facial, análisis de sentimientos y sistema de recomendación: presentar ejemplos concretos de asistentes virtuales y otras aplicaciones para Android basadas en sistemas de diálogo y similares a la aplicación que se quiere desarrollar para el presente proyecto.

- Definición de los requisitos funcionales de la aplicación para Android: definir el funcionamiento básico de la aplicación y de las necesidades que debe cubrir sin entrar en detalles.

- Estudio de las tecnologías necesarias para desarrollar la aplicación: estudio de los sistemas de gestión de base de datos, del formato para el intercambio de datos JSON, y de la implementación de algunas operaciones en aplicaciones Android como son la ejecución de tareas en segundo plano, la utilización de mapas, la vista de tarjetas para mostrar los resultados de las actividades.

- Planificación y estimación de costes:

1.3.2. Fase 2: Diseño

Diseño detallado: división de las distintas funcionalidades en los diferentes módulos y submódulos del sistema.

1.3.3. Fase 3: Desarrollo

Programación de la aplicación: programación de la aplicación en Android utilizando Android Studio.

1.3.4. Fase 4: Pruebas y evaluación

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: realización de las preguntas del cuestionario de evaluación de la aplicación basado en un estudio previo acerca de la evaluación de sistemas de recomendación y aplicaciones móviles. Posteriormente, se lleva a cabo la recogida de los datos y finalmente un análisis de los resultados obtenidos.

1.3.5. Fase 5: Cierre

Redacción de la memoria del Proyecto Final de Carrera.
Preparación de la presentación.

La planificación temporal de las fases de desarrollo descritas en el presente apartado se realiza en la Sección 7.1 de esta memoria.

1.4. Estructura de la memoria

A continuación, se resume el contenido de cada capítulo del presente documento:

Capítulo 1: Introducción. Incluye la motivación del proyecto, los objetivos que persigue, las fases de desarrollo, los medios empleados y la estructura de la memoria.

Capítulo 2: Estado del Arte. En este capítulo se realiza un estudio detallado de los sistemas de diálogo, análisis de sentimientos, sistemas de recomendación y reconocimiento facial. Además, se incluyen un análisis de las alternativas que ofrece Android para integrar el reconocimiento de voz, la síntesis de texto a voz, reconocimiento facial, análisis de sentimientos y el sistema de recomendación en una aplicación. El capítulo concluye con la descripción de ejemplos concretos de asistentes virtuales y otras aplicaciones de interés para el presente proyecto.

Capítulo 3: Descripción General del sistema. Este capítulo comienza con una presentación general del sistema. Para ello, se presenta la arquitectura modular del sistema y se realiza una descripción general de cada uno de sus módulos en cuanto a funcionalidad, arquitectura y tecnologías utilizadas. A continuación, se realiza un estudio completo de las tecnologías empleadas en el desarrollo del sistema y se describe la implementación de las operaciones generales.

Capítulo 4: Descripción Detallada de los módulos del sistema. En este capítulo se describe de forma detallada los módulos del sistema en cuanto a su funcionalidad, arquitectura y flujo de datos. Además, se presenta posibles escenarios de uso para cada módulo.

Capítulo 5: Evaluación de la aplicación. Este capítulo comienza con una descripción de la metodología empleada en la evaluación de los sistemas de diálogo multimodales. Posteriormente, se realiza la evaluación de la aplicación multimodal de recomendación desarrollada para Android en el presente proyecto. Para ello, se recoge a través de un cuestionario, basado en la metodología de evaluación descrita previamente, las valoraciones subjetivas de los usuarios de la aplicación.

Capítulo 6: Conclusiones y futuras líneas de trabajo. En este capítulo se exponen las principales conclusiones extraídas de la realización del proyecto y las posibles líneas de trabajo que se podrían generar a partir de este proyecto con el fin de mejorar la aplicación desarrollada.

Capítulo 7: Gestión del proyecto. La Gestión del proyecto incluye la planificación temporal de las tareas en las que se divide el proyecto. Además, se cuadran en el calendario.

Capítulo 8: Presupuesto. La Gestión del proyecto también incluye un análisis de costes y conocer la viabilidad del proyecto. Se desglosa en coste de personal y del recurso que ha sido necesario para llevarlo a cabo. Finalmente se realiza un resumen del coste total.

Glosario. En este apartado se recogen los principales términos utilizados en el presente documento y se acompañan de su respectiva definición o explicación con el objetivo de facilitar su comprensión al lector.

Bibliografía. En este apartado se listan las referencias bibliográficas que se han consultado para la realización tanto del proyecto y de la memoria

Capítulo 2

2. Estado del Arte

El presente capítulo tiene como objetivo contextualizar este Proyecto Fin de Carrera. En primer lugar, se describe el análisis de sentimientos y su arquitectura/ módulos. Del mismo modo, se describen los aspectos fundamentales de la computación afectiva, el reconocimiento facial, los sistemas de recomendación y por último los sistemas de dialogo.

Seguidamente, se realiza una breve descripción las aplicaciones existentes en el mercado de los sistemas anteriormente descritos

Se concluye el capítulo presentando las fases iniciales del ciclo de vida de un proyecto como son los requisitos funcionales de la aplicación, la planificación, recursos a utilizar y presupuesto.

2.1. Análisis

2.1.1. Los sistemas de análisis de sentimientos

El análisis de sentimientos es el proceso de determinar el tono emocional que hay detrás de una serie de palabras, y se utiliza para intentar entender las actitudes, opiniones y emociones expresadas en mención positiva o negativa de un concepto.

Sentimental Analysis o SA es más complejo de lo que parece en un principio ya que hay que analizar el texto en diferentes niveles: nivel global del documento, a nivel de palabra, nivel de frase, nivel de concepto, aspectos relacionados, analizar el sentido.

En el artículo "A survey on opinion mining and sentiment análisis: tasas, approaches and aplicativos" de Kumar Ravi y Vadlamani Ravi [11], indica la complejidad de este proceso con la siguiente imagen:

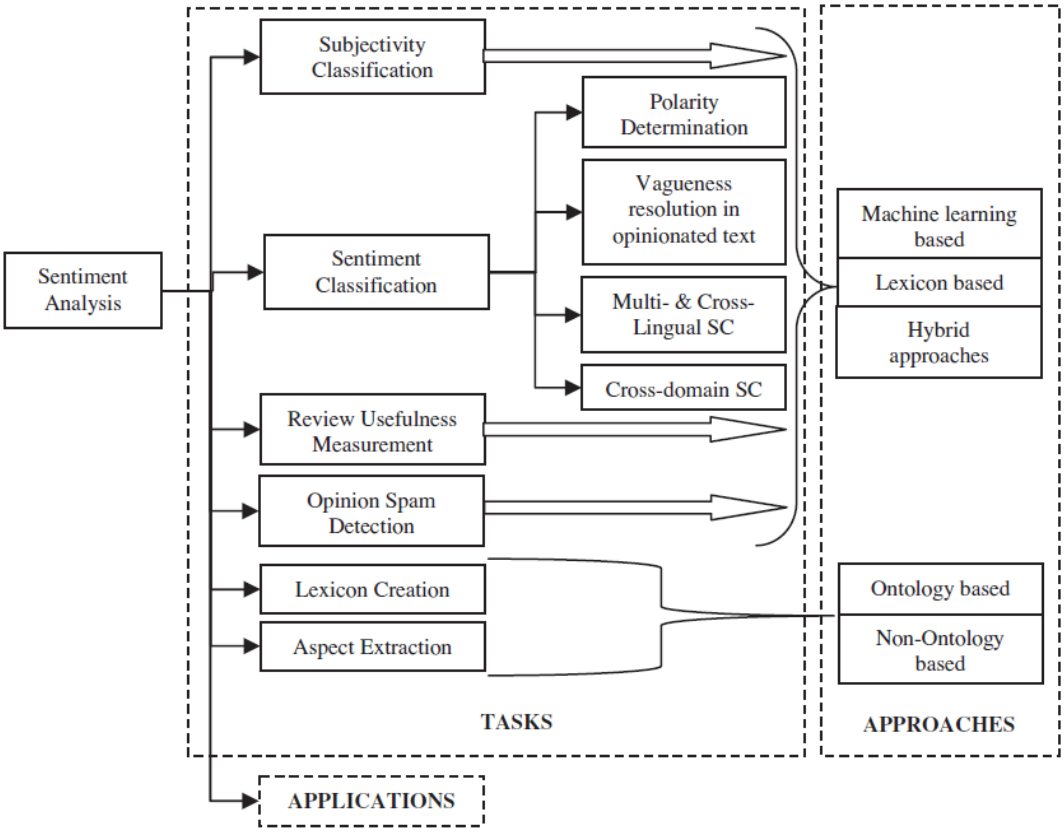


Figura 1: Resumen análisis de sentimientos [11]

En la Figura 1, se puede observar la necesidad de clasificar los sentimientos de un texto a través del aprendizaje automático (Machine learning). Esta técnica implica un alto grado de preprocesamiento a través de entrenamiento y prueba. Se apoya en diccionarios indicando el grado positivo, negativo o neutral de palabras/grupos de palabra. Alguna de las técnicas utilizadas según comenta el artículo, son Decisión

Tree (DT)m, SVM, Neural Network (NN), Naive Bayes y Maximun Entropy(ME). Y el objetivo principal de todas ellas es encontrar patrones en los datos y predecir eventos futuros.

Una vez realizada la clasificación con todo el tiempo computacional que implica, el proceso de aprendizaje no es suficiente. La ironía, el sarcasmo, las metáforas, la ambigüedad… son problemas añadidos al análisis de sentimientos. Técnicas como la frecuencia de aparición de términos, estadística, semántica y características emocionales (SenticNet 3.0) ayudan a este reto. A través de algoritmos como Chi-Cuadrado se puede seleccionar la característica más prominente del texto.

Incluso no todos los idiomas tienen el mismo grado de expresividad. Se utiliza lexicom-based y corpus-base approach para establecer la relación entre idiomas, una clasificación estadística y subjetiva.

El contexto en el que se está escribiendo también influye en el significado y la connotación subjetiva que expresas con una frase o palabra. A través de la medida Frequently Coocurran Entropy(FCE) para generalizar unos tesauros, ontologías, F-mesure.

2.1.2. La computación afectiva

La computación afectiva es tanto el estudio como el desarrollo de sistemas que pueden reconocer, interpretar, procesar y estimular las emociones humanas. Es un campo multidisciplinario que abarca la ciencia, la psicología y la ciencia cognitiva de la computación. Este campo es la evolución del estudio filosófico de la emoción donde se estudia el significado de “afecto” que es un sinónimo de “emoción. Rosalind Picard realiza un estudio muy extenso de la rama más moderna de la ciencia de la computación afectiva [20]. Una motivación para la investigación es la capacidad de simular la empatía. La máquina debe interpretar el estado emocional de los humanos y adaptar su comportamiento a ellos, proporcionándoles una respuesta adecuada para estas emociones.

El estado emocional de un usuario se percibe a través de sensores, un micrófono, cámaras y la lógica del software. Las tecnologías de computación afectiva tras percibir el estado emocional responden mediante funciones de servicios predefinidas y específicas. Una forma de observar la computación afectiva es la interacción entre los humanos y los ordenadores en que un dispositivo tiene la capacidad de detectar y responder adecuadamente a las emociones de su usuario y otros estímulos.

Un dispositivo de computación con esta capacidad puede dar pistas de las emociones del usuario desde una variedad de orígenes. Los cambios de las expresiones faciales, las posturas, la pronunciación, la fuerza o el ritmo de las pulsaciones de las teclas y la temperatura de la mano en un ratón pueden significar cambios en el estado emocional del usuario y los puede detectar e interpretar un ordenador. Una cámara incorporada captura imágenes del usuario y utiliza

algoritmos para procesar los datos y ofrecer información significativa. El reconocimiento de gestos y de voz está entre otras tecnologías que se están explorando para las aplicaciones de computación afectiva.

El reconocimiento de información emocional requiere la extracción de muestras significativas de los datos recopilados. Se realiza mediante técnicas de aprendizaje automático que procesan modalidades diferentes, como el reconocimiento de voz, el procesamiento del lenguaje natural o la detección de expresiones faciales.

Inteligencia emocional artificial o Emoción AI también se conoce como reconocimiento de emociones o tecnología de detección de emociones. En la investigación de mercado esto se conoce comúnmente como codificación facial.

El modo en que una persona se siente no es azaroso, sino que responde a una serie de factores personales y del contexto, los cuales influyen y explican esa manera subjetiva de encontrarse. El comportamiento, incluyendo la manera de sentir, reaccionar, actuar y pensar, es producto de la interacción con el entorno, es decir una persona actúa, siente o piensa en respuesta a un ambiente con el que participa, y que, por tanto, nos presenta situaciones y experiencias que no nos dejan indiferentes, una persona interpreta los hechos, se emociona y actúa ante ellos, y lo hace de un modo diferente.

Una emoción es un modo subjetivo de sentirse ante un hecho, situación o estímulo concreto. Tiene una duración limitada en el tiempo. Cuando estos estados emocionales se prolongan más en el tiempo (horas, días, temporadas), se habla de "Estado Anímico". Para simplificar, se puede clasificar en "Positivos" (Cuando la experiencia subjetiva es agradable, satisfactoria, relajante, placentera) y "Negativos" (La experiencia subjetiva es de malestar, sufrimiento, tensión, desagrado) [5].

Los estados de ánimo surgen como respuesta a nuestras experiencias con el entorno (las situaciones cotidianas, noticias recibidas, la actuación de las personas con las que se relacionan, las consecuencias de sus actos), y pueden estar influidos por otras variables: La hora del día, el clima, la época del año, la alimentación, el estado de salud, la calidad del sueño, el nivel de energía-cansancio. Éstas son variables que pueden ejercer algún efecto en las personas [4]. Hay pocas vivencias cotidianas que le generen a una persona una respuesta emocional neutra; las experiencias suelen provocar respuestas emocionales muy diversas, acordes con la situación (o con nuestro modo de interpretar la situación): enfado, asco, sorpresa, relajación, alegría, tensión, ilusión, impaciencia. Y esto engloba todos los matices y definiciones que cada persona pueda hacer [4]. La respuesta emocional de cada persona ante un mismo hecho es muy subjetiva, pues depende de varios componentes: la respuesta fisiológica (ej. Tensión muscular, presión en el pecho), el componente cognitivo (la manera de interpretar los hechos y nuestras propias reacciones ante los hechos) y el componente motor (una determinada manera de actuar ante esos hechos). Según esto, los estados emocionales o anímicos de una persona pueden reflejarse a través de lo que dicha persona hace o dice, a través de su lenguaje verbal y no verbal, una sonrisa, una mueca tensa, un comentario pesimista, una lágrima, un discurso agresivo, un exceso de actividad motora. Todos ellos le funcionan a la persona o al resto como señales que ayudan a comprender su estado de

ánimo o el del otro y a elegir el mejor comportamiento en consecuencia. Solo hay que saber detectarlos e "interpretarlos".

La inteligencia emocional artificial o Emoción AI también se conoce como reconocimiento de emociones o tecnología de detección de emociones. En la investigación de mercado esto se conoce comúnmente como codificación facial. Hace referencia a un conjunto de habilidades importantes al a hora de manejar adecuadamente las emociones propias o las de otros.

Estas habilidades engloban:

- Detección de la emoción
- Comprensión de la misma (saber interpretarla)
- Expresión adecuada (saber comunicarla)
- Actuación para su regulación (saber cuál es la mejor manera de proceder ante dicha emoción).

Todas ellas son habilidades que se pueden aprender y ejercitar tanto a nivel individual como el terreno social y ello favorecerá un comportamiento más eficaz en las relaciones interpersonales de una persona y en la regulación de sus propios estados. Esto ayuda a que una persona se conozca mejor: a conocer lo que le afecta, de qué modo le afecta y cómo se comporta cuando se siente de una determinada manera. El conocimiento sobre las emociones y sus causas permite a una persona ejercer un mejor control sobre ellas. El primer paso para comprender nuestras emociones es aprender a detectarlas y localizar qué las genera. Sólo aprendiendo a detectar cuando una persona se siente bien o mal se podrá conocer también cómo volver a generar esas emociones (en caso de que sean positivas) o qué hacer para prevenirlas o modificarlas (en caso de que sean negativas) [12]. Para ello, es posible realizar un "autorregistro", que busca asociar las emociones con sus antecedentes. Una vez recopilada dicha información, es posible establecer relaciones sobre los sucesos y actividades y las emociones que provocan [4].

2.1.3. Los sistemas de reconocimiento facial

Los sistemas de reconocimiento facial permiten identificar a una persona analizando las características biométricas de su rostro. Los sistemas basados en los rasgos faciales como ojos, nariz, boca, etc., miden la distancia de los distintos rasgos y mediciones de los ángulos de la cara. Hay otros sistemas más sencillos, que miden la forma global del rostro completo.

El proceso del reconocimiento facial se realiza principalmente en cinco fases:

- Fase de detección: en la que se recoge la imagen del rostro del usuario a identificar a través del dispositivo elegido, ya sea una cámara fotográfica o una cámara de vídeo.
- Preprocesador de la imagen: en esta fase se realizan tareas esenciales para la extracción de la información biométrica, tales como la alineación de la cara respecto a ciertas propiedades geométricas y para hacerla también independiente de la iluminación de la imagen capturada o de la gama de colores obtenida. Mostrado en la Figura 2.

- Fase de extracción de las características faciales: en la que se obtiene la información biométrica de los rasgos faciales, almacenándose esta en un patrón biométrico facial.
- Fase de comparación: En esta fase se compara la información biométrica obtenida con aquellas almacenadas en la base de datos, a través de una comparación 1: N. Los resultados obtenidos indican el porcentaje de similitud del usuario a identificar, con aquellos almacenados en la base de datos
- Fase de toma de decisiones: en la que, utilizando la matriz de similitudes, se identifica al individuo como aquel que mayor porcentaje de similitud ha obtenido, siempre que se encuentre por encima de un umbral determinado.

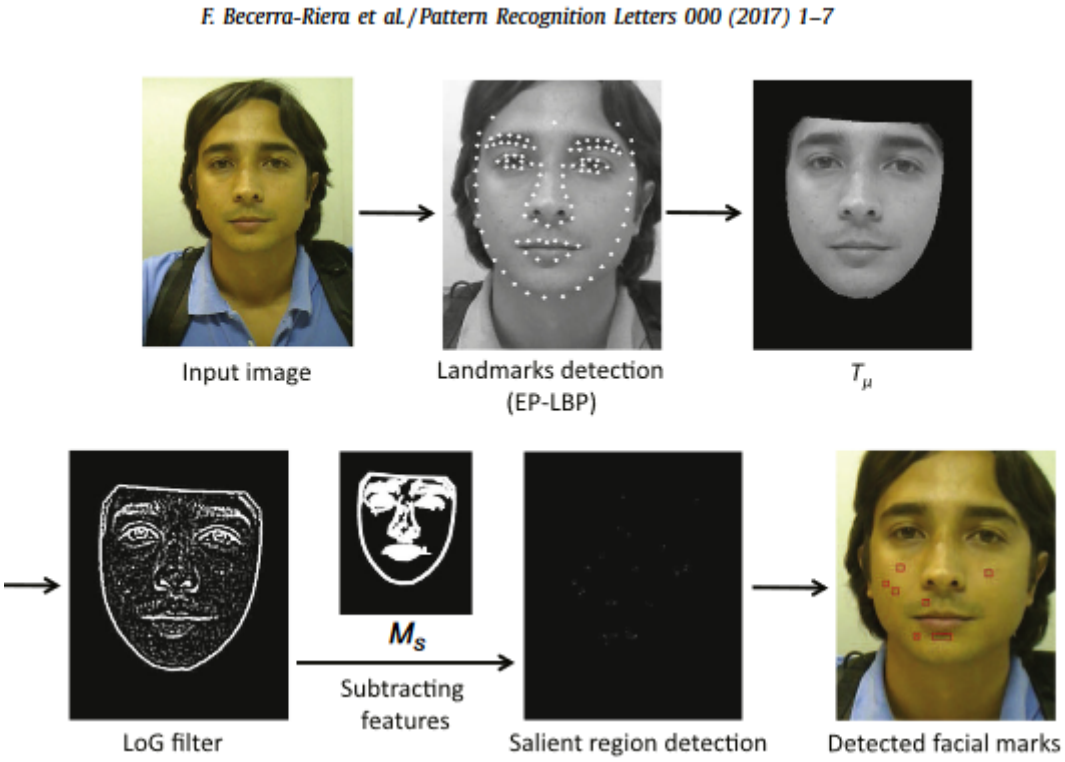


Figura 2: Preprocesado y extracción de las características faciales [43]

Para realizar el reconocimiento facial se utilizan multitud de algoritmos entre los que destacan:

- Eigenfaces: Selección y agranda la parte central de la cara. Con la ayuda de imágenes faciales almacenadas en la base de datos, obtiene la matriz facial [45]

- Fisherfaces 2D [44]: Utiliza el método de Análisis Lineal Discriminante que extrae información discriminante de la cara [46] y el método Análisis de Componentes Principales (PCA) que reduce la dimensión de la cara [46]

Una buena detección es el punto principal para el reconocimiento por lo tanto cualquier factor de iluminación influye en la captura. Incluso la posición de la

cámara o video, que encuadrar bien la cara del usuario y así realizar una mejor detección de los rasgos faciales.

2.1.4. Los sistemas de recomendación

Los sistemas de recomendación forman parte de un sistema de filtrado de información, los cuales presentan distintos tipos de temas o ítems de información (películas, música, libros, noticias, imágenes, páginas web, etc.) que son del interés de un usuario en particular. Generalmente, un sistema recomendado compara el perfil del usuario con algunas características de referencia de los temas, y busca predecir el "ranking" o ponderación que el usuario le daría a un ítem que aún el sistema no ha considerado. Estas características pueden basarse en la relación o acercamiento del usuario con el tema o en el ambiente social del mismo usuario.

Cuando se crea un perfil del usuario, se deben incluir dos métodos para la recolección de características (implícitas o explícitas) [9]. La forma explícita implica que el usuario sea el que responda directamente a la interfaz de usuario. Alguna de las técnicas utilizadas para extraer la información son aquellas donde el usuario tiene que contestar respuestas ponderadas, de preferencia a ciertos elementos de una lista, escoja entre dos opciones. En cuanto a la forma implícita consiste almacenar cualquier tipo de información por parte de la aplicación, que permita obtener información adicional del usuario. Alguna de las técnicas de forma implícita utilizadas son las de guardar un registro con los temas visitados o almacenados comprados y ponderar las visitas a una página entre otras.

Un sistema de recomendaciones es mucho más que un algoritmo o un filtro que selecciona productos con más o menos acierto. Están basados en las preferencias del usuario, en la descripción/ponderación de cada elemento y las preferencias de otros usuarios [37]. Podemos dividir un recomendador en 4 partes [9]: la base de conocimiento (la información, los datos), el procesamiento de la base de conocimientos [37] (Semántica, Machine learning, métodos probabilísticos), la analítica y control de negocio (Feedback algoritmos, estrategia de negocio) y finalmente el interfaz de usuario.

Los sistemas de recomendación normalmente utilizan árboles de decisión, donde cada elemento de la lista de recomendación es ponderado y ordenado en un ranking. Ben Simón [36] indica la importancia de encontrar en la raíz del árbol el factor de decisión más significativo, situando en los nodos inferiores los factores de menos importancia.

En la Figura 3, se muestra el árbol basado en tres factores donde 1 es el más significativo y 3 el menos.

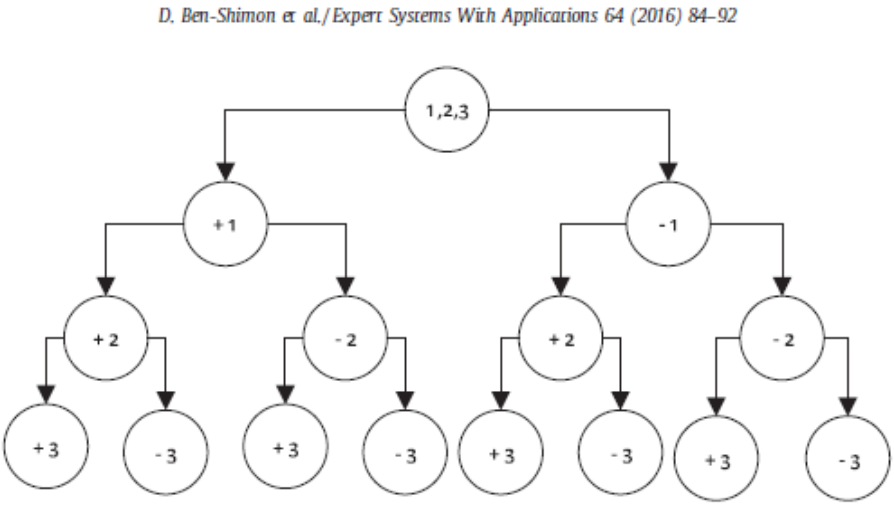


Figura 3: Árbol de decisión

La precisión en obtener el mejor resultado está en la probabilidad de seleccionar un factor como relevante. Influye el número de elementos relevantes que se quieren recomendar (r) y el número total de elementos encontrados (n). Por lo tanto, la precisión está en el resultado de dividir r/n. De aquí la importancia de recomendar un número adecuado de elementos, reducir la lista implica que el usuario no encuentre algún elemento de su agrado y si es excesiva dicha lista se llega a la situación donde el usuario no llega al final de todas las opciones. Un valor adecuado resulta de 5 elementos de muestra.

2.1.5. Los sistemas de diálogo hablado

El Reconocimiento Automático de Voz es un campo de investigación de creciente relevancia. El desarrollo de mejores algoritmos y de modelados más precisos, posibilita la integración de los sistemas de dialogo hombre-máquina a través de la voz en numerosos ámbitos de la sociedad actual. Estos sistemas de dialogo (spoken dialogue systems) permiten el acceso a una gran cantidad de información a través de una forma de comunicación tan natural como es el habla. Estos sistemas informáticos reciben como entrada frases en forma oral del lenguaje natural y generan como salida voz. La finalidad de estos sistemas es la de facilitar un elevado número de servicios interactivos utilizando el teléfono, la televisión o el ordenador como elementos de acceso.

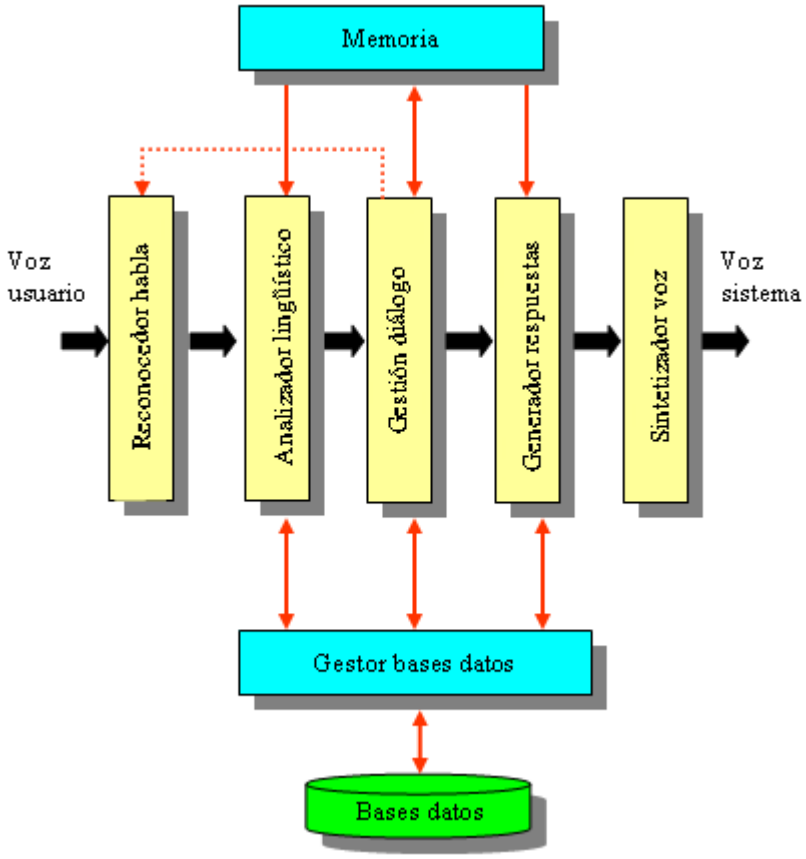


Figura 4: Arquitectura clásica de un sistema de diálogo hablado [1]

La Figura 4 muestra el esquema general de un sistema de diálogo hablado. Estos sistemas están formados por 5 tecnologías que transforman la voz de entrada del usuario en voz del sistema como respuesta, están representadas en la anterior figura con el color amarillo. El primer paso consiste en transformar la señal sonora continua en una representación escrita. El siguiente paso es el de conocer el significado del texto. Una vez interpretada la voz del usuario, el sistema tiene que ofrecer una respuesta. Esta respuesta es transformada a texto y finalmente la conversión a voz de salida. Además, estas tecnologías se apoyan en las bases de datos con información del usuario y la memoria principal del sistema.

Un sistema de dialogo ideal consiste en la naturalidad que supone utilizar el habla en las operaciones de comando y control, la precisión y robustez en la comunicación para diferentes usuarios en diferentes entornos.

El objetivo fundamental de un sistema de diálogo multimodal (multimodal dialogue sistema) es superar las limitaciones de la interacción basada exclusivamente en el habla. En una interacción multimodal el usuario no está restringido a utilizar el habla como único canal de comunicación, sino que puede utilizar varios dispositivos de entrada, como por ejemplo un teclado, un ratón, un micrófono, una cámara, una pantalla sensible al tacto, una PDA, etc. Asimismo, el sistema multimodal puede utilizar diversos canales de salida para proporcionar información al usuario, como, por ejemplo, voz, texto, gráficos o imágenes, con objeto de estimular varios de sentidos del usuario de forma simultánea. Algunos sistemas de diálogo multimodal

permiten incluso que los usuarios puedan elegir entre las diversas modalidades de entrada para llevar a cabo la interacción, permitiendo así una cierta adaptación a las condiciones ambientales de luz, ruido, etc. Esta ventaja permite además que personas con determinadas discapacidades (p. e. personas invidentes) puedan usar estos sistemas mediante alguna de las modalidades de interacción disponibles [39].

2.1.5.1. Reconocimiento Automático del Habla en Android

El lenguaje hablado es una capacidad natural de la especie humana y el habla supone la materialización de esa capacidad a través de una lengua. De todas las posibles formas de comunicación que existen es la más versátil y se manifiesta en todas las sociedades humanas, cosa que no ocurre con la escritura. Conforme aumenta la presencia de aparatos electrónicos en nuestra vida diaria, la interacción con ellos es cada vez más frecuente y sin duda facilitar la comunicación por voz es uno de los grandes retos de la actualidad. Se trata de conseguir que las maquinas más modernas puedan utilizarse a través de la forma de comunicación más antigua: el habla.

Actualmente se trabaja en dos direcciones. La primera es la conversión de voz a texto llamada reconocimiento automático del habla (RAH) y la segunda es comprender el significado del texto a través de bases de conocimiento léxicas, semánticas y modelos de lenguaje conocida como procesamiento del lenguaje natural (PLN)

Para abordar estos problemas hay dos caminos posibles: los sistemas basados en el conocimiento, “knowledge-based”, y los sistemas basados en la estadística, “data-based”. Los primeros buscan, en RAH, que el reconocimiento se lleve a cabo a partir de reglas acústico-fonéticas, que se basan en características de la forma de la onda de entrada. Estas reglas deben ser descritas por un experto humano. Este método ha obtenido resultados muy pobres en RAH y algo mejores en PLN, debido, en gran medida, a la dificultad de los seres humanos de sistematizar su conocimiento para poder implementarlo en un computador.

Los modelos acústicos proporcionan una estimación de la probabilidad de las características observadas en un marco de contexto fonético de una oración dada. Estas características se relacionan generalmente con las mediciones espectrales en un intervalo de tiempo. Un ejemplo de estos modelos se puede estudiar sobre el Modelo oculto de Márkov [40], utilizado por los algoritmos de Google, ya que este modelo es especialmente aplicable al reconocimiento del habla [41].

Desde un principio, Android incorporó la aplicación de búsqueda por voz que viene instalada por defecto en la mayoría de los dispositivos Android. Dicha aplicación muestra inicialmente la ventana con el texto “Hablar ahora” y consecutivamente transfiere el audio a los servidores de Google para que se haga el reconocimiento. El objetivo del Buscador por Voz de Google es reconocer cualquier consulta dictada mediante la voz, es decir, que sea capaz de controlar cualquier solicitud que el buscador de Google pueda manejar. Este hecho es considerado uno de los retos del reconocimiento de voz, ya que el vocabulario y la dificultad de las consultas son demasiado largos

Al estar parte del proceso de reconocimiento de voz en los servidores de Google, es necesario tener acceso a Internet para que la aplicación de búsqueda por voz de Google pueda funcionar. Pero a partir de Android 4.1 Jelly Bean eso cambió, permitiendo usar esta característica sin conexión, siempre y cuando esté activada.

2.1.6. Ejemplo de Aplicaciones

En esta sección, se darán unas pequeñas pinceladas a algunas aplicaciones ya existentes en el mercado de los sistemas comentados anteriormente. Estas aplicaciones son utilizadas a través de dispositivos móviles, como algunas a través de una página web.

2.1.6.1. Aplicaciones que integran análisis de sentimientos.

Hoy en día se están desarrollando aplicaciones donde se analizan los sentimientos de los usuarios como las que se muestran a continuación:

- Mr. tuit:

La primera versión de MrTuit permite conocer la valoración de un comentario en Twitter (positivo, negativo o neutro) y aportar una opinión sobre el acierto

- Moodies:

La aplicación permite descifrar el estado de ánimo del usuario, brinda un "análisis emocional" en 20 segundos con sólo decir algunas palabras al micrófono del Smartphone. Se trata de un autodiagnóstico, y añada un poco de diversión. Esta aplicación agrega una cara de dibujo animado a cada análisis, **y** los usuarios pueden compartir la cara en Facebook, Twitter o a través de un email**.**

2.1.6.2. Aplicaciones del análisis facial

Las aplicaciones del análisis facial están creciendo en los últimos años. Se le está dado usos muy dispares, a continuación, se muestran algunas aplicaciones:

- IObit AppLock :

A los tradicionales patrones de desbloqueo y pines se han ido añadiendo poco a poco otros medios que utilizan datos biométricos para identificarnos y así aumentar la seguridad a la hora de bloquear el teléfono. Esta nueva aplicación para Android permite bloquear o desbloquear el acceso a una aplicación de forma individual con sólo identificar nuestro rostro con la cámara delantera del teléfono. Permite usar el reconocimiento facial para entrar de forma individual en las distintas aplicaciones que tenemos instaladas en el teléfono. Esta aplicación ofrece bloquear el acceso a determinadas aplicaciones mediante el reconocimiento facial, esto quiere decir que podemos seleccionar que el acceso a las redes sociales instaladas, como son Facebook, Twitter o Instagram, sólo sean accesibles mediante reconocimiento facial. Por lo tanto, cada vez que ejecutemos una de estas aplicaciones el teléfono reconocerá nuestro rostro para permitirnos seguir adelante.

- Snapchat:

Esta aplicación móvil utiliza algoritmos para interpretar las características de los objetos en una imagen . Snapchat permite a los usuarios añadir efectos y sonidos en tiempo real. Sin embargo, no puede reconocer que una nariz es una nariz, y no diferencia un rostro de otro.

- Sprinkles:

Es una aplicación de Microsoft que quiere competir con Snapchat. Aprovecha las tecnologías de reconocimiento facial y aplicaciones desarrolladas de inteligencia artificial con las cuales se puede determinar la edad de una persona y a qué famoso se parece el usuario.

- FindFace:

Aplicación rusa que puede identificar a un desconocido mediante el uso de fotografías en menos de un segundo. Con la ayuda de Twitter, la empresa ha creado una base de datos con todas imágenes de las cuentas de la red social y los ha analizado con redes neuronales. reconocimiento facial FindFace tiene una exactitud del 70%. Además, si no se dispone de una fotografía de la persona que se desea encontrar, se puede emplear la imagen de un famoso o de otra persona que se le parezca. La búsqueda ofrece diez rostros similares.

2.1.6.3. Aplicaciones de recomendación

Las diferentes aplicaciones que realizan recomendaciones personalizadas suelen estar ligadas al ámbito emocional. Algunas de las aplicaciones encontradas son:

- Amazon

Amazon es una compañía de comercio electrónico con un catálogo de venta enorme. Maneja un gran volumen de productos por lo que necesita de un sistema de recomendación potente para ayudar a sus usuarios a encontrar los productos que desean. Una buena recomendación es el pilar fundamental de su negocio y motor principal para incrementar sus ventas, por ello guarda información de todos los productos comprados/visitados por el usuario y recomienda aquellos que son similares.

- Minube

La aplicación de Minube ofrece buscar y compartir lugares a la hora de viajar. Se basa en la geo-posición y características del usuario para ofrecer lugares para comer, dormir y visitar.

- FilmAffinity

Es una página web personalizada de votaciones que recomienda series y películas basándose en las valoraciones de usuarios similares.

2.1.6.4. Aplicaciones de los sistemas dialogo, asistentes de voz

Además de la aplicación de búsqueda por voz de Google, han aparecido los asistentes personales: Siri en Apple, Cortana en Windows, Alice en Android. La mayor parte de dichos asistentes de voz integran el reconocimiento de voz de Google para conectarse a los servicios de voz de Google, aunque algunos de ellos utilizan servicios de voz propios.

La mayor parte de los asistentes de voz integran además síntesis de texto a voz. Gran parte de dichos asistentes, lo único que hacen es utilizar el motor de transformación de texto a voz (TTS) para realizar las lecturas. Algunos de los principales asistentes son:

- Assistant (Speaktoit)

Es un asistente virtual que utiliza lenguaje natural para contestar preguntas, buscar información, iniciar aplicaciones y conectarte con servicios web. Actualmente, está integrado con los servicios web: Google, Twitter, Facebook, Foursquare, Evernote, Yelp, TripAdvisor, Wikipedia, Chacha, IMDB, Eventful, News360, Amazon, Gmail, Google Images, Google Calendar, Google Maps, Google Navigation, y Waze

No responde a las preguntas, pero sí que tiene un buen sistema de predicción. También hace llamadas, manda SMS, notifica ciertas acciones del móvil entre otras acciones. Además, se puede personalizar con frases o chistes

- Dragon Mobile Assistant

Dragon Mobile Assistant es una herramienta básica, pero incorpora "Attentive Mode", que permite tener la aplicación activada cuando la pantalla del móvil está bloqueada. El usuario puede elegir entre varios tipos de voces y nombres para este asistente personal.

- Google Now

Google Now es una de las aplicaciones más importantes y con una gran cantidad de utilidades, tanto básicas como avanzadas. Es el asistente personal desarrollado por Google que utiliza una interfaz de usuario de lenguaje natural para responder preguntas, hacer recomendaciones y realizar acciones mediante la delegación de las solicitudes a un conjunto de servicios web. Además, tiene una parte de sistema recomendación que ofrece información al usuario en función de sus hábitos de búsqueda.

- Indigo Virtual Assistant

La característica más notable que posee esta aplicación es el hecho de que la voz que se emite desde tu móvil lo hace con un tono de conversación mucho más normal del que estamos acostumbrados en los asistentes personales. Esta aplicación permite continuar la misma conversación en diferentes dispositivos, accede a Internet

para buscar la respuesta a cualquier pregunta, actualiza estados en redes sociales como Facebook y Tweeter, escucha recomendaciones de restaurantes, dirige al usuario a lugares cercanos, guarda notas y recordatorios, envía y lee SMS y correos, realiza llamadas y reproduce videos de YouTube entre otras opciones.

2.2. Requisitos Funcionales/Casos de uso

Tras realizar un análisis exhaustivo de los objetivos de la aplicación y de los aspectos necesarios que tienen que cumplir los sistemas de recomendación, reconocimiento facial, análisis de sentimientos y sistemas de dialogo, se ha completado el estudio de mercado de aplicaciones similares a una aplicación para la recomendación de actividades de acuerdo con el reconocimiento de emociones del usuario.

En este punto, se describen los aspectos imprescindibles que tiene que cumplir la aplicación con todo este conocimiento adquirido. Se formaliza a través de los requisitos funcionales descritos a continuación. Además, se incluyen los criterios de aceptación que deben cumplir la aplicación final.

Referencia	Tipo de Requisito	Descripción del Requisito	Criterios de Aceptación
PersPla.01	Premisa	Aplicación en Android Studio. API 6.0	Probar que funciona la aplicación en un móvil con la versión 6.0 instalada
PersPla.02	Funcional	Contenga reconocimiento de voz	Hablar a la aplicación y que muestre alternativas de actividades relacionadas con lo que has hablado
PersPla.03	Funcional	Permita al usuario activar la cámara para realizar la detección facial y reconocimiento de emociones	Tras realizar la captura de la cámara, se muestra el estado anímico de la persona
PersPla.04	Funcional	Procesar el texto obtenido para sacar palabras clave.	Tras procesar lo hablado a la aplicación, comprobar que las palabras claves propuestas son acordes
PersPla.05	Funcional	Procesar el texto obtenido y realizar	Tras procesar lo hablado a la aplicación,

		un análisis de sentimientos	comprobar que el análisis es acorde
PersPla.06	Funcional	Procesar toda la información obtenida del usuario y proponer actividades a realizar	Revisión de las actividades propuestas
PersPla.07	Funcional	Realizar un cuestionario donde el usuario pueda seleccionar sus preferencias	Personalización correcta y verificación del correcto funcionamiento de la aplicación
PersPla.08	Funcional	Permitir que el video dure 5min máximo	Comprobar visualmente: Si no se para la cámara antes por petición del usuario, la captura finaliza al superar el Max de 5 min

Tabla 1: Requisitos funcionales de la aplicación

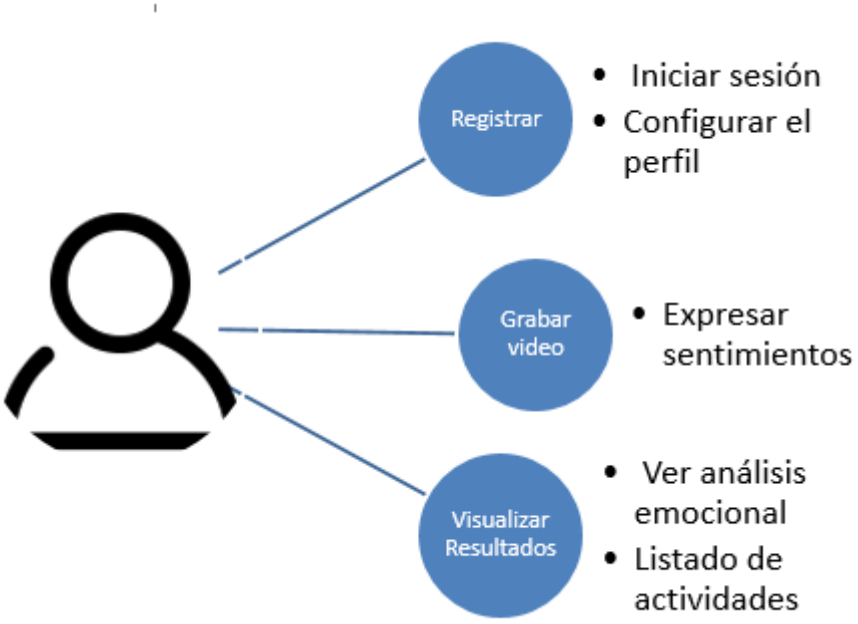


Figura 5: Casos de uso

2.3. Planificación

A continuación, se introducen las duraciones de cada tarea, considerando los días de la semana como días laborables de lunes a viernes, y estableciendo una jornada de trabajo de 3 horas.

Las fases que se han tenido en cuenta son las principales de un ciclo de vida de un proyecto, mencionados en el apartado 1.3 del presente documento.

Para la planificación de este proyecto se utiliza la herramienta de Microsoft Project para incluir las tareas a realizar, recursos necesarios y tiempo estimado. Esta herramienta genera el diagrama de Gantt que permite de forma visual incluir las tareas en un calendario además de las vinculaciones entre tareas. Este diagrama es muy útil para llevar un control sobre las estimaciones iniciales y las desviaciones en tiempo y recursos a lo largo del proyecto.

Por lo tanto, una vez establecidas las fases y tareas del Proyecto Fin de Carrera, se utiliza el mencionado diagrama de Gantt que vincula dichas fases y tareas a un calendario, para poder realizar un seguimiento detallado de las mismas. En el capítulo 7, se incluye el detalle de la planificación realizada.

2.4. Recursos

Para el desarrollo del presente Proyecto Final de Carrera se han empleado los siguientes medios:

Recursos Hardware

- Ordenador portátil Asus

Figura 6: Características del PC utilizado

- Tablet Android LENOVO TB3.850F
- Cable USB.

Recursos Software

- Entorno de desarrollo Android Studio.
- JDK (Java Development kit): kit de desarrollo de Java.
- Servicio de alojamiento de archivos multiplataforma Dropbox.
- Microsoft Project 2010
- Microsoft Word 2010
- Microsoft Excel 2010

Estos recursos tienen un coste asociado que se tienen en cuenta a la hora de calcular el presupuesto del proyecto. En el siguiente apartado se hace un breve resumen del presupuesto del proyecto y en el Capítulo 8, se detalla en profundidad el mismo.

2.5. Presupuesto

En esta sección se realiza el cálculo del presupuesto del proyecto. Ya se ha establecido los objetivos a alcanzar, la planificación del proyecto para llevarlos a cabo y los recursos disponibles para lograr dichos objetivos en el plazo marcado. Todos estos datos son los que influyen en el presupuesto del proyecto. En el Capítulo 8 del presente documento, se realiza un estudio detallado de los costes de personal, coste de equipo y costes indirectos necesarios para realizar la aplicación.

Capítulo 3

3. Descripción general del sistema

En este capítulo se muestra una visión global del sistema, comentando con las posibilidades que se ofrecen al usuario. En los siguientes apartados se examinan las herramientas utilizadas para llevar a cabo el trabajo. Finalmente, la documentación de este capítulo ofrece el esquema de diseño utilizado y cómo interactúan los diferentes módulos.

3.1. Presentación del sistema

La aplicación consiste en un sistema de recomendación de ocio que ofrece actividades al usuario acordes a lo que la aplicación multimodal puede captar del usuario en ese momento a través de la cámara, del lenguaje y de los gestos junto con las preferencias del perfil.

La intención de la aplicación es animar al usuario con varias propuestas de ocio según los parámetros obtenidos a través del móvil y sus preferencias sin tener que estar buscando en todas las propuestas que hay en internet. El fin de esta aplicación es simplificar esta tarea al usuario y ofrecerle alternativas que se ajusten al usuario. Además, la aplicación tiene un potente factor motivador y terapéutico, ya que incluye un diario que va almacenando las emociones detectadas del usuario. Una vez realizado el análisis facial y el análisis de sentimientos, PersonalOcio muestra los

sentimientos detectados para que el usuario se dé cuenta de cómo se encuentra. Reconocer la emoción es el primer paso para poner remedio, tal y como se ha descrito en el apartado de computación afectiva de este documento. El sistema de recomendación se ajusta al estado de ánimo del usuario. Por ejemplo, si esta triste uno de los objetivos de la aplicación es anímale y las propuestas que se le ofrecen van destinadas a ese fin.

3.2. Tecnologías utilizadas

En la presente sección se analizan las tecnologías utilizadas en el desarrollo de la aplicación para Android. Puesto que la aplicación realizada para el presente Proyecto Final de Carrera se ha implementado para la plataforma Android, se va a dedicar el primer apartado al estudio de dicha plataforma. Además, se explica el proceso que se ha seguido para preparar el entorno de desarrollo de la aplicación.

3.2.1. JAVA

Java es un lenguaje de programación orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web.

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

En realidad, Java en sí mismo tiene poco que ver con Android, aunque algo hay. El funcionamiento de las aplicaciones en Android, de cualquier tipo (fuera del propio sistema operativo), hace que estas corran sobre una máquina virtual llamada DALVIK. Esta máquina es la responsable de que no sea necesario diseñar las aplicaciones específicamente para cada teléfono (aunque haya que prestar algo de atención a elementos como la pantalla para que todo funcione un poco mejor), sino que solamente haya que diseñarlos para Android, y que sea ella quien se encargue de decirle a nuestro sistema qué dispositivos deben ser utilizados en cada momento.

Esta máquina, aunque es compatible con JAVA, no es esa misma plataforma estrictamente hablando, pero a la hora de desarrollar, es prácticamente lo mismo para

muchas cosas e incluso existen múltiples herramientas para hacer conversiones de JAVA a Android. Comparte el mismo principio de máquina virtual.

3.2.2. Plataforma Android

En general, la estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework de aplicaciones orientado a objetos sobre el núcleo de las bibliotecas (API) en una máquina virtual mencionada en el apartado anterior, con el nombre de Dalvik, con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (Surface manager), un framework Open Core, una base de datos relacional SQLite, una Interfaz de programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado Web Kit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic.

En la Figura 6, se muestra la arquitectura del sistema operativo Android. Se divide en cinco capas que se detallan en profundidad en las siguientes secciones.

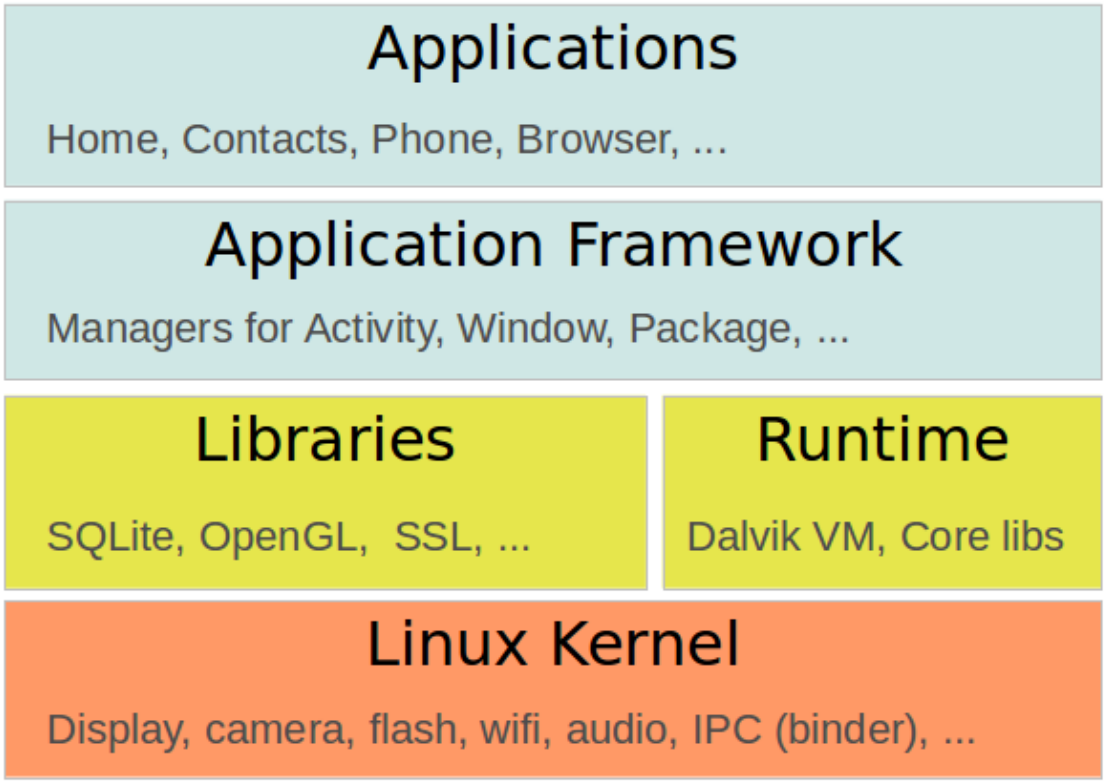


Figura 7: Arquitectura Android [16]

3.2.2.1. Kernel de Linux

El núcleo del sistema operativo Android está basado en el kernel de Linux en su versión 2.6, similar al que puede incluir cualquier distribución de Linux, como Ubuntu, solo que adaptado a las características del hardware en el que se ejecutará Android, es decir, para dispositivos móviles.

La elección de Linux 2.6 es debida principalmente a dos razones: la primera, su Naturaleza de código abierto y libre se ajusta al tipo de distribución que persigue Android; la segunda es que este kernel de Linux incluye de por sí numerosos drivers, además de contemplar la gestión de memoria, gestión de procesos, módulos de seguridad, comunicación en red y otras muchas responsabilidades propias de un sistema operativo.

El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura. El desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores. De esta forma también nos evitamos el hecho de quebrarnos la cabeza para conocer las características precisas de cada teléfono. Si necesitamos hacer uso de la cámara, el sistema operativo se encarga de utilizar la que incluya el teléfono, sea cual sea. Para cada elemento de hardware del teléfono existe un controlador (o driver) dentro del Kernel que permite utilizarlo desde el software.

El Kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación, etc.

3.2.2.2. Librerías

La siguiente capa que se sitúa justo sobre el Kernel la componen las bibliotecas nativas de Android, también llamadas librerías. Están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Estas normalmente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma "más eficiente".

Entre las librerías más importantes de este nivel, se pueden mencionar las siguientes:

- LIBC: La librería libc incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.

- Surface Manager: La librería Surface Manager es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.

- OpenGL/SLy SGL representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware

encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.

- La librería Media proporciona todos los códec necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)

- FreeType, permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.

3.2.2.3. Entorno de ejecución

Como podemos apreciar en la Figura 6, el entorno de ejecución de Android no se considera una capa en sí mismo, dado que también está formado por librerías. Aquí encontramos las librerías con las funcionalidades habituales de Java, así como otras específicas de Android.

El componente principal del entorno de ejecución de Android es la máquina virtual Dalvik. Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja de esto es que las aplicaciones se compilan una única vez y de esta forma estarán listas para distribuirse con la total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación.

Cabe aclarar que Dalvik es una variación de la máquina virtual de Java, por lo que no es compatible con el bytecode Java. Java se usa únicamente como lenguaje de programación, y los ejecutables que se generan con el SDK de Android tienen la extensión. dex (Dalvik Executable) que es específico para Dalvik, y por ello no podemos correr aplicaciones Java en Android ni viceversa.

3.2.2.4. Framework de aplicaciones

La siguiente 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 de las capas anteriores a través de la máquina virtual Dalvik.

Se encuentran los siguientes módulos:

- Activity Manager: Se encarga de administrar la pila de actividades de nuestra aplicación, así como su ciclo de vida.

- Windows Manager: Se encarga de organizar lo que se mostrará en pantalla. Básicamente crea las superficies en la pantalla que posteriormente pasarán a ser ocupadas por las actividades.

- Content Provider: Esta librería es muy interesante porque crea una capa que encapsula los datos que se compartirán entre aplicaciones para tener control sobre cómo se accede a la información.

- Views: En Android, las vistas los elementos que nos ayudarán a construir las interfaces de usuario: botones, cuadros de texto, listas y hasta elementos más avanzados como un navegador web o un visor de Google Maps.

- Notification Manager: Engloba los servicios para notificar al usuario cuando algo requiera su atención mostrando alertas en la barra de estado. Un dato importante es que esta biblioteca también permite jugar con sonidos, activar el vibrador o utilizar los leds del teléfono en caso de tenerlos.

- Package Manager: Esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes. Con paquete nos referimos a la forma en que se distribuyen las aplicaciones Android, estos contienen el archivo .apk, que a su vez incluyen los archivos .dex con todos los recursos y archivos adicionales que necesite la aplicación, para facilitar su descarga e instalación.

- Telephony Manager: Con esta librería podremos realizar llamadas o enviar y recibir SMS/MMS, aunque no permite reemplazar o eliminar la actividad que se muestra cuando una llamada está en curso.

- Resource Manager: Con esta librería podremos gestionar todos los elementos que forman parte de la aplicación y que están fuera del código, es decir, cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos o rayos. En un post relacionado a la estructura de un proyecto Android veremos esto más a fondo.

- Location Manager: Permite determinar la posición geográfica del dispositivo Android mediante GPS o redes disponibles y trabajar con mapas.

- Sensor Manager: Nos permite manipular los elementos de hardware del teléfono como el acelerómetro, giroscopio, sensor de luminosidad, sensor de campo magnético, brújula, sensor de presión, sensor de proximidad, sensor de temperatura, etc.

- Cámara: Con esta librería podemos hacer uso de la(s) cámara(s) del dispositivo para tomar fotografías o para grabar vídeo.

- Multimedia: Permiten reproducir y visualizar audio, vídeo e imágenes en el dispositivo.

3.2.2.5. Aplicaciones

En la última capa se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, las nativas (programadas en C o C++) y las administradas (programadas en Java), las que vienen preinstaladas en el dispositivo y aquellas que el usuario ha instalado. En esta capa encontramos también

la aplicación principal del sistema: Inicio (Home) o lanzador (launcher), porque es la que permite ejecutar otras aplicaciones mediante una lista y mostrando diferentes escritorios donde se pueden colocar accesos directos a aplicaciones o incluso widgets, que son también aplicaciones de esta capa.

3.2.3. Fundamentos de la creación de aplicaciones Android

Las aplicaciones Android deben estar escritas en el lenguaje de programación Java. El SDK de Android, o mejor dicho las herramientas del SDK de Android compilan el código, junto con los datos y archivos de recursos en un paquete de Android, en un archivo comprimido con extensión '.apk'. Todo el código queda reducido a un solo archivo '.apk' que se considera en sí mismo la aplicación, y es este archivo el que se utiliza para la instalación de aplicaciones en el dispositivo.

Una vez instalada en el dispositivo la aplicación corre en su propio recinto o entorno limitado de seguridad:

- El sistema operativo Android es un sistema multi-usuario de Linux en el que cada aplicación es un usuario diferente.

- Por defecto, el sistema asigna a cada aplicación de una única ID de usuario de Linux (el ID es utilizado únicamente por el sistema y desconocido para la aplicación). El sistema establece permisos para todos los archivos en una aplicación para que sólo el ID de usuario asignado a esa aplicación puede acceder a ellos.

- Cada proceso tiene su propia máquina virtual, por lo que el código de una aplicación se ejecuta de forma aislada de otras aplicaciones.

Por defecto, cada aplicación se ejecuta en su propio proceso de Linux. Android lo inicia cuando alguno de los componentes de la aplicación se ejecuta, a continuación, cierra el proceso cuando ya no se necesita o cuando el sistema debe recuperar la memoria para otras aplicaciones, este caso en concreto lo explicaremos más en detalle en el ciclo de vida de una actividad, ya que es un rasgo distintivo de Android.

Con esto, el SO Android implementa el principio de privilegios mínimos, que consiste en que cada aplicación, por defecto, sólo puede acceder a los componentes requeridos para hacer su trabajo. Esto genera un entorno muy seguro en el que la aplicación en cuestión no puede acceder a partes del sistema para las cuales no se le ha otorgado permiso.

De cualquier modo, existen manera para que una aplicación pueda compartir datos con otras aplicaciones, o también que, una aplicación pueda acceder a los servicios del sistema:

- En el caso en el que dos aplicaciones compartan el mismo ID de usuario de Linux, son capaces de acceder cada una a los archivos de la otra. Para ahorrar recursos del sistema, en este caso en el que dos aplicaciones tienen el mismo ID de usuario también se pueden organizar para ejecutar en el mismo proceso de Linux y

compartir la misma máquina virtual (para ello las solicitudes deben estar firmadas con el mismo certificado).

- Una aplicación puede solicitar permiso para acceder a los datos del dispositivo, tales como los contactos del usuario, mensajes SMS, el almacenamiento externo (tarjeta SD), cámara, Bluetooth, y mucho más.

Todos los permisos de la aplicación deben ser autorizados por el usuario durante la instalación.

3.2.3.1. Componentes de las Aplicaciones Android

A continuación, se expondrán los fundamentos para el desarrollo de una aplicación en Android. Los principales componentes son:

- Actividades (Activistas): Cada pantalla de una aplicación. Utilizan Vistas(Views) como componentes que muestran información y responden a las acciones del usuario

- Servicios (Services): Componentes de la aplicación que se ejecutan deforma invisible, actualizando los datos y las Actividades, y disparando Notificaciones. Realizan el procesamiento normal de la aplicación que debe continuar incluso cuando las Actividades de la aplicación no están visibles.

- Proveedores de Contenidos (Content Providers): Almacenes de datos compartidos. Gestionan las Bases de Datos para las aplicaciones.

- Intenciones (Intente): Mecanismo que permite el paso de mensajes destinados a ciertas Actividades o Servicios, o a todo el sistema (Intenciones de broadcast). Exponen la intención de que se haga algo. El sistema determinará el destinatario que lo efectuará.

- Receptores de Broadcast (Broadcast Receivers): Los crean las aplicaciones como consumidores de las Intenciones de broadcast que cumplan ciertos criterios.

- Notificaciones (Notifications): Mecanismo que permite a las aplicaciones señalar algo a los usuarios sin interrumpir la Actividad en primer plano.

3.2.3.2. Intents

Los Intent son utilizados para lanzar otros componentes, representan la intención de realizar una acción. Pueden iniciar una Activity mediante el método **startActivity**, un *Broadcast Receiver* mediante broadcastIntent, arrancar un Service gracias a startService o comunicarse con él a través de bindService. Es decir, son capaces de lanzar otras actividades, pero también de generar eventos que pueden ser capturados desde otra aplicación.

Pueden encontrarse dos tipos de Intent:

- Intent explícito: especifican el componente invocado gracias a *subcomponente(ComponentName)* o *setClass(Context, Class)*.
- Intent implícito: no especifican un componente para realizar la acción, en su lugar, deben proveer suficiente información para que el sistema pueda decidir cuál de los componentes disponibles se adapta mejor al Intent ejecutado.

También, permiten pasar variables de una actividad a otra a través de los métodos *putExtra y get<tipo>Extra* siendo tipo String, Float, Int, Byte… Incluso se puede pasar un objeto de tipo Parcelable o Serialize.

A continuación, en la Figura 7 se muestra un ejemplo de la actividad origen, donde a través de la función putExtra se incluye el identificador del usuario las métricas faciales obtenidas tras realizar el video y el número de veces que se ha solicitado la detección de la cara para poder hacer la media de las mediciones.

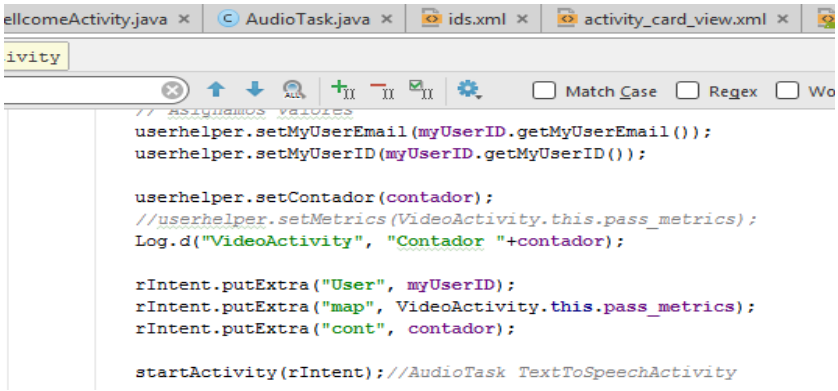


Figura 8: Actividad origen (intent)

En cambio, en la actividad destino como muestra la Figura 8, se recogen las variables de identificación de usuario, el mapa de métricas y el contador de peticiones de detección a través del *getParcelableExtra y el getSerializableExtra*.

Figura 9: Actividad destino (Intent)

3.2.3.3. El Manifiesto de una Aplicación Android (Android Manifest)

Toda aplicación desarrollada en Android incluye un fichero de Manifiesto, el AndroidManifest.xml. Este fichero define la estructura de la aplicación y sus

componentes. Incluye un nodo raíz y un nodo para cada uno de sus tipos de componentes. A través de filtros de intenciones determina como interactuará la aplicación en cuestión con otras aplicaciones.

Algunos de los nodos más importantes serán:

- Nodo raíz *manifest*. Incluye el nombre del paquete de la aplicación.

- Nodo *application*. Indica metadatos de la aplicación (título, icono, tema, etc.) y contiene los nodos de actividades, servicios, proveedores de contenido y receptores de broadcast.

- Nodo *uses-permission*. Declara loes permisos que la aplicación necesita para operar. Estos permisos serán presentados al usuario durante la instalación para que los acepte o deniegue, estos permisos serán necesarios para infinitud de servicios nativos de Android, como enviar SMS, hacer llamadas, uso de servidor, etc.

- Nodo *permission*. Define un permiso que se requiere para que otras aplicaciones puedan acceder a partes restringidas de la aplicación. Las otras aplicaciones necesitarán poner un uses-permission en su Manifiesto para utilizar este permiso.

- Nodo *instrumentation*. Permite definir test de ejecución para las Actividades y Servicios.

3.2.3.4. Creación y destrucción de Aplicaciones y Actividades (Ciclo de vida)

Las aplicaciones Android no son iguales a las aplicaciones en los sistemas operativos tradicionales, en Android sólo hay una aplicación en primer plano que normalmente estará ocupando toda la pantalla. Las aplicaciones estarán formadas por Actividades (pantallas).

Al arrancar una nueva aplicación, pasa a primer plano situando una Actividad encima de la que hubiera, formando se así una pila de actividades. En el momento en el que el usuario presiona el botón “back”, se cierra la actividad en primer plano y recupera la de la cima de la pila. Las aplicaciones Android no tienen control ninguno sobre su propio ciclo de vida, esto implica que deben estar pendientes de posibles cambios en su estado y reaccionar como corresponda. En particular deben estar preparadas para su terminación o destrucción en cualquier momento.

En general, cada aplicación se ejecuta en un proceso que ejecuta uña instancia de Dalvik. El runtime de Android gestiona el proceso de cada aplicación, y por extensión de cada actividad que contenga.

La actividad se puede encontrar en los siguientes estados:

- Activo (*Running*): La actividad está encima de la pila, es visible, tiene el foco (recibe la entrada del usuario). Cuando otra actividad pase a estar activa, esta pasará a estar pausada.

- Pausado (*Paused*): La actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad transparente o que no ocupa toda la pantalla. Cuando una actividad es tapada por completo pasa a estar parada.

- Parado (*Stopped*): Cuando la actividad no es visible. Permanece en memoria reteniendo su estado. Cuando una actividad entra en parada puede ser bueno que salve todos sus datos y el estado de la interfaz de usuario.

- Destruido (*Destroyed*): Cuando la actividad termina, o es matada por el runtime de Android. Sale de la pila de actividades. Necesita ser reiniciada para volver a estar activa.

Y del mismo modo existen una serie de métodos de transición entre unos estados y otros:

- onCreate (*Bundle*): Se invoca cuando la Actividad se arranca por primera vez. Se utiliza para tareas de inicialización a realizar una sola vez, como crear la interfaz de usuario de la Actividad. Su parámetro es null o información de estado guardada previamente por onSaveInstanceState ().

- *onStart* (): Se invoca cuando la Actividad va a ser mostrada al usuario.

- *onResume* (): Se invoca cuando la Actividad va a empezar a interactuar con el usuario.

- *onPause* (): Se invoca cuando la actividad va a pasar al fondo porque otra actividad ha sido lanzada para ponerse delante. Se utiliza para guardar el estado persistente de la Actividad

- *onStop* (): Se invoca cuando la actividad va a dejar de ser visible y no se necesitará durante un tiempo. Si hay escasez de recursos en el sistema, este método podría no llegar a ser invocado y la Actividad ser destruida directamente.

- *onRestart* (): Se invoca cuando la Actividad va a salir del estado de parada para volver a estar activa.

- *onDestroy* (): Se invoca cuando la Actividad va a ser destruida. Si hay escasez de recursos en el sistema, este método podría no llegar a ser invocado y la Actividad ser destruida directamente.

- *onSaveInstanceState*(*Bundle*): Se invoca para permitir a la actividad guardar su estado, por ejemplo, la posición del cursor en una caja de texto. Normalmente no necesita ser redefinido porque la implementación de la clase activity ya guarda todo el estado de todos los componentes de la Interfaz de Usuario.

- *onRestoreInstanceState*(*Bundle*): Se invoca para recuperar el estado guardado por *onSaveInstanceState* (). Normalmente no necesita ser redefinido porque la implementación de la clase activity ya recupera todo el estado de todos los componentes de la Interfaz de Usuario.

La Figura 8 muestra las relaciones descritas para la creación y destrucción de actividades y aplicaciones.

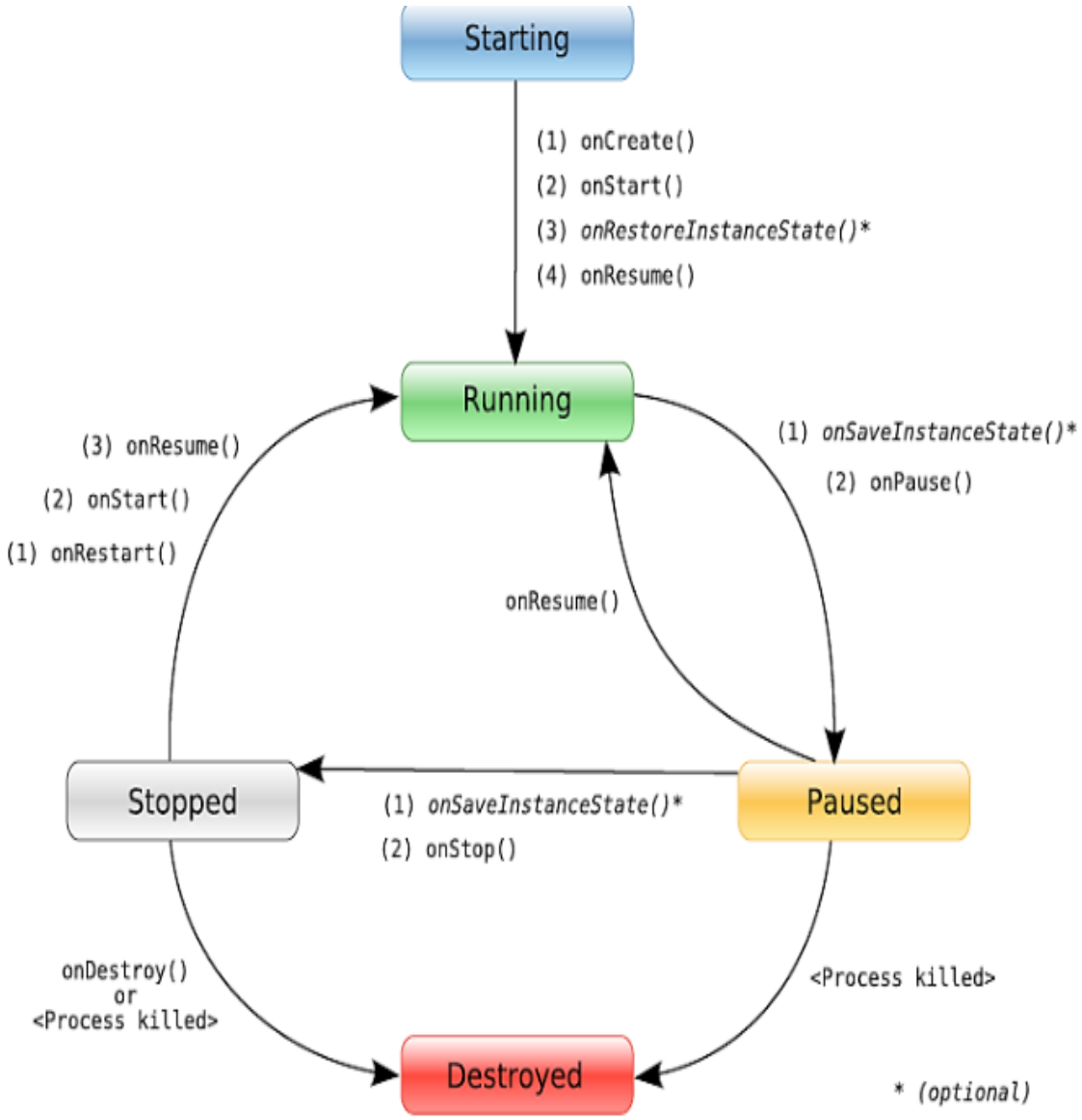


Figura 10: Ciclo de estados de las actividades [18]

El proceso que mantiene a la actividad puede ser eliminado cuando se encuentra en *onPause* () o en *onStop* (), es decir, cuando no tiene el foco de la aplicación. Android nunca elimina procesos con los que el usuario está interactuando en ese momento. Una vez se elimina el proceso, el usuario desconoce dicha situación y puede incluso volver atrás y querer usarlo de nuevo.

Entonces el proceso se restaura gracias a una copia y vuelve a estar activo como si no hubiera sido eliminado. Además, la actividad puede haber estado en segundo plano, invisible, y entonces es despertada pasando por el estado *onRestart ()*.

Obviamente, los recursos son siempre limitados, más aún cuando se está hablando de dispositivos móviles. En el momento en el que Android detecta que no hay los recursos necesarios para poder lanzar una nueva aplicación, analiza los procesos existentes en ese momento y elimina los procesos que sean menos prioritarios para poder liberar sus recursos.

Cuando el usuario regresa a una actividad que está dormida, el sistema simplemente la despierta. En este caso, no es necesario recuperar el estado guardado porque el proceso todavía existe y mantiene el mismo estado. Sin embargo, cuando el usuario quiere regresar a una aplicación cuyo proceso ya no existe porque se necesitaba liberar sus recursos, Android lo crea de nuevo y utiliza el estado previamente guardado para poder restaurar una copia fresca del mismo. El usuario no percibe esta situación ni conoce si el proceso ha sido eliminado o está dormido.

3.2.3.5. Recursos de las aplicaciones Android

Una aplicación para Android se compone de algo más que código que requiere de recursos que están separados del código fuente, como imágenes, archivos de audio y todo lo relativo a la presentación visual de la aplicación. Por ejemplo, se deben definir las animaciones, menús, estilos, colores y el diseño de interfaces de usuario de la actividad con archivos XML.

Para todos estos recursos Android proporciona un identificador único dentro de la aplicación, que puede utilizarse para hacer referencia al recurso en el código de aplicación o de otros recursos definidos en XML.

Hay distintos tipos de recursos, que se definen en ficheros XML alojados en la subcarpeta denominada “res”:

- Valores simples (carpeta valúes): Cadena de caracteres tipo String, colores y dimensiones. Cada uno de ellos se definen como un elemento. Cada fichero XML contiene la definición de uno o más de estos elementos. Todos estos recursos se identifican con el valor de su atributo name.

- Recursos dibujables (carpeta drawable): Ficheros con imágenes, incluyendo el icono de la aplicación. Son usados para ficheros de bitmaps o de imágenes “estirables” (. 9.png). Estos recursos se identifican con su nombre de fichero, y los recuadros de color con el valor de su atributo name.

- Animaciones (carpeta anim): Usados para animaciones sencillas sobre uno o varios gráficos: rotaciones. Fading, movimiento, etc. Cada animación se define en un fichero xML. Se identifican con su nombre de fichero.

- Menús (carpeta menu): Existen tres tipos de menús: de opciones, contextual y submenú. El menú de opciones y el menú contextual se identifican con su nombre de fichero y el submenú con el valor de su atributo id.

- Diseños (carpeta layout): Cada layout se define en un fichero XML. Dentro del layout se definen los elementos que lo componen, como puedan ser los

Views o los ViewGroups. Se identificará por su nombre de fichero y los elementos del layout se podrán identificar con su atributo id.

- Estilos (carpeta valúes): Un estilo es uno o más atributos que se aplican a un elemento. El tema se define como uno o más atributos que se aplican a todo lo que hay en pantalla, este se asigna como atributo a una actividad en el Manifiesto. El estilo se referencia con el valor de su atributo name.

- Varios (carpeta xml): Usado para recursos con características especiales.

3.2.3.6. Procesos y prioridades

Tal y como se ha explicado, cada aplicación Android se ejecuta en su propio proceso. Este proceso se crea cada vez que una aplicación necesita ejecutar parte de su código, y seguirá existiendo hasta que la aplicación finalice o hasta que el sistema necesite utilizar parte de sus recursos para otra aplicación considerada prioritaria.

El orden en que los procesos se van matando para liberar recursos lo determina la jerarquía que Android establece evaluando la case de componentes que están ejecutándose y el estado de los mismos, en orden de importancia serán:

1. Prioridad critica:
- Procesos activos o en primer plano: Los que están interactuando con el usuario, hay muy pocos y se les mata solo como último recurso.

2. Prioridad alta:
- Procesos visibles: Visibles, pero no activos, por ejemplo, tapados parcialmente por otra actividad, se les destruirá solo en situaciones extremas.
- Procesos de servicio arrancados: Procesamiento que debe continuar, aunque no tenga interfaz visible. Se les matara solo en casos extremos.

3. Prioridad baja:
- Procesos de fondo o en segundo plano: Alojan actividades que no caen en las categorías anteriores. Hay muchos de estos procesos. Se les mata cuando se necesitan recursos para el resto de procesos.
- Procesos vacíos: Android mantiene en memoria las aplicaciones que han terminado, por si son relanzadas. Se destruirán estos procesos rutinariamente.

3.2.3.7. Ejecución asíncrona con AsyncTask

La implementación de todas las tareas que requieran de un tiempo de ejecución considerable en la aplicación Android desarrollada debe ejecutarse de manera asíncrona por medio de la clase Android *AsyncTas*k. Si no se realiza de esta manera, el hilo principal de ejecución queda bloqueado durante un tiempo superior al establecido por Android y se muestra en pantalla el mensaje de error Application Not Responding, lo que provoca el cierre de la aplicación por parte del sistema.

Para ejecutar cualquier operación por medio de AsyncTask es necesario crear una nueva clase que extienda de AsyncTask y sobrescriba sus métodos:

- *onPreExecute* () – Este método se ejecuta antes del código principal que se ejecutará de forma asíncrona. Es útil para realizar todas las operaciones previas como la inicialización de mensajes informativos sobre el progreso de la operación.

- *doInBackground* () – El método más importante de las clases AsyncTask, donde se realizan las operaciones en segundo plano.

- *onPostExecute* () – Es llamado de forma automática cuando finalizan las acciones llevadas a cabo en el método doInBackground (). Su invocación representa el momento idóneo para cerrar procesos abiertos o dar lugar a nuevos eventos.

En la aplicación se ha utilizado esta técnica en varias ocasiones para lanzar procesos secundarios de conversión de voz a texto, guardar el audio mientras se está capturando las métricas faciales, lanzar varios analizadores del texto entre otros.

En la Figura 10, se muestra el ejemplo de tarea asíncrona para convertir el audio de voz almacenado a texto. Se puede ver que extiende de la clase AsyncTask y contiene los principales métodos descritos anteriormente *doInBackground y onPostExecute.*

En la función doInBackground, llama a su vez a startSpeechToText que realiza todo el envío del audio y recepción del texto convertido. El método *startSpeechToText* inicializa los componentes que realizan la petición HTTP a Google Speech, indicando la URL del servicio, el tipo de petición y como parámetro el audio procesado y encriptado en base64. Este encriptado es uno de los requisitos impuestos por la librería y que se describirá detalladamente en el apartado 3.2.11.2 de este documento. Por último, se trata el objeto JSON recuperado para extraer el texto convertido.

El segundo método, *onPostExecute* habilita de nuevo el botón para continuar con la siguiente actividad de análisis cuando el usuario lo pulse. Y cierra la barra de progreso que indicaba al usuario el proceso de análisis.

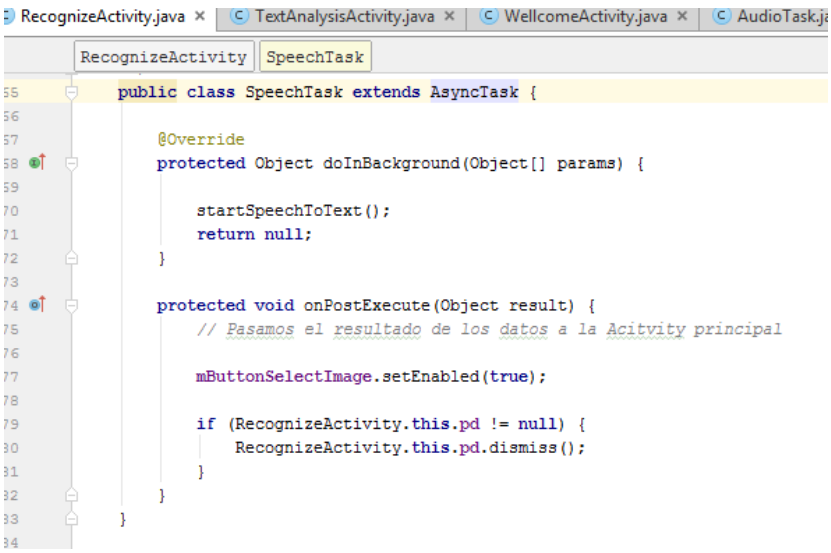


Figura 11: AsyncTask SpeechToText

Mencionar que existe librerías con métodos síncronos o asíncronos para poder utilizar uno u otro según la aplicación lo requiera. El api de Google Speech permite esta posibilidad con cierta limitación en cuanto a tiempo de procesado de audio que se ampliará en el apartado 3.2.11.2.

3.2.3.8. Ejecución asíncrona con Thread

Android proporciona directamente crear hilos secundarios dentro de los cuales se puede ejecutar las operaciones más costosas. Esto lo conseguimos en Android instanciando un objeto de la clase Thread. El constructor de la clase Thread recibe como parámetro un nuevo objeto Runnable que es necesario construir implementando su método *run* (), dentro del cual se va a realizar nuestra tarea de larga duración. En la Figura 12, se puede observar la creación del hilo y la utilización de la función run para iniciar el proceso de almacenamiento del audio.

Hecho esto, se debe llamar al método *start* () del objeto Thread definido para comenzar la ejecución de la tarea en segundo plano.

Figura 12: Hilo para guardar el audio

3.2.4. Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado, llamado también IDE (siglas en inglés de *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

3.2.4.1. Justificación IDE escogido

El primer IDE que se ha venido utilizando desde la aparición de Android para el desarrollo de sus aplicaciones ha sido Eclipse. Google proporciona de forma gratuita el SDK o Kit de Desarrollo de Software, y los plug-in para el Entorno de Desarrollo Integrado (IDE) Eclipse. Pero en junio de 2013 Google sacó a la luz su propio IDE llamado Android Studio.

Ambos IDEs proporcionan un marco de desarrollo amigable que nos asiste en todas las etapas de la creación de una aplicación para Android, desde su programación en código, pasando por su emulación mediante un emulador que simula el comportamiento de la aplicación en un dispositivo real, hasta la firma digital de la aplicación para su posterior difusión al mundo. Este proyecto se realiza por Android Studio por diversas razones:

- Es el futuro y el más utilizado hasta ahora para aplicaciones móviles

- De esta forma, el IDE de Android, llegaba con novedades como *Instant Run*, una función que nos permite revisar todos los cambios que hagamos al código casi al instante en un emulador, pudiendo editar código y ver las consecuencias casi al instante.

- Su particularidad más reseñable podría ser su nueva forma de construir los apk. Más serio, más versátil, más potente, más actual, y más parecido a un proyecto en java. Esto es debido a que utiliza Gradle

- Android Studio destaca, sobre todo, con su nueva actualización, por ser más rápido escribir código en él, (de 2 a 2,5 veces más rápido), y actualizar código puede ser hasta 50 veces más rápido.

- Destaca por la facilidad de probar el código en distintos smartphones en base a su tamaño de pantalla, y ver cómo sienta nuestra aplicación en los distintos terminales. El código es más rápido, y es más fácil gestionar cada error**.**

- Herramientas Lint (detecta código no compatible entre arquitecturas diferentes o código confuso que no es capaz de controlar el compilador) para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones.

- Utiliza ProGuard para optimizar y reducir el código del proyecto al exportar a APK (muy útil para dispositivos de gama baja con limitaciones de memoria interna).

Los pasos para tener operativo el IDE con su SDK están claramente explicados en la página web de Android www.android.com en el apartado para desarrolladores (Developers), concretamente para el caso de Android Studio en el siguiente enlace:
http://developer.android.com/sdk/installing/studio.html, con lo que se comenta en el proceso de instalación en esta memoria.

3.2.5. XML

XML (eXtensible Markup Language) es uno de los formatos más utilizados para intercambiar información entre aplicaciones de diferentes plataformas. Son ficheros de texto donde los campos o elementos de información que contienen se delimitan mediante pares de etiquetas.

La idea de los lenguajes de etiquetado (o de marcas) que permitían almacenar documentos de información de forma estructurada surge con el Lenguaje de Etiquetado General (GML) desarrollado por IBM. Este fue origen del estándar SGML. a partir del cual, la W3C desarrolló las especificaciones del XML (*eXtensible Markup Language*).

Gran parte de su éxito radica en que es un estándar abierto que permite almacenar y organizar cualquier clase de información en un formato ajustado a las necesidades particulares de un entorno o contexto concreto (por ejemplo, tratar específicamente libros de literatura general o documentación técnica o información logística -pedidos, facturas, - o de cualquier otro ámbito de actividad). Está pensado como un sistema de intercambio de información completamente abstracto e independiente de cualquier plataforma y origen, pudiendo por ejemplo utilizarlo entre gestores de bases de datos distintos implementados en sistemas operativos diferentes.

Desde un punto de vista más técnico, se trata de un metalenguaje, es decir, un lenguaje que nos permite describir y definir otros lenguajes de marcado, que sean adecuados a unos objetivos o usos determinados. Hablamos por tanto de una especificación, un conjunto de reglas, más que de un lenguaje concreto. Hay muchos ejemplos conocidos de lenguajes/formatos basados en XML, como XHTML, SVG, MathML, RSS o RDF.

Android utiliza archivos XML denominados layouts que definen la interfaz gráfica con la que interactúa el usuario. Android permite también utilizar código Java para definir la interfaz, pero el uso de layouts basados en XML proporciona independencia entre funcionalidad y vista.

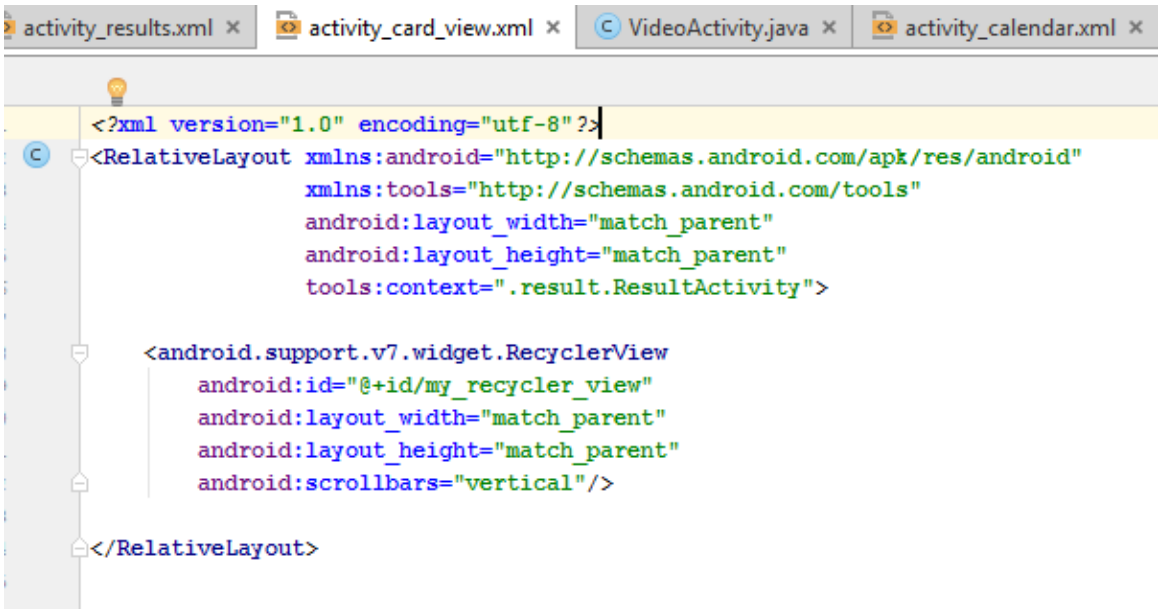


Figura 13: Ejemplo de layout XML

En la Figura 13, se muestra un layout relativo de la clase utilizada para mostrar las actividades. Contiene un elementa CardView descrito en el apartado 3.2.8

3.2.6. HTTP

Los servidores web siguen un estándar internacional llamado *Hipertext Transfer Protocol* HTTP. Este protocolo consiste en reglas sencillas de transferencia de recursos o archivos entre equipos interconectados a una red.

La comunicación se establece a través de una petición de envío, la cual contiene los datos del cliente, como el sistema operativo que usa, el navegador web desde donde se hace la petición, la ubicación del archivo solicitado (URL), etc. Este cliente lanza la petición al servidor que contiene el recurso o el espacio para almacenar.

Una petición puede tener múltiples objetivos dependiendo del método que se elija. Los tipos de peticiones más comunes son el retorno de datos y la publicación de datos. Técnicamente se les conoce como los métodos GET y POST.

Una aplicación necesita tener permisos de Internet y acceso al estado de la red para poder tener acceso a estas conexiones GET/POST de un dispositivo. Estos permisos se incluyen en el archivo AndroidManifest.xml:

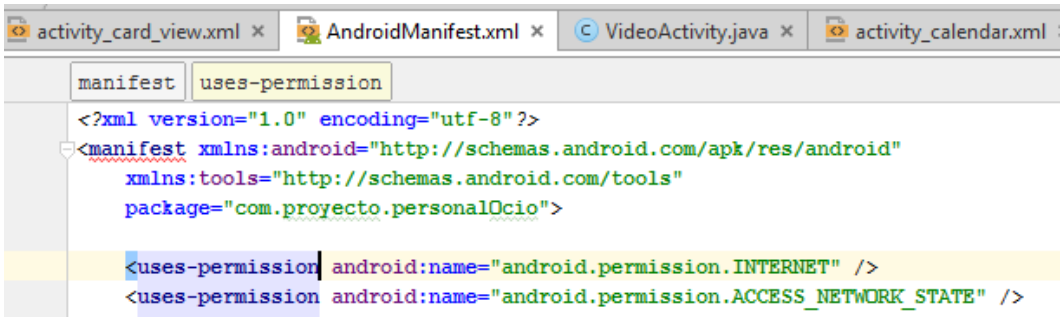


Figura 14: Permisos de la aplicación para conexión HTTP

La clase HttpUrlConnection del paquete java.net.* permite a nuestros dispositivos Android asumir las características de un cliente HTTP ligero. Con esta clase podremos recibir y enviar información a través de la web, lo que potenciará nuestras aplicaciones Android. A continuación, se muestran los pasos que se deben realizar para establecer una conexión.

El primer paso para iniciar la comunicación es abrir la conexión hacia el recurso alojado en el servidor. Tal y como se muestra en la Figura 15, se utiliza el método *openConnection* () de la clase URL. El resultado que se obtiene debe ser asignado a HttpUrlConnection para instanciar el cliente:

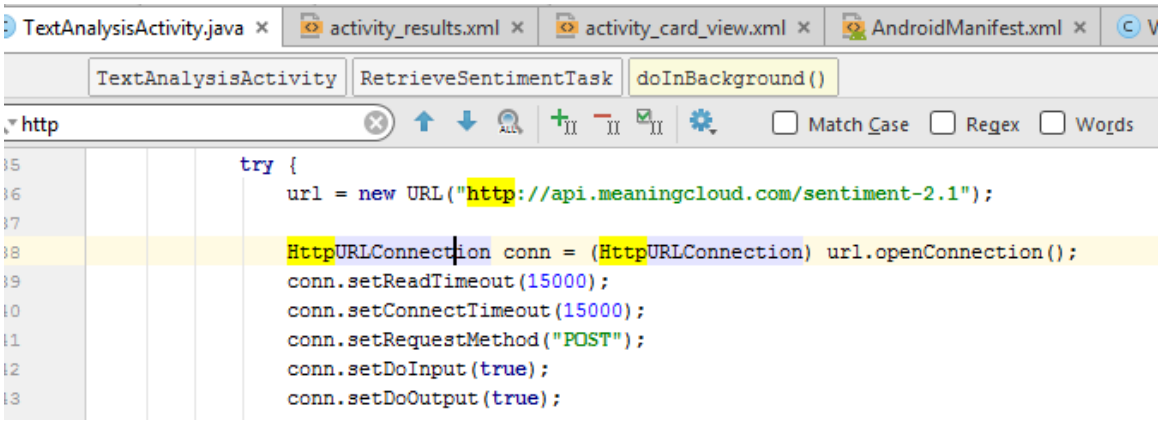


Figura 15: HttpUrlConnection *openConnection ()*

Posteriormente se añaden a los parámetros de la url a la query y se abre un buffer de escritura indicando el tipo de datos, en este caso UTF-8 como se puede ver en la Figura 16.

Figura 16: HttpURLConnection query

Finalmente, se realiza la conexión a través de la funcion *connect*(). En este punto la petición al servidor esta completada. El siguiente paso es recuperar la respuesta con la funcion getInputStream y tratar los datos recibidos como se puede visualizar en la Figura 17.

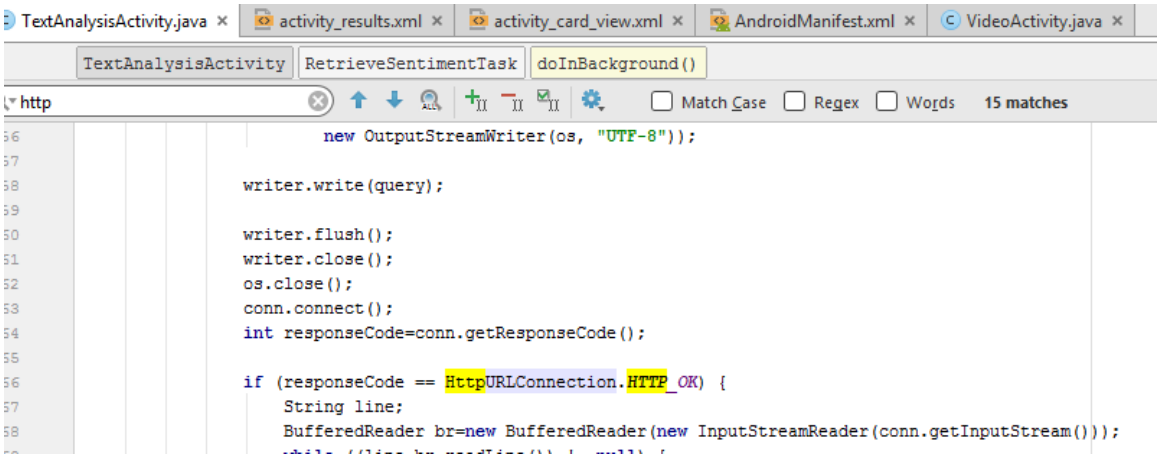


Figura 17: HttpURLConnection connect

3.2.6.1. Okhttp3

La librería Okhttp3 se utiliza para la conexión con los servidores de análisis de texto de manera similar a HttpUrlConnection. Al comienzo del presente proyecto se utilizó esta librería para la conexión con los servidores de análisis de texto. Ofrece la ventaja de realizar de forma transparente un control de errores en la red y una gestión de las conexiones. Sin embargo, tras realizar la incorporación en la aplicación y varias pruebas se llegó a la decisión de no incluirla en la aplicación final. El motivo es el de no aportar ninguna ventaja en el rendimiento.

La Figura 18 muestra el código que es similar a los pasos mencionados con el uso del HttpUrlConnection:

1- Iniciar la comunicación: En este caso, se realiza a través del objeto OKHttpClient

2- Construir la petición: Se utiliza RequestBody para incorporar los parámetros de la petición junto con el tipo de datos. Se añaden estos parámetros, la URL del servidor y las cabeceras de la petición al objeto Request. En el ejemplo mostrado en la figura, se trata de la petición realizada al servidor Meaning Cloud para la clasificación del texto. Se crea la estructura RequestBody con la clave del servidor, el texto a analizar, el lenguaje y el modelo de datos que se va a utilizar.

3- Realizar la conexión: Se lanza la petición a través de una nueva llamada al que se le pasa la petición y la función execute. La forma de realizar este proceso es el siguiente: *client. newCall(request). execute ()*

4- Obtener la respuesta: En este caso, no es necesario realizar ningún tratamiento adicional. La respuesta se devuelve como resultado de la llamada anterior realizada en el paso 3.

```
package com.proyecto.personalOcio.utilidades.text;


import java.io.IOException;
import java.io.Reader;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import java.util.List;

/**
 * Created by s2054cv on 23/08/2016.
 */
import com.google.gson.Gson;

import android.os.AsyncTask;
import android.util.Log;
import com.squareup.okhttp.MediaType;
import com.squareup.okhttp.OkHttpClient;
import com.squareup.okhttp.Request;
import com.squareup.okhttp.RequestBody;
import com.squareup.okhttp.Response;

public class TextProcess extends AsyncTask<String, String, String>{

    private Exception exception;
    public String processed="";
    public String key;

    public TextProcess(String _key){
        super();
        key=_key;
    }

    @Override
    protected String doInBackground(String... texto) {
        OkHttpClient client = new OkHttpClient();

        MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");
        RequestBody body = RequestBody.create(mediaType, "key="+key+"&lang=es&txt=" + texto + "&model=
        Request request = new Request.Builder()
                .url("http://api.meaningcloud.com/sentiment-2.1")
                .post(body)
                .addHeader("content-type", "application/x-www-form-urlencoded")
                .build();
        try {
            Response response = client.newCall(request).execute();
|          // Deserialize HTTP response to concrete type.
               Gson gson=new Gson();
               DescriptionText desc = gson.fromJson(response.body().string(), SentimentBean.class);
               processed = desc.toString();
```

Figura 18: Utilización Okhttp3

3.2.7. JSON

JSON (JavaScript Object Notation) es un formato para el intercambio de datos, básicamente describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. Se ideó como una alternativa a XML y, una de las mayores ventajas que tiene su uso es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre

distintas tecnologías. Además, debido a su naturaleza y al ser más compacto suele ser mucho más rápido trabajar con JSON antes que con XML.

Los distintos tipos de datos permitidos en este formato se pueden visualizar fácilmente en la Figura 19.

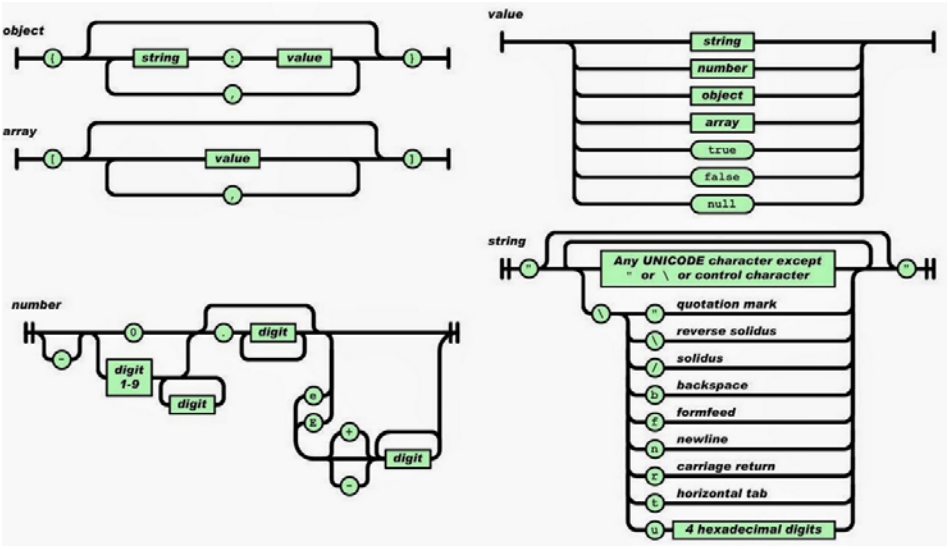


Figura 19: Estructura JSON

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un array asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa con arrays, vectores, listas o secuencias.

Los tipos de valores pueden ser de la siguiente forma:

- Objeto: conjunto desordenado de pares nombre/valor. Un objeto comienza con "{" y termina con "}". Cada nombre es seguido por ":" y los pares nombre/valor están separados por ",".
- Un array es una colección de valores. Un array comienza con "[" y termina con "]" . Los valores se separan por ",".
- Un valor puede ser una cadena de caracteres con comillas dobles, o un número, o true o false o null, o un objeto o un array. Estas estructuras pueden anidarse.
- Una cadena de caracteres es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape.
- Un carácter está representado por una cadena de caracteres de un único carácter. Una cadena de caracteres es parecida a una cadena de caracteres C o Java.
- Un número es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales.

En la Figura 20, se muestra un ejemplo del JSON de respuesta obtenido tras analizar los sentimientos de un texto. Se puede observar los distintos tipos de estructuras. En primer lugar, aparece un objeto llamado "Status" que contiene el campo "code" con su valor a "0", y tres campos más "msg", "credits" y "remaining_credis". El resto de elementos son del tipo cadena/valor salvo los denominados sentence_list, sentimented_entity_list, y sentiment_concept_list que corresponde a estructuras tipo array. Por ejemplo, el primero de ellos, sentences_list contiene 2 elementos separados por comas. Cada elemento del array a su vez contiene 10 campos más del tipo simple (cadena/valor), tipo objeto o tipo array.

Figura 20: Respuesta JSON (MeaningCloud Sentiment)

En una aplicación Android ya se ha comentado como se envían consultas HTTP, pero es necesario recuperar la respuesta de la petición y poder tratarla. Las respuestas suelen estar en el formato JSON. A continuación, se describe la forma de recuperar, tratar y poder usar los datos recibidos en este formato.

En primer lugar, se utiliza la clase JSONObject para cargar la respuesta recibida en el campo "body" de una respuesta HTTP.

Y a partir de aquí se puede recuperar el valor de objeto a través de la función getXXX y el nombre [8] . En la Tabla 2 se muestra los métodos utilizados en la aplicación.

Método	Descripción
JSONArray getJSONArray (Sting name)	Devuelve el array correspondiente al nombre pasado por parámetro o una excepción si no existe

JSONObject getJSONObject (Sting name)	Devuelve el objeto correspondiente al nombre pasado por parámetro o una excepción si no existe
String getString (String name)	Devuelve como cadena el objeto correspondiente al nombre pasado por parámetro o una excepción si no existe

Tabla 2: Métodos para recuperar el valor del JSON

Si se trata de un array es algo más complejo ya que se necesita un objeto JSONArray que carga la cadena completa de array. Una vez recuperado el array, hay que recorrer toda la lista para obtener cada elemento del mismo.

A continuación, en la Figura 21, se muestra un ejemplo utilizado para recuperar el análisis de sentimientos de un texto. Se puede observar cómo se obtiene el valor tipo String de los campos simples "score_tag" y "irony" a través del objeto JSONObject. Además, se extraen los datos de la estructura array del campo denominado "sentimented_concept_list". Una vez obtenido el objeto JSONArray se recorren todos sus elementos a través de un bucle for en este caso, para obtener los campos y valores de cada uno de ellos.

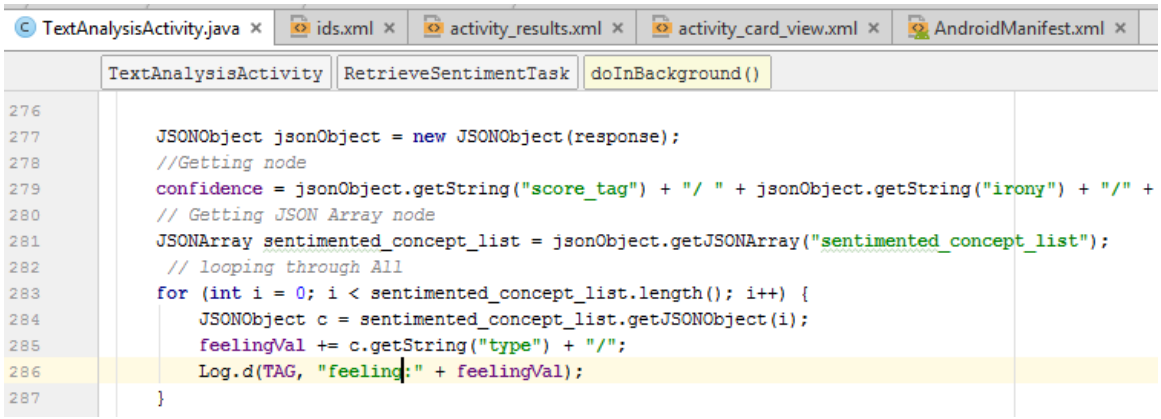


Figura 21: Ejemplo JSONObject y JSONArray (Meaning Cloud Sentiment)

3.2.7.1. GSON

La biblioteca Gson nació como un proyecto interno de Google para su propio uso. Google Gson es una biblioteca de código abierto para el lenguaje de programación Java que permite la serialización y deserialización entre objetos Java y su representación en notación JSON. Finalmente se decidió publicarla bajo una licencia Apache License 2.0.

La diferencia principal respecto al JSON original es la de agilizar todo el parseo de los datos a través de una simple llamada en lugar de recorrer todas las estructuras del JSON y extraer cada una de ellas de forma manual. Es neceario crear una clase

con todos los campos de la respuesta y con la ayuda de la funcion fromJson. De esta forma, parsea la respuesta de la petición HTTP de manera transparente. Se descartó finalmente este tratamiento, ya que se trata de un JSON muy extenso y la aplicación PersonalOcio únicamente requiere la extracción de unos pocos datos y no es necesario un objeto completo del mismo.

```
public TextProcess(String _key){
    super();
    key=_key;
}

@Override
protected String doInBackground(String... texto) {
    OkHttpClient client = new OkHttpClient();

    MediaType mediaType = MediaType.parse("application/x-www-form-urlencoded");
    RequestBody body = RequestBody.create(mediaType, "key="+key+"&lang=es&txt=" + texto + "&model=general");
    Request request = new Request.Builder()
            .url("http://api.meaningcloud.com/sentiment-2.1")
            .post(body)
            .addHeader("content-type", "application/x-www-form-urlencoded")
            .build();
    try {
        Response response = client.newCall(request).execute();
       // Deserialize HTTP response to concrete type.
          Gson gson=new Gson();
          DescriptionText desc = gson.fromJson(response.body().string(), SentimentBean.class);
          processed = desc.toString();
       |

        Log.i("TextProcessTextProcess","Procesado"+processed);

    } catch ( UnsupportedEncodingException e) {
        Log.e("TextProcess:", e.toString());

    }catch ( IOException e) {
        Log.e("TextProcess:", e.toString());
    }
    return processed;
}
@Override
protected void onPostExecute(String post){
    super.onPostExecute(post);
    Log.i("TextProcess","Procesado4:"+processed);

    }
public String getProcessed() {
    return processed;
}
```

Figura 22: Uso Gson (MeaningCloud Class)

En una primera versión de aplicación se utilizo la librería Gson como forma de parsear la respuesta de las peticiones. En la Figura 22, se muestra su utilización al recuperar la respuesta en la clasificación de texto. Se creo la clase SentimentBean con toda la estructura que devuelve el servicio de clasificación de texto. Una vez creado el objeto Gson se llama al método *fromJson* pasando como parametros *SentimentBean.class* y *response.body().string()*.

3.2.8. Card View

El componente llamado CardView es la implementación que nos proporciona Google del elemento visual en forma de tarjetas de información que tanto utiliza en muchas de sus aplicaciones, entre ellas Google Now, quizá la que más a ayudado a popularizar este componente.

Hasta la llegada de Android 5.0, para utilizar estas “tarjetas” en la interfaz de nuestras aplicaciones teníamos que recurrir a librerías de terceros o bien trabajar un poco para implementar nuestra propia versión. Sin embargo, las tenemos disponibles desde hace varias versiones en forma de nueva librería de soporte oficial, proporcionada junto al SDK de Android.

La implementación de este tipo de vista es muy sencilla, sin embargo, su adaptación para que tenga la propiedad de ser seleccionada a través de un click es algo más compleja. Es necesario una adaptador del RecyclerView (Figura 22) y una clase DataObject que contenga los elementos de la tarjeta

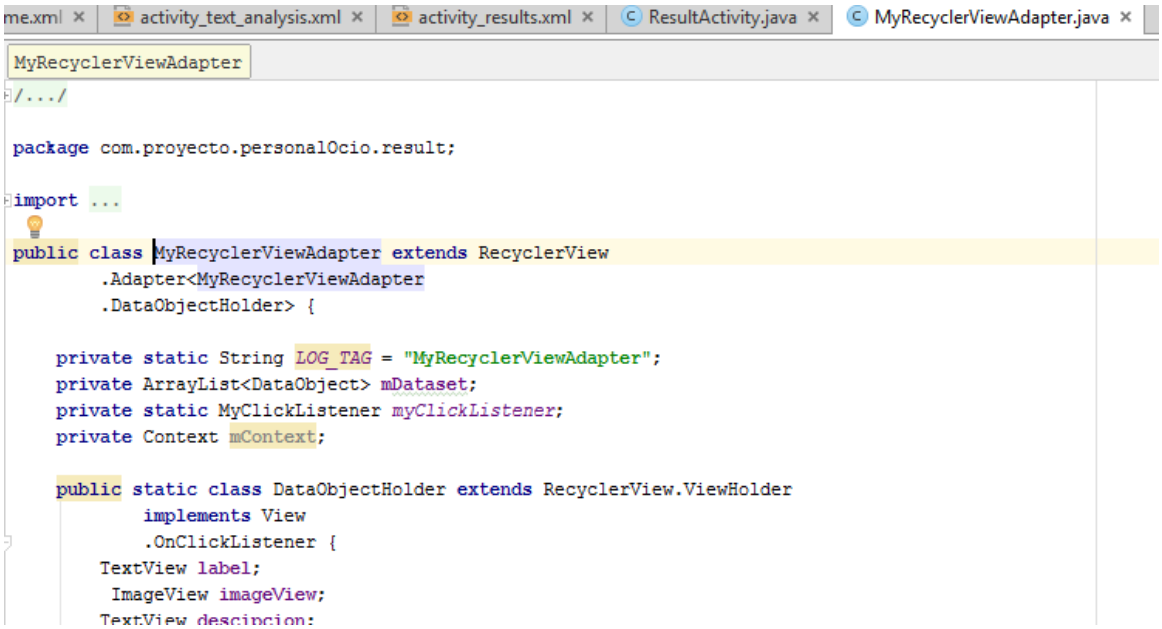


Figura 23: Adaptador del CardView

3.2.9. Librería Firebase

Firebase es una plataforma de desarrollo web y móvil que proporciona gran parte de la infraestructura "de servidor" o de backend que podemos necesitar. Ofrece una serie de servicios clave para las aplicaciones Android (también da soporte a otros sistemas operativos) , como pueden ser:

- Almacenamiento (Storage) [48]
- Base de datos en tiempo real (Realtime database)[49]
- Sistema de autenticación (Authentication)[25]
- Mensajería y Notificaciones (Cloud Messaging)
- Estadísticas de uso (Analytics)
- Reporte de errores (Crash Reporting)º
- Publicidad (AdMob)

Para llevar a cabo la aplicación de recomendación de ocio a través del análisis de sentimientos es necesario de los tres primeros servicios y de los que se detallaran en los siguientes apartados: el de autenticación, base de datos y almacenamiento.

3.2.9.1. Autenticación

La librería Firebase proporciona una gran ayuda en la autenticación de los usuario para aplicaciones móviles. Su incorporación es muy sencilla y permite varios tipos de autenticación posibles. A continuación, se describen los principales métodos de autenticación que soporta esta libería :

- Proovedores: Admite la autenticación a través de Google, Facebook, GitHub y Twitter de una forma facil. Si el usuario dispone de alguna cuenta de usuario en alguno de estos proveedores, puede utilizarla como forma de identificación.
- Correo/Contraseña: Permite autenticar cuentas de usuario a través del correo electronico y contraseña. Ofrece la opcion de creación y administración de los perfiles de usuario
- Numero de telefono: La autenticación se realiza a traves del envío de un mensaje SMS al móvil del usuario.
- Anónima: Permite crear cuentas temporales para el acceso parcial a una aplicación.
- Sistemas de autenticación personalizados

Además de ofrecer una integración sencilla con Android y otros sistemas operativos, Firebase permite facilmente administrar los usuarios de la aplicación a través de su consola. Es necesario dar de alta la aplicación en este servicio y acceder a la consola de administración. En la Figura 24, se muestra la página principal un resumen del analisis realizado en los últimos 30 dias de los usuarios activos, fallos obtenidos junto con lso usuarios afectados

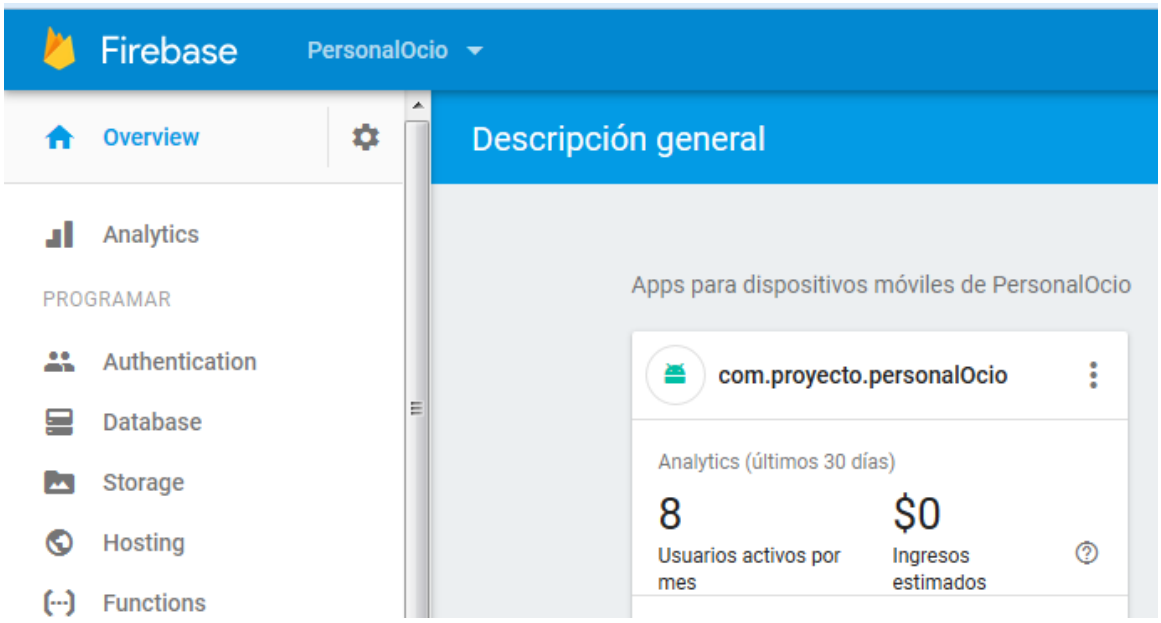


Figura 24: Consola de Administración de Firebase[47]

Además, facilita una administración completa de los usuarios, indicando el primer y último acceso. Controlar lo méétodos de autenticación admitidos en la

aplicación y utilizar plantilla para mandar al usuario si necesita reestablecer la contraseña.

3.2.9.2. Base de datos

Firebase Realtime Database es una base de datos alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. Cuando se compila la app con nuestros SDK de Android todos los clientes comparten una instancia de Realtime Database y reciben actualizaciones de forma automática con los datos más recientes.

En la Figura 25, se muestra como se guarda el objeto con todas las metricas faciales recogidas unidas al análisis de sentimientos dentro de la base de datos. La base de datos de Google, tiene una estructura en forma de arbol (Figura 26), siendo el nodo raiz el usuario. Tras el identificador del usuario se crea un nuevo nivel con "realizado-hoy" y "metrica". A su vez, en otro nivel y debajo de "metrica", se encuentra la fecha del cual se añaden los valores obtenidos del análisis de sentimientos.

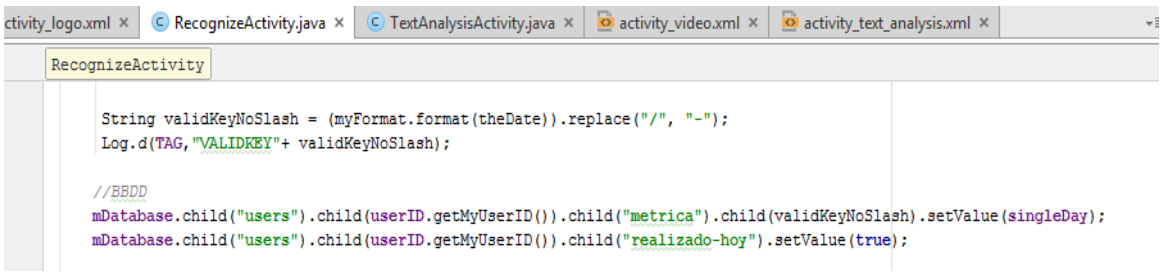


Figura 25: Uso de Firebase Realtime Database

Figura 26: Árbol de usuarios en Firebase Database

3.2.9.3. Almacenamiento

Firebase Storage se creó para programadores de apps que necesitan almacenar y suministrar contenido generado por usuarios, como fotos o videos.

Firebase Storage agrega la seguridad de Google a las cargas y descargas de archivos para las apps de Firebase, sin importar la calidad de la red. Puede utilizarse para almacenar imágenes, audio, video y otros tipos de contenido generado por usuarios. Firebase Storage se basa en la tecnología de Google Cloud Storage, un servicio de almacenamiento de objetos potente, simple y rentable.

En la Figura 27 se puede observar cómo se utiliza esta librería. Es necesario crear una instancia con la referencia del almacenamiento y una vez obtenida se añade también en estructura en árbol el fichero que se desea guardar.

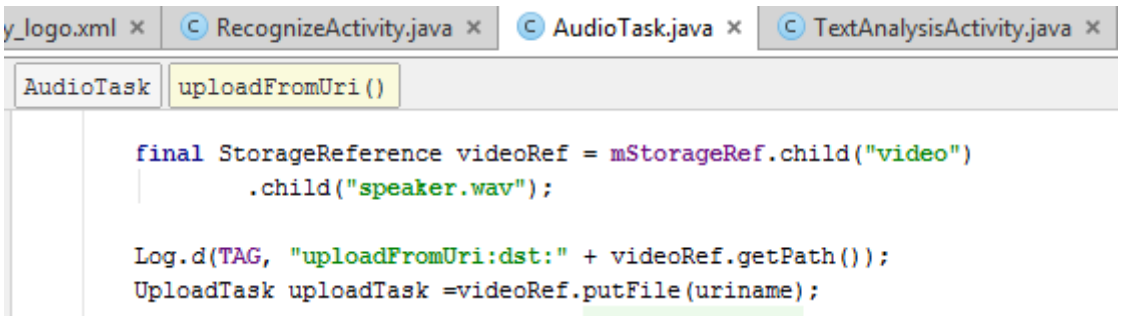


Figura 27: Uso de Firebase Storage

En la Figura 28, se muestra cómo se encuentra almacenado el archivo y toda la información propia del mismo (ubicación, tamaño, fecha de creación). Estas propiedades aparecen en la siguiente figura en el lateral derecho.

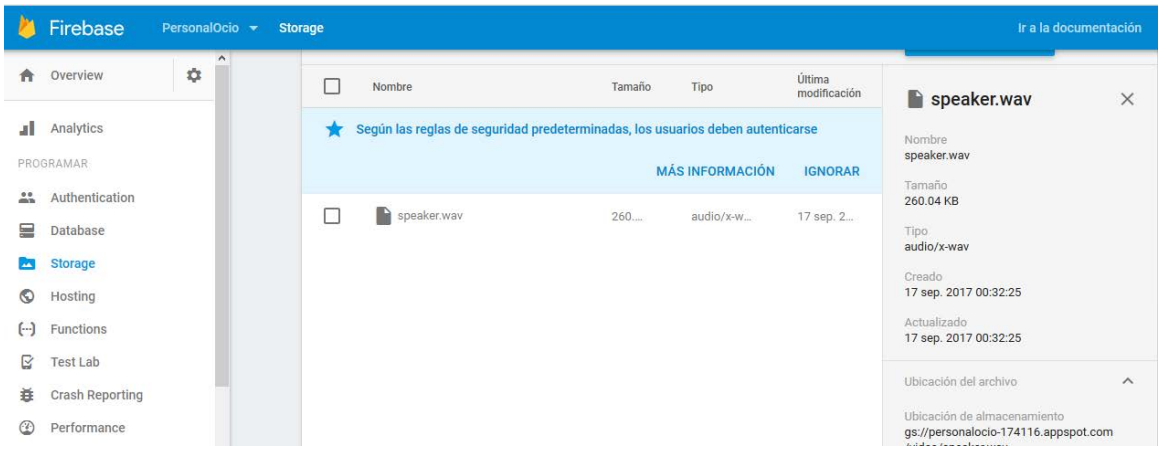


Figura 28: Audio almacenado en Firebase Storage

3.2.10. Librería de análisis del texto

Meaning Cloud y Google ofrecen esta funcionalidad para las aplicaciones Andoid. A continuación, Se describen las librerias utilizadas para analizar el texto

3.2.10.1. Meaning Cloud Classification Text (Clasificación de texto)

Meaning Cloud es un servicio de analítica y minería de datos. A través de su API permite a los usuarios integrar análisis de texto y procesamiento semántico en cualquier aplicación. Entre la funcionalidad que ofrece esta la extracción de temas que identifica aparición de entidades en el texto. Además, puede clasificar un texto en una o varias categorías y realizar un análisis de sentimiento analizando la polaridad (positiva, negativa o neutra) de un documento.

Las solicitudes se realizan mediante las presentaciones de datos POST al punto de entrada de la API. La Tabla 3 muestra los principales parámetros soportados en la solicitud y los utilizados en la aplicación.

Campo	Descripcion	Utilizado
key	Este campo indica la clave de acceso	Si
of	Este campo indica el formato salida	Si, con el valor por defecto json
txt	Este campo indica el texto a analizar	Si, con el texto codificado UTF-8
model	Este campo indica el modelo de clasificación sentimiento seleccionado	Si, con el valor "MOcio". Incluso se ha probado con IAB_es.

Tabla 3: Classification Text- Solicitud[28]

Como modelo de clasificación, la herramienta permite crear modelos propios. Se ha creado para la aplicación el modelo MOcio con las palabras relacionadas con "futbol", "restaurantes" y "cine"

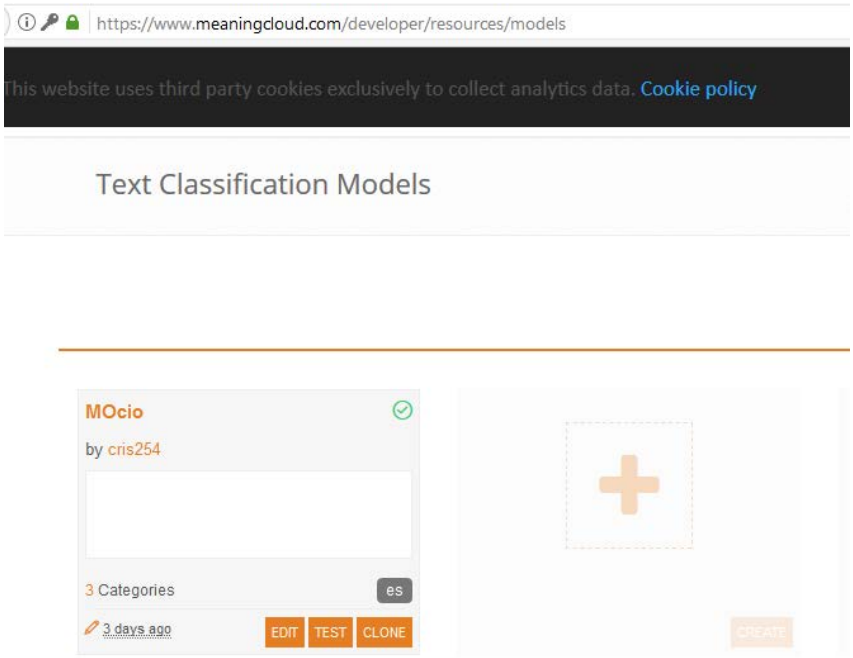


Figura 29: Modelo de datos MOcio

En la Tabla 4, mostrada a continación, se encuentra la respuesta de tipo JSON tras la llamada al servidor de clasificación de texto. Se incluyen los principales campos y los utilizados por la aplicación.

Campo	Descripcion	Utilizado
status	Este campo contiene el resultado del proceso de analizar el sentimiento de texto. Se trata de un objeto que contiene los campos code (identificador) , msg (descripción del estado), credits (creditos consumidos en la solicitud) y remaining_credits (creditos disponibles)	Si
category_list	Este campo contiene una lista de categorias. A su vez cada categoria contiene el campo code (identificador de la categoria), label (descripcion) , abs_relevance (valor de relevancia de la categoria) , relevance (valor % relativo a la categoria y term_list con la lista de terminos	Si

Tabla 4: Classification – Respuesta JSON [29]

3.2.10.2. Meaning Cloud Sentiment

La API de Análisis de Sentimiento realiza un análisis de sentimiento multilingüe detallado a partir de información proveniente de diversas fuentes.

El texto proporcionado se analiza para determinar si expresa un sentimiento positivo, neutro o negativo (o si es imposible detectar ningún sentimiento). Para ello, se identifica la polaridad local de las diferentes frases en el texto y se evalúa la relación entre ellas, lo que resulta en un valor de polaridad global para el texto en su conjunto.

Además, Sentiment Analysis detecta si el texto procesado es subjetivo u objetivo y si contiene marcas de ironía [beta] a nivel global, dando al usuario información adicional sobre la fiabilidad de la polaridad obtenida a partir del análisis de sentimiento.

El uso de esta API se lleva a cabo a través de una petición POST. La Tabla 5 contiene los principales parámetros admitidos y aquellos tratados por la aplicación.

Campo	Descripcion	Utilizado
key	Este campo indica la clave de acceso	Si
of	Este campo indica el formato salida	Si, con el valor por defecto json
lang	Este campo indica el idioma en el que se analiza el texto	Si, con el valor ‘es’ que indica el idioma español
txt	Este campo indica el texto a analizar	Si, con el texto codificado UTF-8
model	Este campo indica el modelo de sentimiento seleccionado	Si, con el valor “general”
rt	Esta campo indica la fiabilidad del texto a analizar en cuanto a ortografia, tipologia...	Si, con el valor ‘y’ que indica habilitado para todo los recurso

Tabla 5: Sentiment -Solicitud[27]

La salida contiene información sobre el estado de la solicitud, las diferentes polaridades identificadas en el texto, su subjetividad y si contiene marcas de ironía. La información proporcionada es la misma para los diferentes formatos de salida.

La Tabla 6 muestra los principales campos JSON incluidos en el objeto de respuesta y aquellos tratados por la aplicación.

Campo	Descripcion	Utilizado
status	Este campo contiene el resultado del proceso de analizar el sentimiento de texto. Se trata de un objeto que contiene los campos code (identificador) , msg (descripción del estado), credits (creditos consumidos en la solicitud) y remaining_credits (creditos disponibles)	Si
model	Modelo utilizado en el análisis	Si
score_tag	Polaridad encontrada. • P+ =positivo fuerte • P=positivo • NEU = neutro • N = negativo • N+ = negativo fuerte • Ninguno = sin sentimiento	Si
agreement	Este campo indica si los elementos tienen la misma polaridad con el valor "ACUERDO". "DESACUERDO" si indican distinta polaridad	No
subjetivity	Este campo indica si es objetivo o subjetivo el contenido del texto	No
confidence	Esta campo indica el valor de confianza asociada al texto.	No
irony	Este campo indica si el texto contiene ironia o no con los valores NONIRONIC o IRONIC	Si

Tabla 6: Sentiment – Respuesta JSON[26]

3.2.10.3. Google Natural Language

La API Natural Language de Cloud descubre la estructura y el significado del texto mediante modelos de aprendizaje automático en una API REST fácil de usar. Se puede utilizar para extraer información sobre personas, lugares, eventos y muchos elementos más que se mencionen en documentos de texto, artículos de noticias o entradas de blogs. Es posible analizar el texto que se suba en la solicitud o integrar la función en tu almacenamiento de documentos de Google Cloud Storage.

En la Figura 30 y 31, se muestra cómo se realiza el uso de esta librería. En la primera de ellas, se crea una instancia del servicio con la clave de Google. Posteriormente se configura la petición a realizar con el documento a analizar y las propiedades del análisis, en este caso detección de entidades.

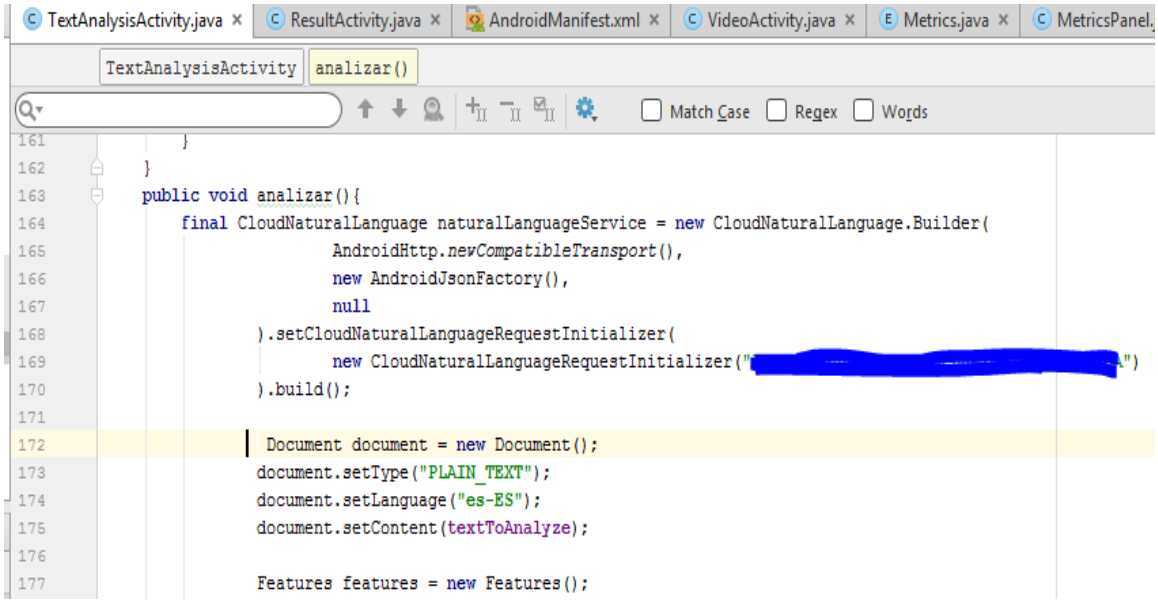


Figura 30: Llamada API Cloud Natural Language

En la Figura 28, se lanza la petición con el método *execute()*, se realiza el tratamiento de la respuesta, obteniendo el listado de entidades que se recorre para obtener el campo nombre.

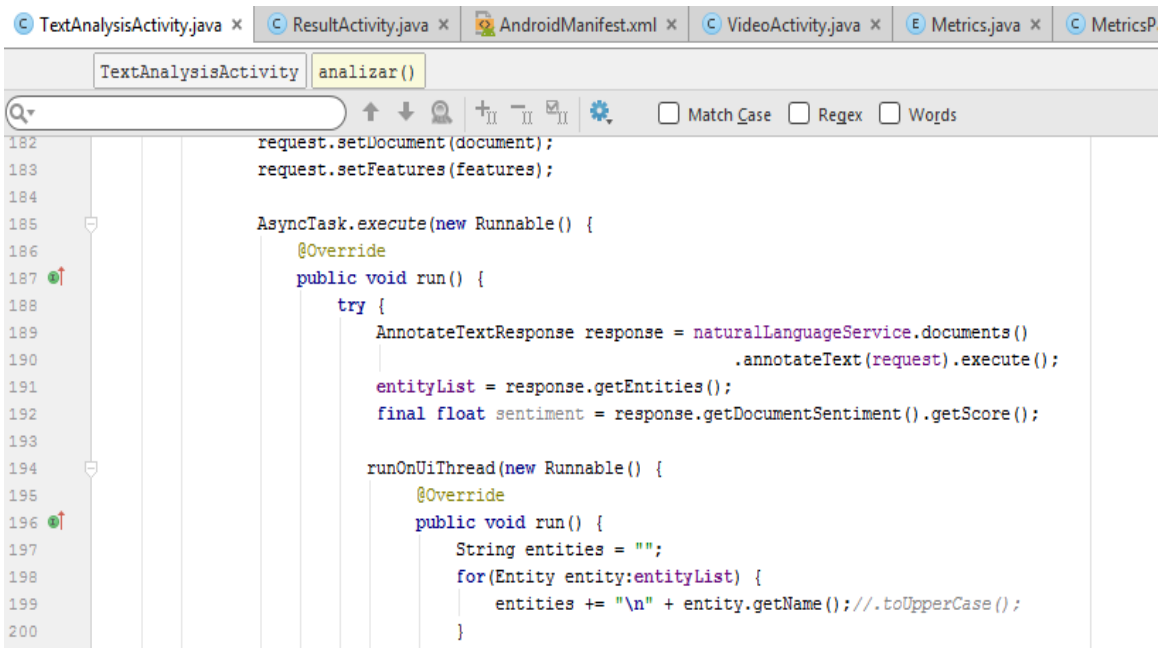


Figura 31: Tratamiento de respuesta Api Cloud Natural Language

3.2.11. Librería de reconocimiento de voz

Para la integración del reconocimiento de voz existen varias alternativas que se muestra a continuación

3.2.11.1. Recognizer Intent

Android no puede reconocer el habla, de manera que un dispositivo Android típico tampoco puede reconocer el habla. La única manera de que lo haga y la más fácil es pedir a otra aplicación que realice el reconocimiento. Pedir a otra aplicación que haga algo en Android se llama uso de intenciones.

Nuestro dispositivo de destino debe tener al menos una aplicación que pueda procesar la Intención para el reconocimiento del habla, la cual es llamada por la acción RecognizerIntent.ACTION_RECOGNIZE_SPEECH.

Una de esas aplicaciones es Google Voice Search. Es uno de los mejores reconocedores disponibles para Android y es compatible con varios idiomas. Este servicio requiere una conexión con Internet debido a que el reconocimiento de voz se lleva a cabo en los servidores de Google. Esta aplicación tiene una Actividad muy simple que informa a los usuarios que pueden hablar. El momento en que el usuario deja de hablar, se cierra el diálogo y nuestra aplicación (intent caller) recibe una gama de cadenas con el reconocimiento del habla.

La aplicación necesita hacer lo siguiente:

- Recibir una solicitud de reconocimiento de voz
- Consultar la disponibilidad de la aplicación para el reconocimiento del habla
- Si el reconocimiento del habla está disponible, entonces debe llamar a la intención para éste y recibir los resultados
- Si el reconocimiento del habla no está disponible, entonces debe mostrar el diálogo para instalar Google Voice Search y, si acepta, redirigir al usuario a Google Play

En primer lugar, es necesario dar a la aplicación los permisos necesarios para que soporte el reconocimiento de voz. Dado que para que funcione el reconocimiento de voz es necesario tener acceso a Internet, se dará dicho permiso en el archivo AndroidManifest.xml a través de la siguiente línea de código.

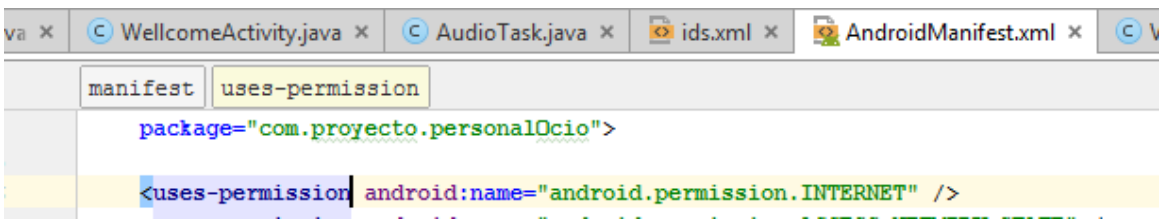


Figura 32: Permisos para el reconocimiento de voz

En primer lugar, se crea un objeto de la clase Intent (“API”, 2014) cuya acción es del tipo RecognizerIntent.ACTION_RECOGNIZE_SPEECH. Dicha actividad es la que devuelve los resultados del reconocimiento. Posteriormente, se configuran los parámetros de la actividad de reconocimiento de voz a través de la llamada del objeto de la clase Intent a putExtra(String name, String value) que especifica un par <clave,valor>. Estos parámetros indican el modelo de lenguaje utilizado a través de la clave EXTRA_LANGUAJE_MODEL.

En el ejemplo de la Figura 33, se muestra el modelo de lenguaje FREE_FORM. Además, se especifica el lenguaje por defecto y el mensaje que se mostrará al usuario cuando se solicite que hable a través de la clave EXTRA_PROMPT. Finalmente, se llama al método *startActivityForResult(Intent intent, int requestCode)*, que se encarga de invocar a la actividad de reconocimiento de manera que devuelva los resultados del reconocimiento a través del método *onActivityResult(int requestCode, int resultCode, Intent data)*. La constante SPEECH_RECOGNITION _CODE se utiliza para identificar a la actividad que invoca a dicho intent y se declara al principio de la actividad.

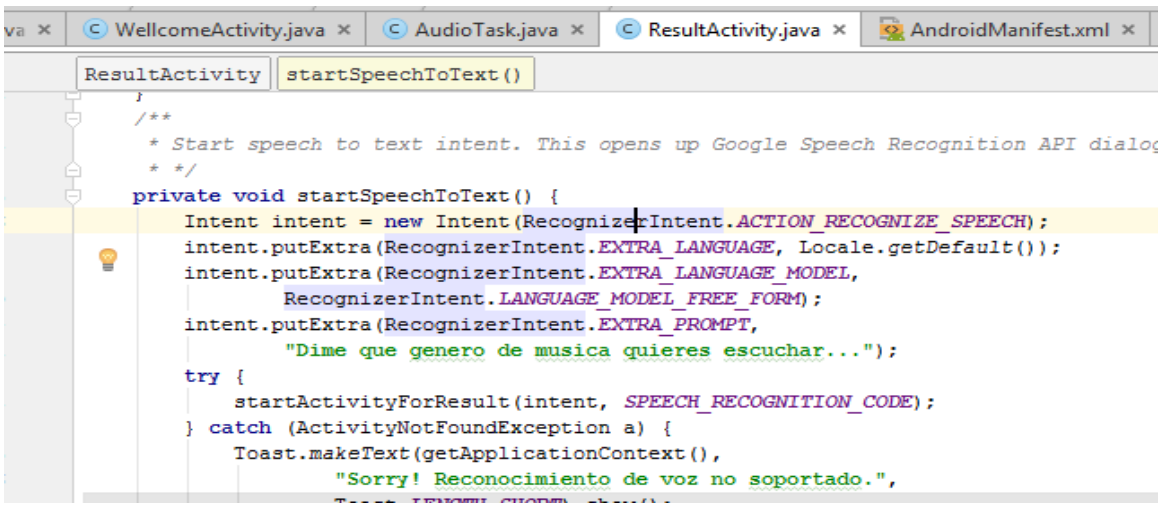


Figura 33: Creación del RecognizerIntent

En la Figura 34, se procesan los resultados en el método onActivityResult

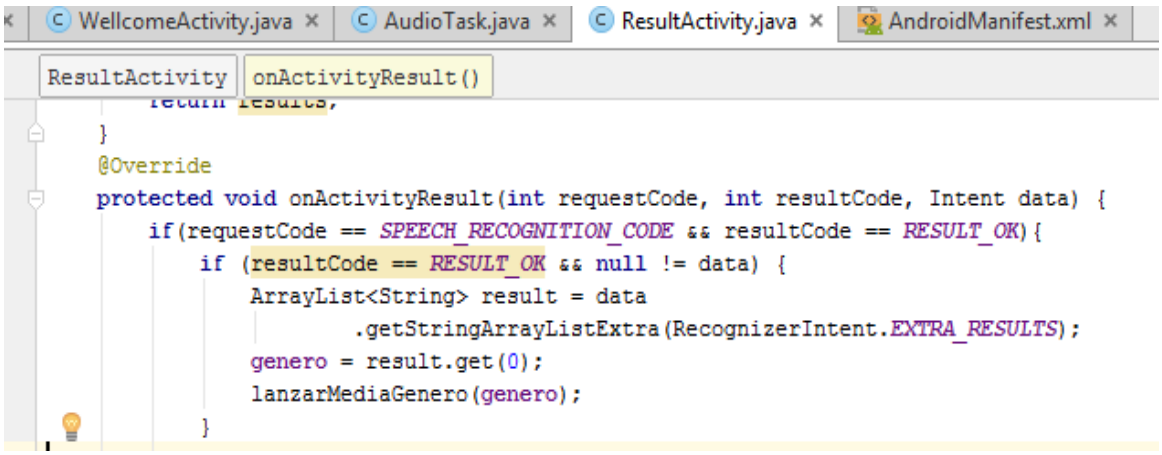


Figura 34: Resultado del RecognizerIntent

3.2.11.2. Google Speech

La API Speech de Google Cloud permite convertir audio en texto aplicando modelos de redes neuronales en una API fácil de usar. La API reconoce más de 80 idiomas y variantes. Puedes transcribir el texto que los usuarios dictan al micrófono de una aplicación, habilitar el control por voz o transcribir archivos de audio, entre muchas otras funciones. Es posible reconocer el audio subido en la solicitud e integrarlo en tu almacenamiento de audio de Google Cloud Storage. Y todo, con la misma tecnología que empleamos en Google para nuestros productos.

En el ejemplo de la Figura 35, se muestra cómo se inicializa el servidor de Google Speech al que se le pasa la clave de Google para su acceso. Después, se configura el reconocimiento de voz con el tipo de codificado, la frecuencia en 16000 Hz y el idioma en español. El siguiente paso es hacer referencia al audio, en este caso es la uri donde se almacena el audio del usuario.

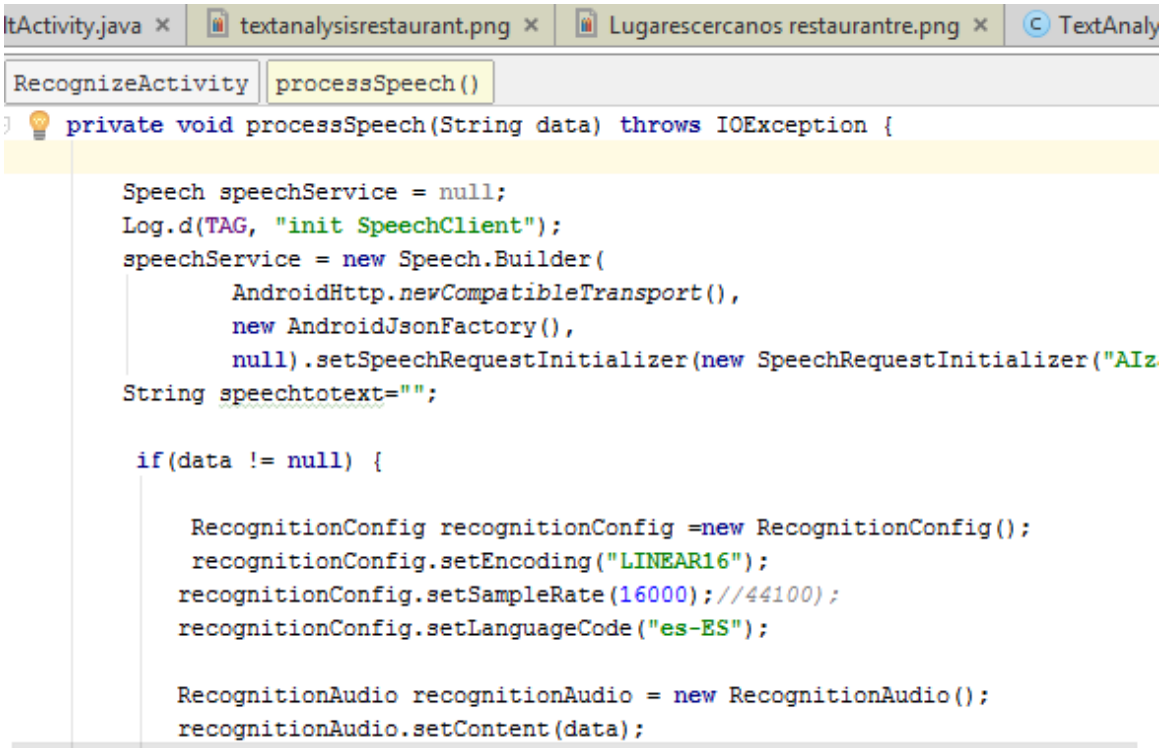


Figura 35: Uso de Google Speech

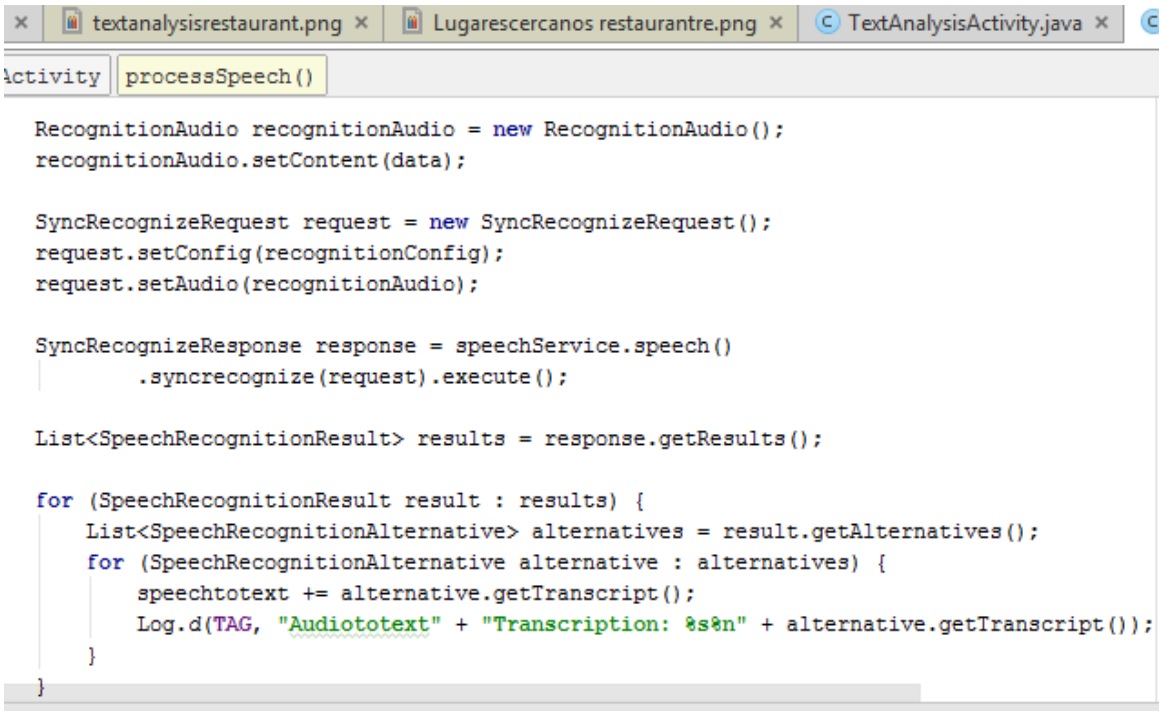


Figura 36: Recuperar la respuesta de Google Speech

En la Figura 36, se observa cómo se construye la petición SyncRecognizeRequest con la configuración, RecognitionConfig y el audio, RecognitionAudio. Una vez lanzada la petición con el método *execute()*, se recoge la respuesta con *getResults()*. Y se va recorriendo la lista de alternativas para sacar la transcripción de cada una de ellas.

3.2.12. Librería de reconocimiento facial

A continuación, se detallan las librerías encontradas para el reconocimiento facial. Se han llegado a probar varias opciones, que se muestran en este primer grupo como librerías descartas y posteriormente se detalla la librería utilizada

3.2.12.1. Librerías descartadas

Las siguientes librerías hacen manejo de imágenes de reconocimiento facial junto con la funcionalidad adicional de aportar información básica de los sentimientos obtenidos del rostro:

- Vision [21]

Google Cloud Visión API permite entender el contenido de la imagen. Esta librería permite clasificar por categorías las imágenes, detectar objetos/caras, leer palabras impresas en la imagen.

La opción más interesante para este proyecto es la relacionada con el análisis de sentimientos. Por lo que permite analizar las emociones faciales de una fotografía.

Sin embargo, no permite el procesamiento en tiempo real de un video.

- Face API [19]

La librería Face es una de las funcionalidades de Mobile Vision API. Permite centrarse en las características faciales. Por ahora permite dos tipos de clasificaciones: uno de ellos es detectar si los ojos están abiertos o cerrados y el otro es comprobar si está sonriendo o no.

Face API detecta las caras inclinadas con distinto ángulo a través del sistema de coordenadas XYZ y la medición de los ángulos de Euler Y y Z (la librería no mide el ángulo de Euler X). Según estos grados de referencia y la detección de los ojos como punto de referencia del reconocimiento facial, la librería determina qué marcas puede detectar. Las marcas pueden ser el lado derecho o izquierdo de los ojos, nariz, boca, orejas y barbilla o incluso ambos si la cara se encuentra de frente

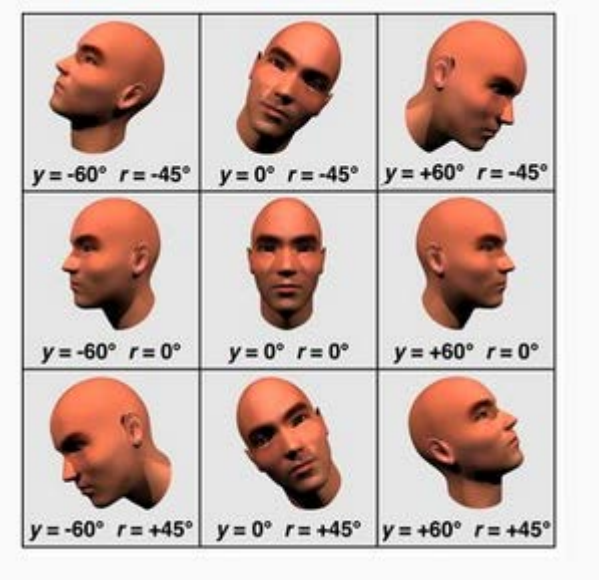


Figura 37: Posición de la cara y los ángulos de Euler[19]

Cada punto de referencia detectado incluye su posición asociada en la imagen. Como se ha comentado anteriormente, las dos posibles clasificaciones que realiza esta librería se calculan indicando la probabilidad de que la característica facial este presente. Por lo tanto, si el valor es un 0,7 o más para la característica de sonrisa indica que es muy probable que la persona este sonriendo.

Ambas clasificaciones solo funcionan en caras frontales donde el ángulo de Euler Y sea pequeño (como mucho +/+ 18 grados)

Esta librería también requiere de imágenes para su procesado. La aplicación descrita en este documento requiere que procese un video en tiempo real. Por lo tanto, esta librería se descarta.

- Emotion 1.0[22]

Es una librería que permite la utilización de video. Este video se puede subir al servidor o en tiempo real. Esta es una de las opciones más factibles y la aplicación descrita en el presente documento comenzó incorporando esta funcionalidad, pero

finalmente se tomó la decisión de descartarla por ser de pago, de caducar la versión de prueba y estar próxima a ser discontinuada.

- Face 1.0 [23] Microsoft.

El API Face 1.0 detecta una o varias caras humanas en una imagen y obtenga rectángulos faciales donde se encuentran las caras en la imagen, junto con atributos faciales que contienen predicciones de características faciales basadas en aprendizaje automático. Las características de atributos faciales disponibles son: edad, emoción, sexo, postura, sonrisa y vello facial, junto con 27 puntos de referencia para cada cara de la imagen.

Con esta librería se puede obtener la edad de la persona, el género, la intensidad de la sonrisa, en la barba, pose de la cabeza, si lleva no gafas, tamaño de la nariz, si lleva maquillaje entre otras características faciales.

Una vez evaluada, se descartó por finalizar el periodo de prueba y ser una aplicación de pago. Además, realiza la funcionalidad con imagen no a través de un video.

3.2.12.2. Librería utilizada para la aplicación: Affectiva

La cara proporciona un lienzo de emoción. Los seres humanos están innatamente programados para expresar y comunicar emociones a través de expresiones faciales. La librería Affectiva [24] mide y reporta científicamente las emociones y las expresiones faciales utilizando técnicas de visión artificial y aprendizaje de máquinas. Tiene la posibilidad de medir 7 métricas de emoción: ira, desprecio, disgusto, miedo, alegría, tristeza y sorpresa. Además, ofrece 20 métricas de expresión facial.

La siguiente tabla muestra la relación entre las expresiones faciales y los predictores de emociones.

Emoción	**Aumentar la probabilidad**	**Disminuir Probabilidad**
Alegría	Sonreír	Levantamiento de cejas Surco de la frente
Enfado	Surco de la frente Apertura de Ojo Mentón levantado Boca abierta	Levantamiento interno de la frente Levantamiento de cejas

	Labios apretados	Sonreír
Asco	Arrugas de la nariz Levantamiento de Labios Superior	Labios apretados Sonreír
Sorpresa	Levantamiento interno de la frente Levantamiento de cejas Ojo Ampliar Gota de mandíbula	Surco de la frente
Miedo	Levantamiento interno de la frente Surco de la frente Apertura de Ojo Estiramiento labial	Levantamiento de cejas Labios hacia abajo Gota de mandíbula Sonreír
Tristeza	Levantamiento interno de la frente Surco de la frente Labios hacia abajo	Levantamiento de cejas Apertura de Ojo Labios apretados Boca abierta Sonreír
Desprecio	Surco de la frente Sonrisa afectada	Sonreír

Tabla 7: Predictores de emociones/Expresiones faciales

Las puntuaciones de las métricas Emoción y Expresión indican cuándo los usuarios muestran una emoción o expresión específica (por ejemplo, una sonrisa) junto con el grado de confianza. Las métricas pueden considerarse como detectores: a medida que la emoción o expresión facial se produce e intensifica, la puntuación sube de 0 (sin expresión) a 100 (expresión totalmente presente).

También proporcionan las siguientes métricas sobre el aspecto físico:

- Años: El clasificador de edad intenta estimar el rango de edad. Rangos soportados: Menores de 18, de 18 a 24, 25 a 34, 35 a 44, 45 a 54, 55 a 64 y 65 Plus.

- Etnicidad: El clasificador étnico intenta identificar la etnia de la persona. Clases de apoyo: caucásico, negro africano, sur asiático, asiático oriental e hispano.

En el nivel actual de exactitud, los clasificadores étnicos y de edad son más útiles como una medida cuantitativa de la demografía que para identificar correctamente la edad y la etnia sobre una base individual. Siempre estamos buscando diversificar las fuentes de datos incluidas en el entrenamiento de esas métricas para mejorar sus niveles de precisión.

- Género: El clasificador de género intenta identificar la percepción humana de la expresión de género.

En el caso de video o frames en vivo, los clasificadores de género, edad y etnia rastrean una cara para una ventana de tiempo para crear confianza en su decisión. Si el clasificador no puede llegar a una decisión, el valor del clasificador se informa como "Desconocido".

- Gafas: Un nivel de confianza de si el sujeto en la imagen está usando anteojos o gafas de sol.

Esta librería permite analizar imagen, video y secuencia de fotograma. Esta última opción es la más ágil y apropiada para la aplicación. Utiliza la clase FrameDetector. El FrameDetector rastrea las expresiones en una secuencia de tramas en tiempo real. Espera que cada trama tenga una marca de tiempo que indique la hora en que se capturó el fotograma. Las marcas de tiempo llegan en un orden creciente. El FrameDetector detectará un rostro en un marco y le entregará información sobre él, incluyendo las expresiones faciales.

El constructor de FrameDetector espera tres parámetros {context, maxNumFaces (por defecto 1) y faceDetectorMode}. El último parámetro indica el tamaño de la cara respecto al frame, es decir, si ocupan la mayor parte del frame o no. Y existe el FrameDetector en background llamado AsyncFrameDetector, en la Figura 29 se puede ver la utilización de este detector.

Los detectores utilizan funciones de devolución de llamada definidas en las clases de interfaz para comunicar eventos y resultados. Los oyentes de eventos deben ser inicializados antes de que se inicie el detector: FaceListener es una interfaz de devolución de llamada del cliente que envía notificación cuando el detector ha iniciado o detenido el rastreo de una cara. Es necesario llamar a setFaceListener para configurar FaceListener:

Después de configurar un detector utilizando los métodos anteriores, la inicialización del detector se puede activar llamando al método de inicio *"start()",* se puede observar la llamada en la Figura 38.

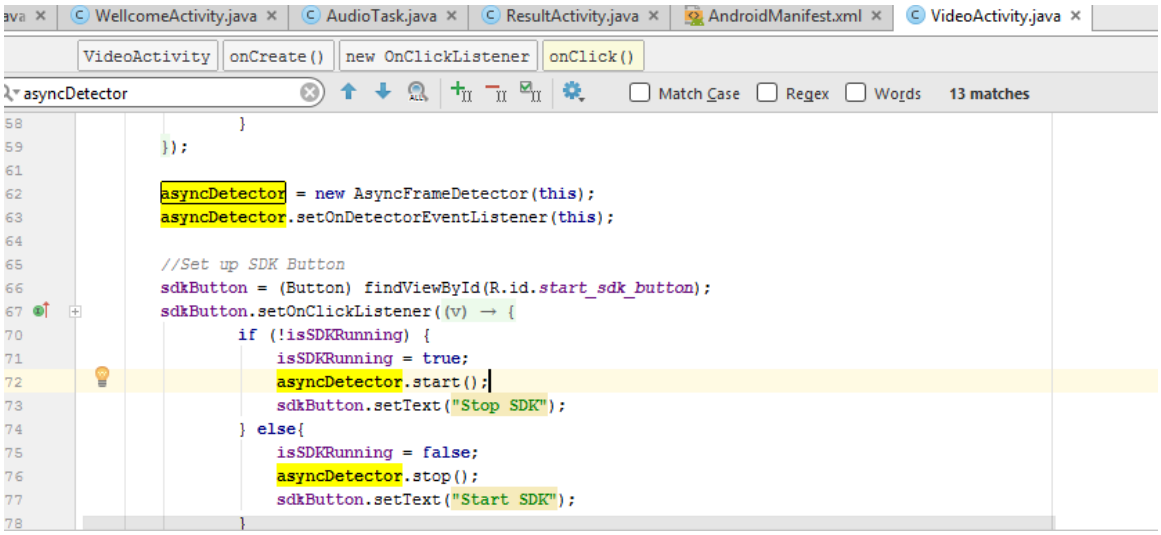


Figura 38: Uso de la detección asíncrona

Necesita de los siguientes permisos incluidos en AndroidManifest.xml

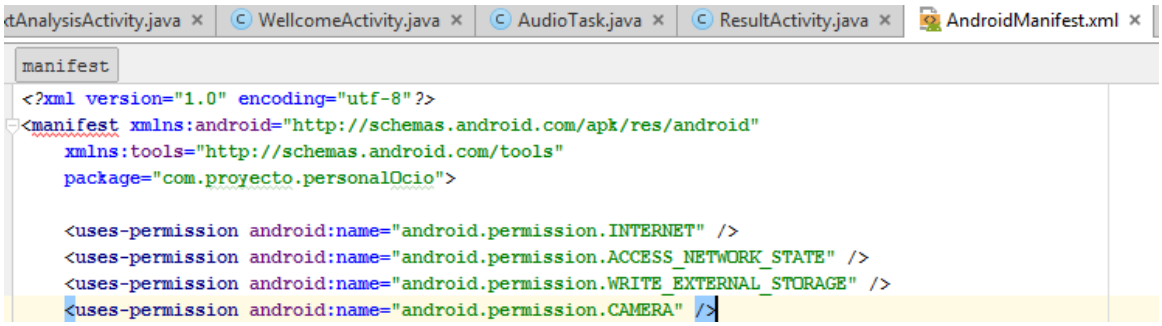


Figura 39: Permisos necesarios para el reconocimiento facial

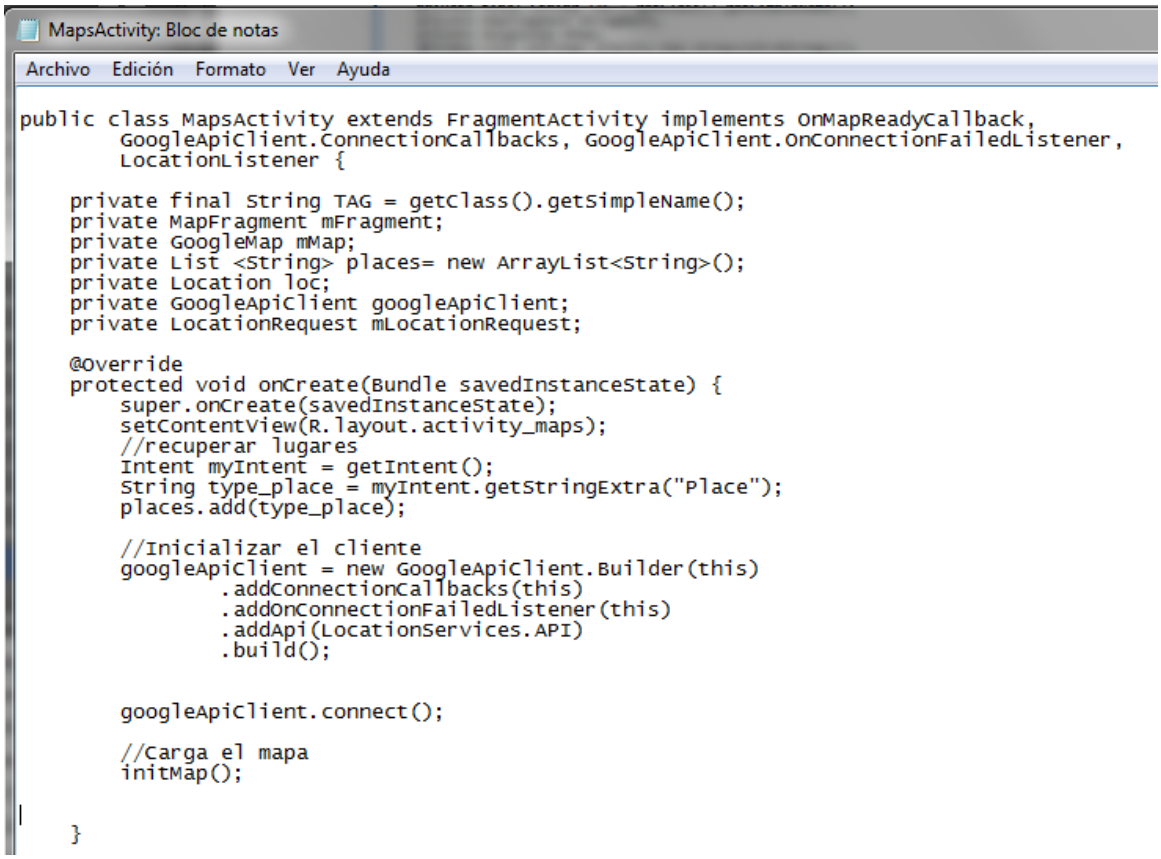
3.2.13. Mapas

Place Picker [32] Google, "es un widget simple, integrado y flexible de la IU, que forma parte de Google Places API for Android. Proporciona un diálogo de IU que muestran un mapa interactivo y una lista de sitios cercanos, incluidos los sitios que corresponden a direcciones geográficas y negocios locales. Los usuarios pueden elegir un sitio y, a continuación, una aplicación puede recuperar los detalles del sitio seleccionado. Sin embargo, la aplicación Personal Ocio necesita un filtrado por las preferencias del usuario y la opción de incorporar filtros no es posible hoy en día. Por lo tanto, esta opción ha sido descartada.

Google Place Api [34] permite hacer búsquedas por tipo de sitio que es lo que busca la aplicación una vez analizado el texto. Sin embargo, al crear un mapa y añadir los lugares la interfaz gráfica es muy simple.

En la Figura 40 y 41, se puede apreciar cómo se inicializa un mapa con la librería de Google. En primer lugar, se crea un cliente de la librería con el servicio de localización activo. Y posteriormente se inicializa el mapa a través del

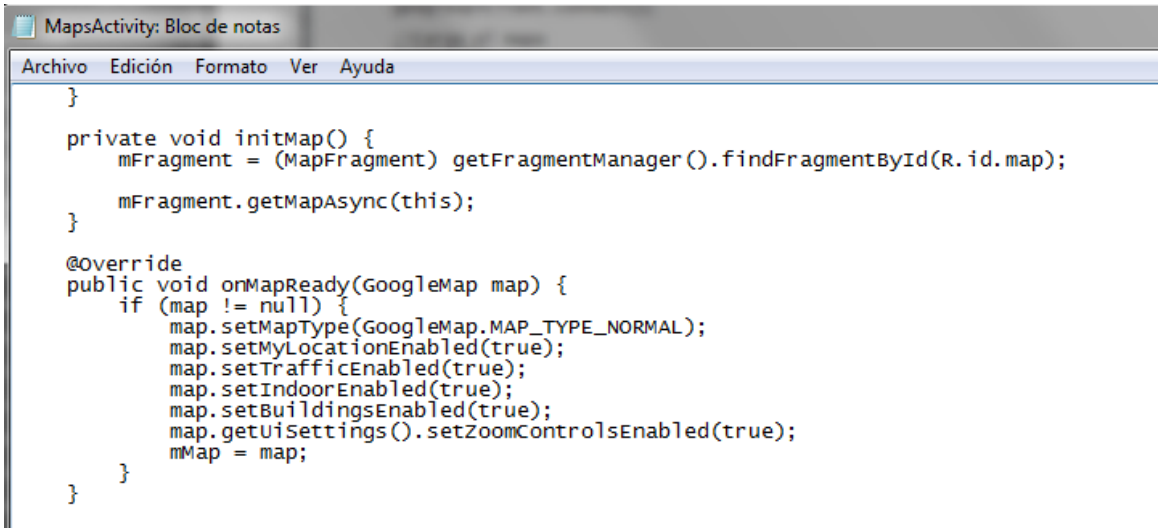
getMapAsync propio de la clase MapFragment. Una vez recibida la respuesta se configura el mapa. En la aplicación desarrollada se activaron todas las opciones de la localización, tráfico, edificios e interiores para realizar las pruebas iniciales y aportar al usuario la mayor cantidad de información posible. La idea inicial era acotar esta información tras realizar alguna prueba. Sin embargo, esto no llegó a suceder por la decisión de utilizar la aplicación Google Maps instalada ya en todos los móviles.


MapsActivity: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
        LocationListener {

    private final String TAG = getClass().getSimpleName();
    private MapFragment mFragment;
    private GoogleMap mMap;
    private List <String> places= new ArrayList<String>();
    private Location loc;
    private GoogleApiClient googleApiClient;
    private LocationRequest mLocationRequest;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        //recuperar lugares
        Intent myIntent = getIntent();
        String type_place = myIntent.getStringExtra("Place");
        places.add(type_place);

        //Inicializar el cliente
        googleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();


        googleApiClient.connect();

        //Carga el mapa
        initMap();


    }
```

Figura 40: Creación del mapa incrustado en la aplicación

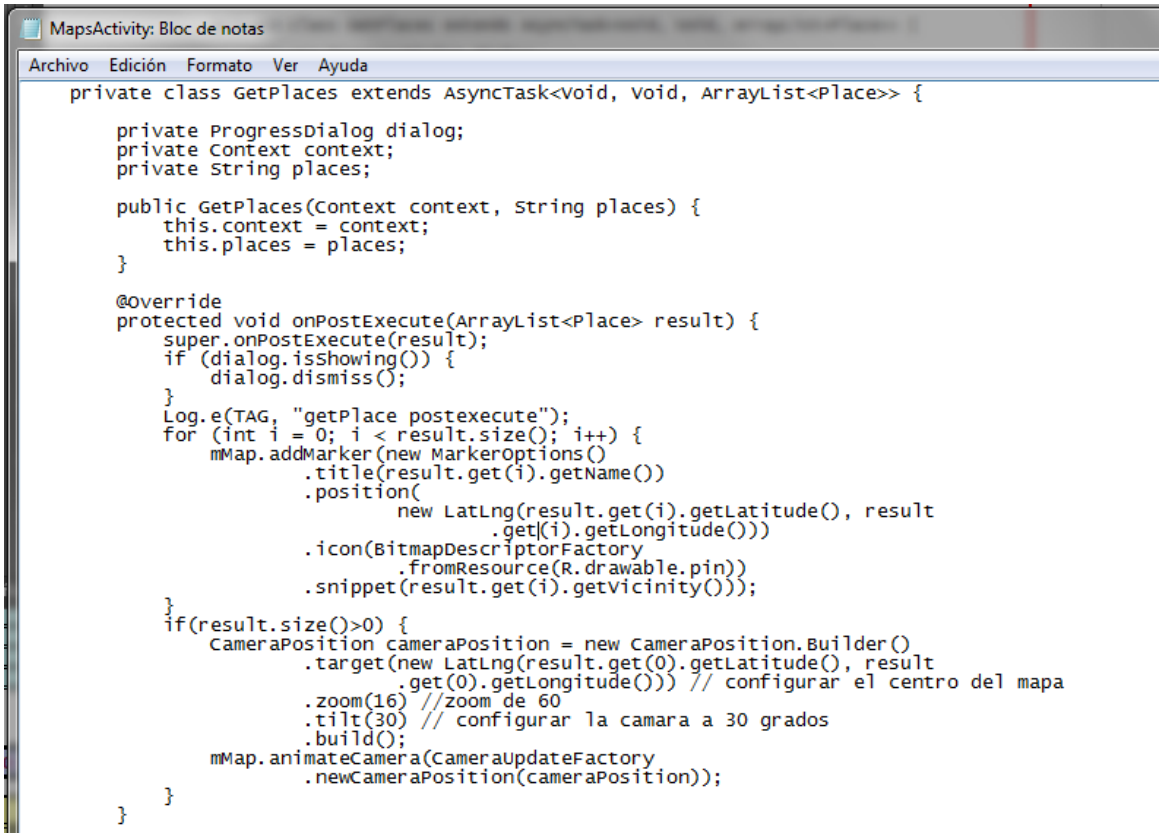

MapsActivity: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
    }

    private void initMap() {
        mFragment = (MapFragment) getFragmentManager().findFragmentById(R.id.map);

        mFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap map) {
        if (map != null) {
            map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
            map.setMyLocationEnabled(true);
            map.setTrafficEnabled(true);
            map.setIndoorEnabled(true);
            map.setBuildingsEnabled(true);
            map.getUiSettings().setZoomControlsEnabled(true);
            mMap = map;
        }
    }
```

Figura 41: Inicialización del mapa incrustado en la aplicación

Una vez creado el mapa , se utilizan los marcadores para añadir al mapa los lugares cercanos acordes a la búsqueda a través de la función *addMarker* como indica la Figura 42 . Este método requiere el nombre del lugar, la posición latitud/longitud y el icono con el que se quiere marcar. Tras añadir los marcadores se suele configurar la posición de la cámara y añadir la animación típica de Google que realiza la transición de una localización a otro. Para configurar la posición de la cámara es necesario indicar el centro del mapa y la forma de visualización del mismo en cuanto al zoom y rotación de la imagen en grados.


MapsActivity: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
    private class GetPlaces extends AsyncTask<Void, Void, ArrayList<Place>> {

        private ProgressDialog dialog;
        private Context context;
        private String places;

        public GetPlaces(Context context, String places) {
            this.context = context;
            this.places = places;
        }

        @Override
        protected void onPostExecute(ArrayList<Place> result) {
            super.onPostExecute(result);
            if (dialog.isShowing()) {
                dialog.dismiss();
            }
            Log.e(TAG, "getPlace postexecute");
            for (int i = 0; i < result.size(); i++) {
                mMap.addMarker(new MarkerOptions()
                        .title(result.get(i).getName())
                        .position(
                                new LatLng(result.get(i).getLatitude(), result
                                        .get(i).getLongitude()))
                        .icon(BitmapDescriptorFactory
                                .fromResource(R.drawable.pin))
                        .snippet(result.get(i).getVicinity()));
            }
            if(result.size()>0) {
                CameraPosition cameraPosition = new CameraPosition.Builder()
                        .target(new LatLng(result.get(0).getLatitude(), result
                                .get(0).getLongitude())) // configurar el centro del mapa
                        .zoom(16) //zoom de 60
                        .tilt(30) // configurar la camara a 30 grados
                        .build();
                mMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(cameraPosition));
            }
        }
```

Figura 42: Uso del *addMarker* para añadir un lugar

Otro punto importante relacionado con los mapas en la localización del usuario. Para ello se utiliza el cliente de Google previamente creado en la Figura 43 y las clases *LocationRequest* y *LocationServices*. Estas clases actualizar la localización a través del método *requestLocationUpdates* y *getLastLocation* [30]. En la siguiente Figura se puede observar su implementación.

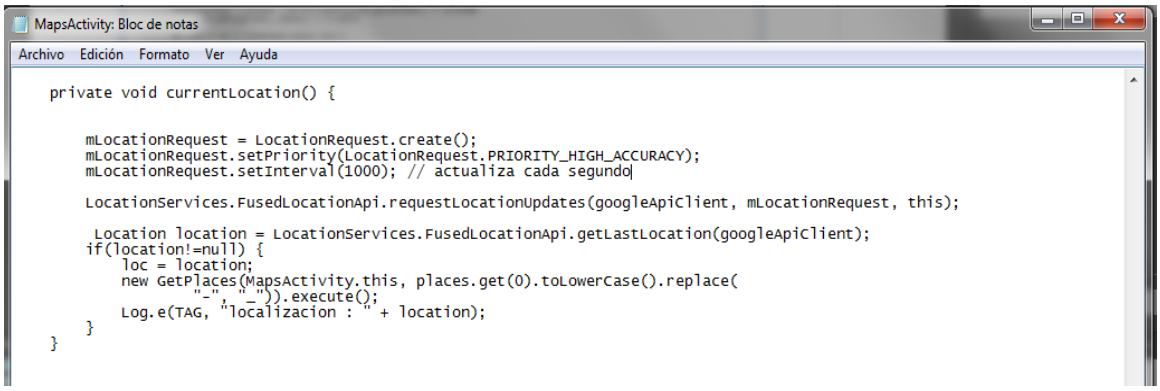


Figura 43: Uso de la geolocalización

Una vez explicado la forma de incluir un mapa en la aplicación, se describe la segunda opción y más visual. Se trata de utilizar los intent [31]Google, para iniciar la actividad del mapa como se ha mencionado brevemente al principio de la sección. Permite lanzar la nueva actividad desde la aplicación desarrollada indicando desde esta última, una búsqueda concreta. La forma de indicar la búsqueda de un lugar por un tipo concreto es incluir en la llamada al Intent el parámetro tipo Uri con la query deseada. El formato de esta query es:

geo:latitud?longitud?q=sitio&label=nombre

- Latitud, longitud: Estos campos se utilizan para indicar la búsqueda de lugares cercanos a esta posición. En la aplicación desarrolla se usan con valor 0 ,0 para indicar la localización actual.
- Sitio: Este campo se utiliza para indicar el tipo de lugar para realizar la búsqueda. La aplicación hace uso de este campo incluyendo el valor obtenido tras realizar la clasificación del audio grabado del usuario.
- Nombre: Este campo realiza un filtro de los lugares obtenidos y muestra los que corresponden con este valor. La aplicación no utiliza este campo ya que restringe excesivamente el número de resultados. La filosofía de la aplicación es ofrecer posibilidades al usuario, acordes a sus preferencias, pero ofrecer un número amplio de alternativas.

En la Figura 44, se visualiza el código de la aplicación donde lanza el mapa con la búsqueda descrita anteriormente.

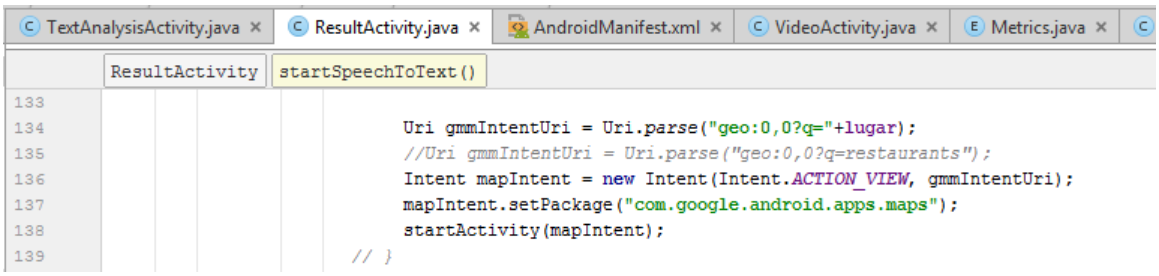


Figura 44: Query a través del Intent

Se ha escogido esta opción por su sencillez y mejor visualización a pesar de haber desarrollado el mapa incrustado.

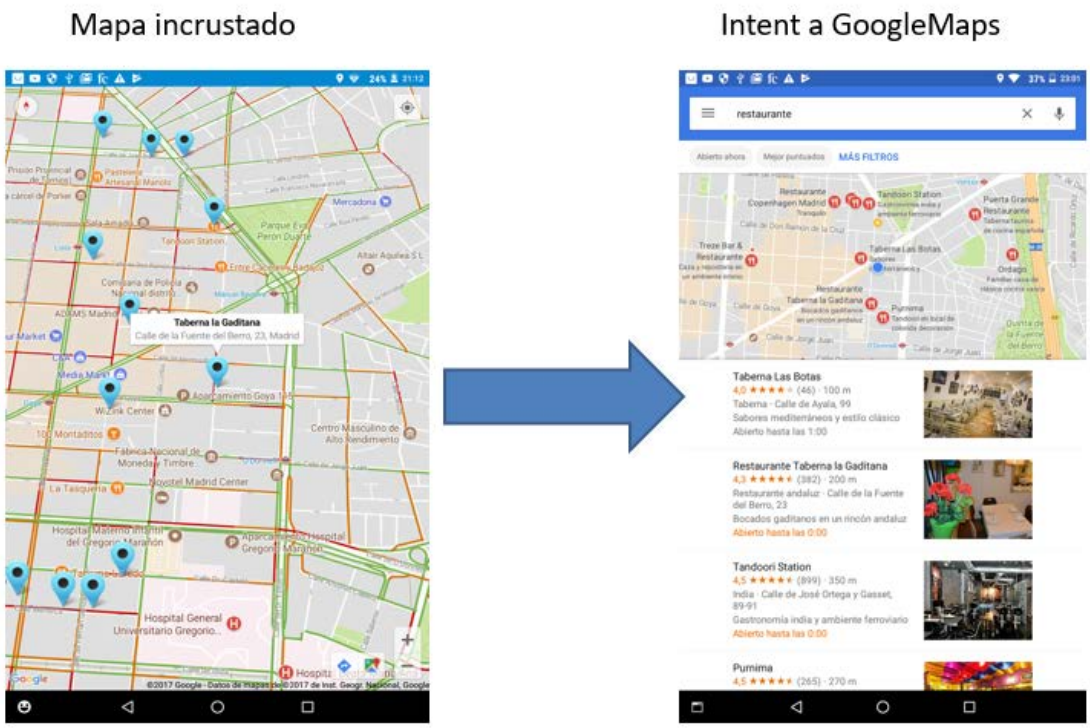


Figura 45: Comparativa entre los dos tipos de mapas

3.2.14. SettingActivity

Las aplicaciones generalmente incluyen una configuración que permite que los usuarios modifiquen las funciones y los comportamientos de las aplicaciones. Cada preferencia que se agrega tiene un par clave-valor correspondiente que el sistema utiliza para guardar la configuración en un archivo *SharedPreferences* [33] predeterminado para las configuraciones de la aplicación. Cuando el usuario cambia una configuración, el sistema actualiza el valor correspondiente en el archivo SharedPreferences.

Capítulo 4

4. Descripción detallada de los módulos del sistema

En este capítulo se describe de forma detallada cada uno de los módulos que componen la aplicación para Android desarrollada. Para cada módulo se detalla la funcionalidad, la arquitectura y flujo de datos. Además, se muestran ejemplos de posibles escenarios de uso.

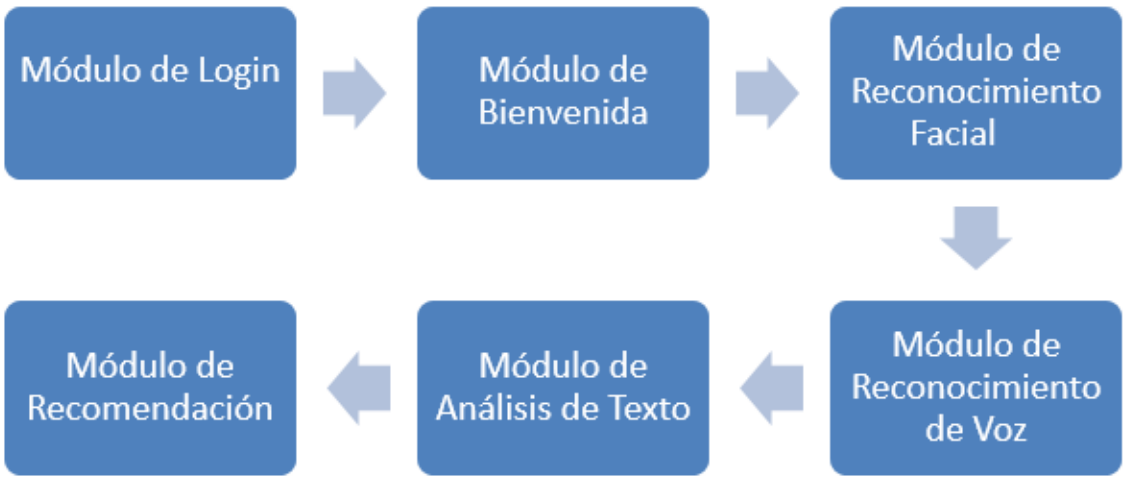


Figura 46:Arquitectura de la Aplicación

4.1. Módulo de login

En este módulo de inicio de sesión se detalla la arquitectura del mismo y el diseño realizado.

4.1.1. Arquitectura

La arquitectura de este módulo consiste en utilizar la librería Firebase para el manejo de los usuarios. Se divide en 4 módulos principales, el primero se encarga de la autenticación correo/contraseña con el inicio de sesión de usuario ya registrado, el segundo con nuevo usuario y los otros dos módulos son los encargados de iniciar sesión a través de Google y otro a través de Facebook.

Figura 47: Arquitectura del módulo de login

4.1.2. Diseño

Es el primer módulo con el que se encuentra el usuario al abrir la aplicación. Permite crear un usuario, iniciar una sesión y con la ayuda de la librería Firebase también puede entrar con su cuenta de Google o Facebook como se ha comentado antes en el apartado de arquitectura.

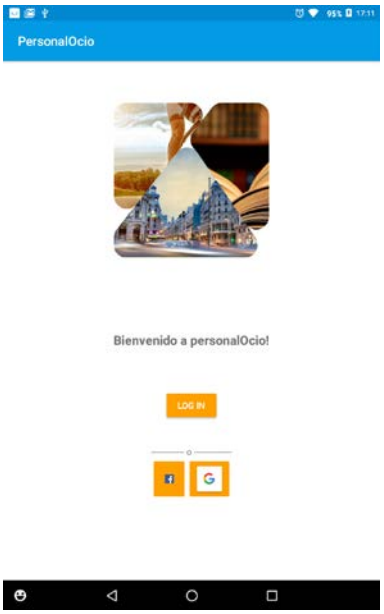


Figura 48: Login Activity

A continuación, se muestra la actividad para iniciar sesión que controla la gestión de usuarios nuevos como aquellos ya registrados. Cabe mencionar que se ha incluido un sencillo control de los datos introducidos.

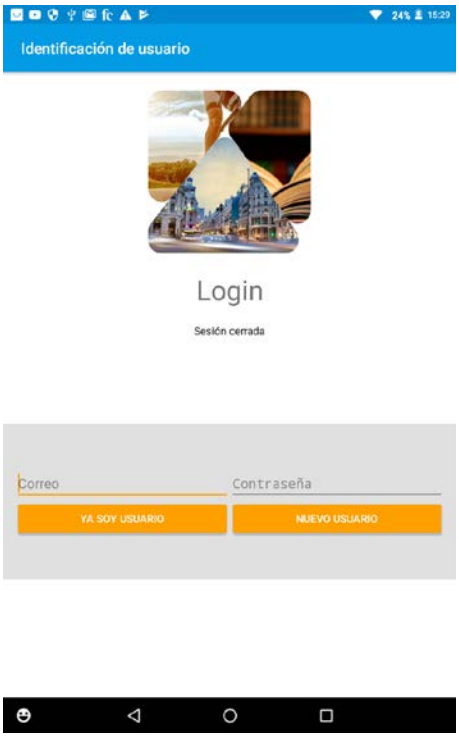


Figura 49: EmailPasswordActivity

4.2. Módulo de bienvenida

En este módulo de bienvenida se detalla la arquitectura del mismo y el diseño realizado a través de capturas realizadas en la Tablet (las características se encuentran en el apartado 2.4 del presente documento) como dispositivo utilizado.

4.2.1. Arquitectura

Este módulo es el más sencillo de todos, una vez autenticado el usuario puede solicitar las preferencias y almacenarlas. Se incluye un elemento calendario para proporcionar la funcionalidad de diario que se da a la aplicación.

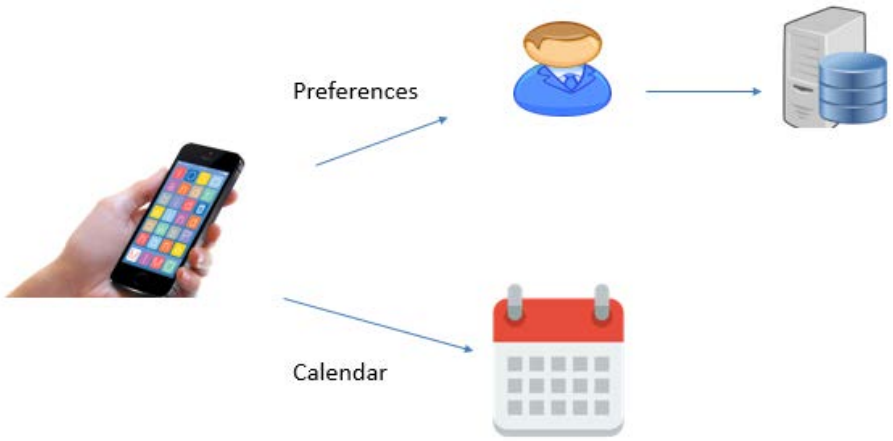


Figura 50: Arquitectura módulo de bienvenida

4.2.2. Diseño

Una vez iniciada la sesión por cualquier método de autenticación, aparece la página de bienvenida. Este módulo está planteado para tener un trato más cercano con el usuario, como un diario en el que puede contar a PersonalOcio cómo se siente hoy o que le apetece hacer.

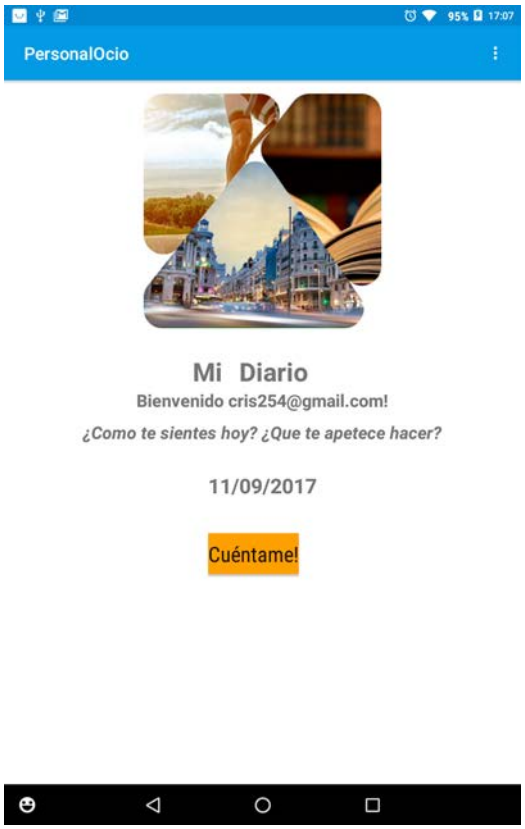


Figura 51: Wellcome Activity

Además, permite cambiar la configuración de los gustos de música, cine y deporte para que la aplicación pueda conocer un poco mejor al usuario. Y, por último, le prepara para comenzar el análisis.

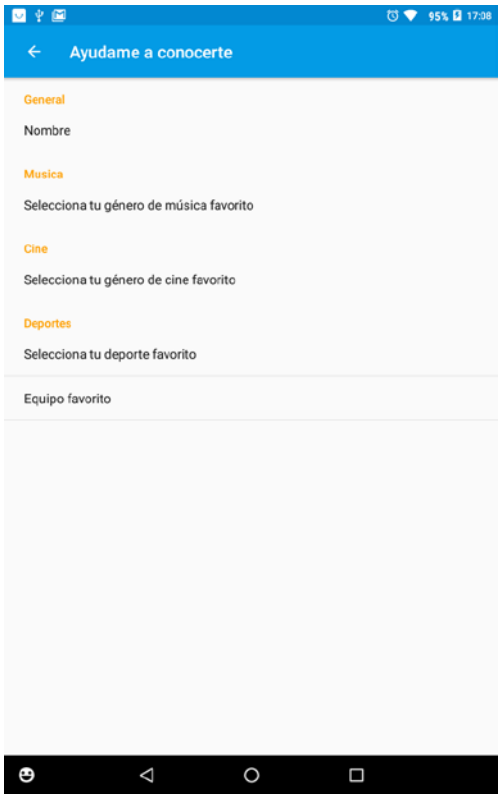


Figura 52: Preferences Activity

El usuario puede seleccionar sus preferencias a través de diálogos como el siguiente:

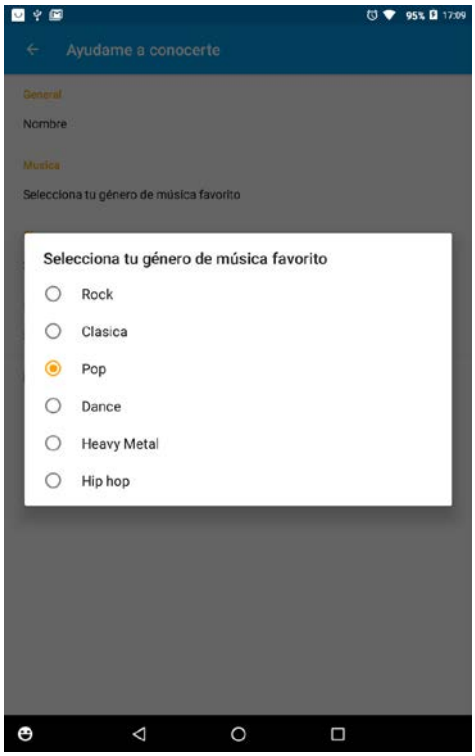


Figura 53: Selección de la preferencia sobre géneros de música

4.3. Módulo de reconocimiento facial

En este módulo de reconocimiento se detalla la arquitectura del mismo y el diseño realizado a través de capturas de la Tablet utilizada.

4.3.1. Arquitectura

En este módulo el usuario realiza un video contando sus gustos y aficiones. Durante este proceso el módulo de reconocimiento facial va enviando a la librería Affectiva los frame que captura con la previsualización de la cámara. Al mismo tiempo necesita almacenar el audio del usuario en Google Cloud Storage.

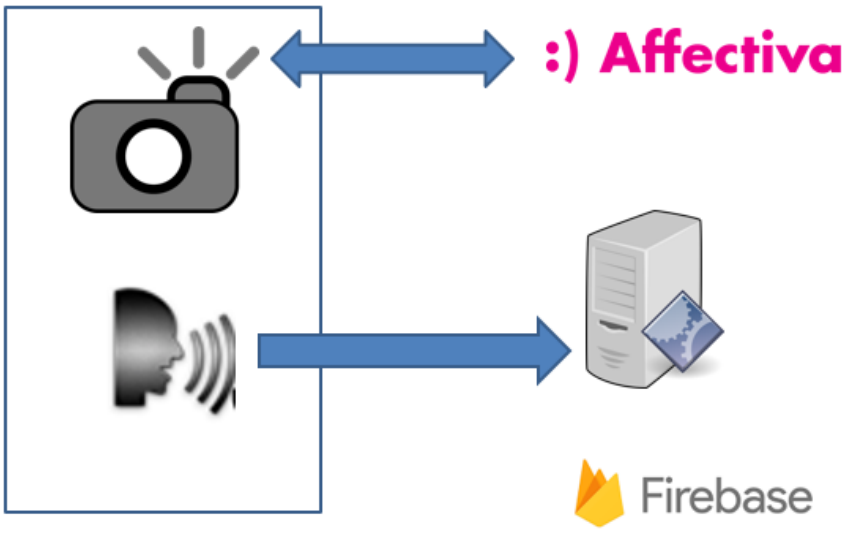


Figura 54: Arquitectura del módulo de reconocimiento facial

4.3.2. Diseño

Este módulo consiste en la previsualización de la cámara que captura los frames y se los manda a la librería de reconocimiento facial. Esta previsualización se activa con el botón “start”. Además, si el dispositivo lo permite puede seleccionar la cámara trasera o la frontal. En paralelo se guarda el audio para procesarlo en Google Cloud.

Figura 55: Video Activity

4.4. Módulo de reconocimiento de voz

En este módulo de reconocimiento se detalla la arquitectura del mismo y el diseño realizado a través de capturas de la Tablet utilizada.

4.4.1. Arquitectura

En este módulo se realiza el procesamiento del audio almacenado a texto. El inconveniente principal encontrado con el Recognizer Intent es la limitación de lanzarse en el hilo principal a pesar de la rapidez de conversión. Por este motivo se ha tenido que buscar otras alternativas como Google Speech. Para obtener el audio conecta directamente con la url indicada de Firebase Storage.

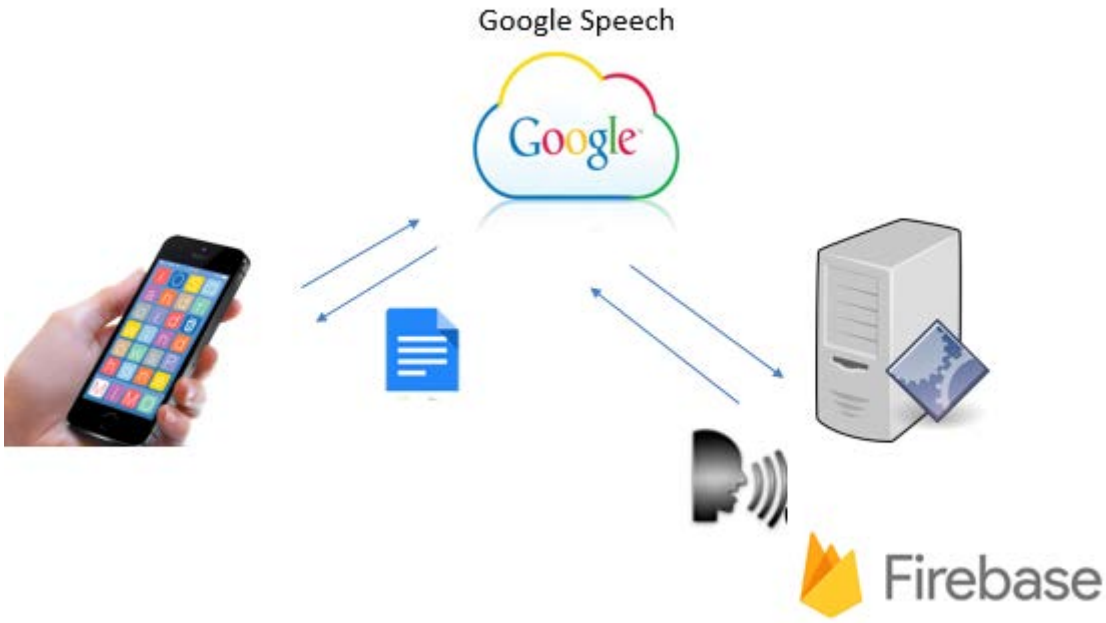


Figura 56:Arquitectura del módulo de reconocimiento de voz

4.4.2. Diseño

El diseño de este módulo es el siguiente. Lo primero que aparece es el proceso de conexión con Google Speech, una vez terminado la barra de proceso finaliza.

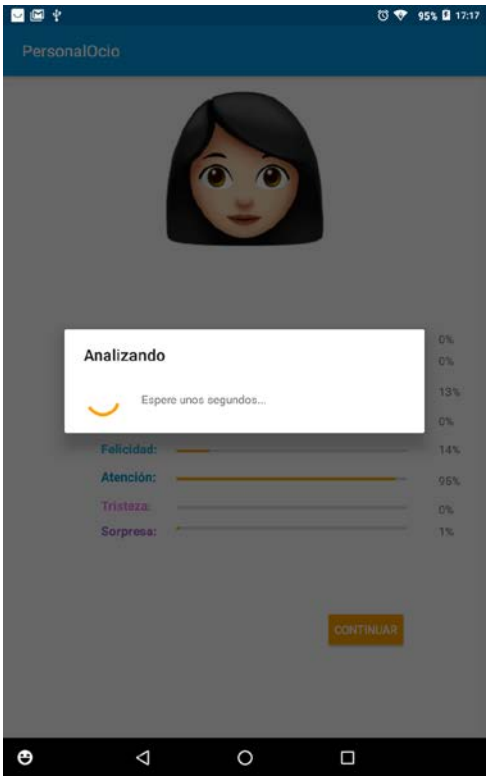


Figura 57: Progreso de la actividad Recognize

Además de realizar la conversión, se muestran los resultados tras el reconocimiento facial. Se muestra el porcentaje de probabilidad de que el usuario este en esa emoción. La librería ofrece muchos puntos de evaluación. Los escogidos para esta primera versión son los siguientes parámetros evaluados:

- Enfado
- Desprecio
- Disgusto
- Miedo
- Alegría
- Tristeza
- Sorpresa

Estos parámetros son almacenados en la base de datos de Google Firebase para llevar un diario o historial de seguimiento. En esta versión del proyecto, no se realiza ningún tipo de manipulación de esos datos. Pero en futuras versiones es interesante como modo terapéutico el poder llevar seguimiento emocional del usuario.

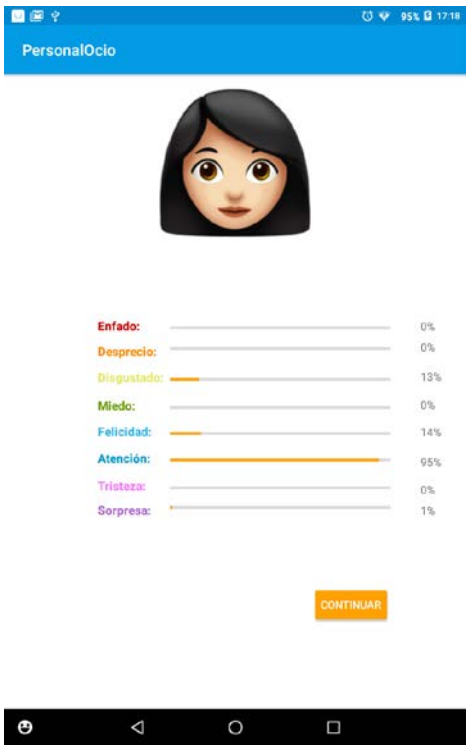


Figura 58: Recognize Activity

Como tratamiento de error, si no se realiza una identificación facial correcta del genero del usuario puede volver atrás para repetir el video y se muestra la siguiente captura:

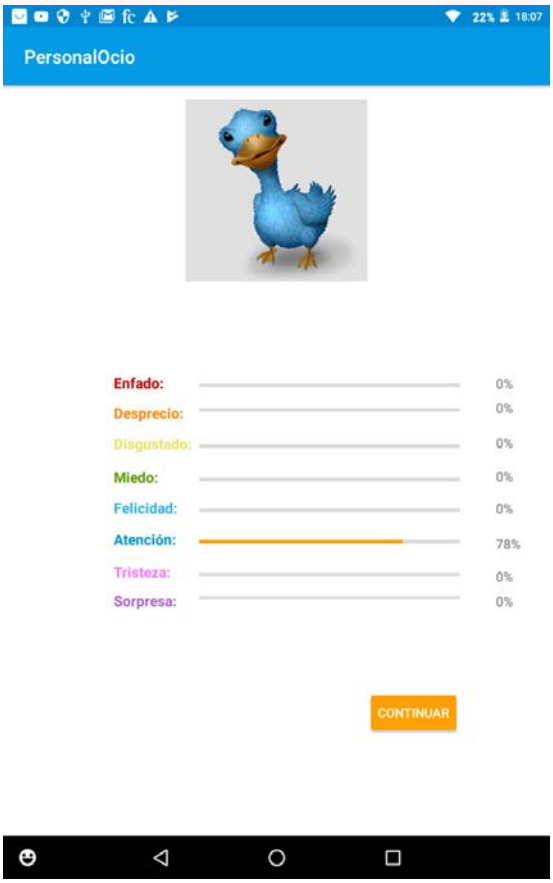


Figura 59: Género facial no reconocido

4.5. Módulo de análisis de texto

En este módulo de análisis de texto se detalla la arquitectura del mismo y el diseño realizado a través de capturas de la Tablet utilizada.

4.5.1. Arquitectura

En este módulo se realiza el análisis del sentimiento y de clasificación del texto a través de las librerías de Meaning Cloud y Google. Simplemente puntualizar que las conexiones de Meaning Cloud se realiza a través del protocolo HTTP y la respuesta es de tipo JSON, descritos ambos en el Capítulo 3.

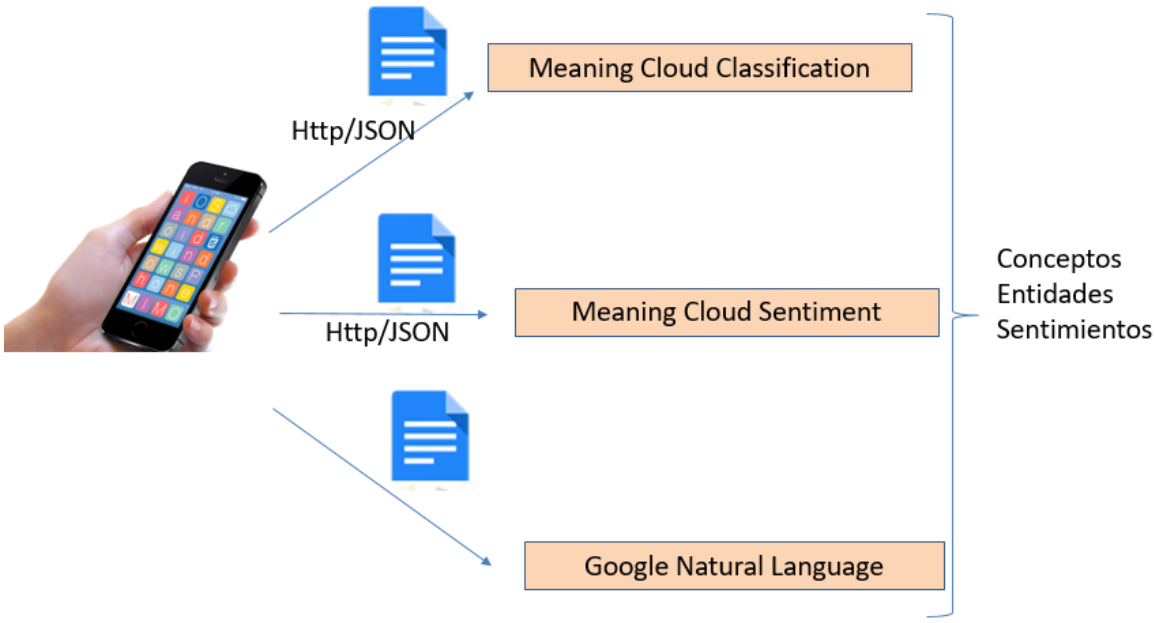


Figura 60: Arquitectura del módulo de análisis de texto

4.5.2. Diseño

En este módulo ya se ha convertido el audio a texto y se procesa a través de tres librerías el contenido y el análisis de sentimientos. Se utiliza la librería Google Natural Language que se muestra en conceptos. Y la clasificación de texto se realiza a través de la API MeaningCloud (referencia) con el modelo de datos creado MOcio que clasifica el texto en cine, restaurantes y deporte. Se ve reflejada en la pantalla en la parte inferior izquierda y se mostrará un icono diferente según la clasificación.

Las otras dos imágenes corresponden al resultado de la librería de análisis de sentimientos de MeaningCloud, donde identifica si el texto incluye ironía o no y la polaridad del texto. Según la polaridad el emoticono se ve modificado.

Figura 61: Text Analysis Activity

4.6. Módulo de recomendación

En este módulo de reconocimiento se detalla la arquitectura del mismo y el diseño realizado a través de capturas de la Tablet utilizada.

4.6.1. Arquitectura

En este módulo se une todos los datos almacenados del usuario y las preferencias introducidas por el mismo, se ha creado un sistema de recomendación de actividades. Se ha creado un clasificador de actividades donde se da más ponderación a aquellas que tiene como preferencia o las que pueden animar más al usuario según las métricas obtenidas.

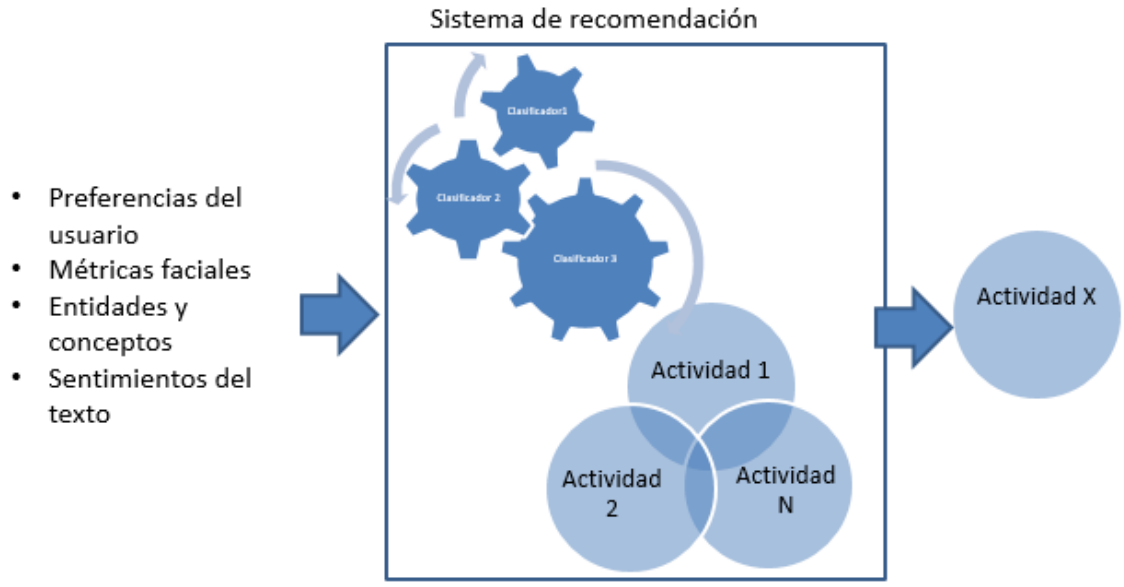


Figura 62:Arquitectura del módulo de recomendación

4.6.2. Diseño

Una vez obtenido todo el análisis del audio y video del usuario el módulo de recomendación entra en juego para ofrecer al usuario las mejores opciones. Si se ha detectado que está triste, PersonalOcio intentara animar al usuario con frases positivas, con su música favorita, a salir y contactar con algún amigo. Si, por el contrario, el usuario está contento se le anima a seguir así y la aplicación le ofrece alternativas diferentes de música.

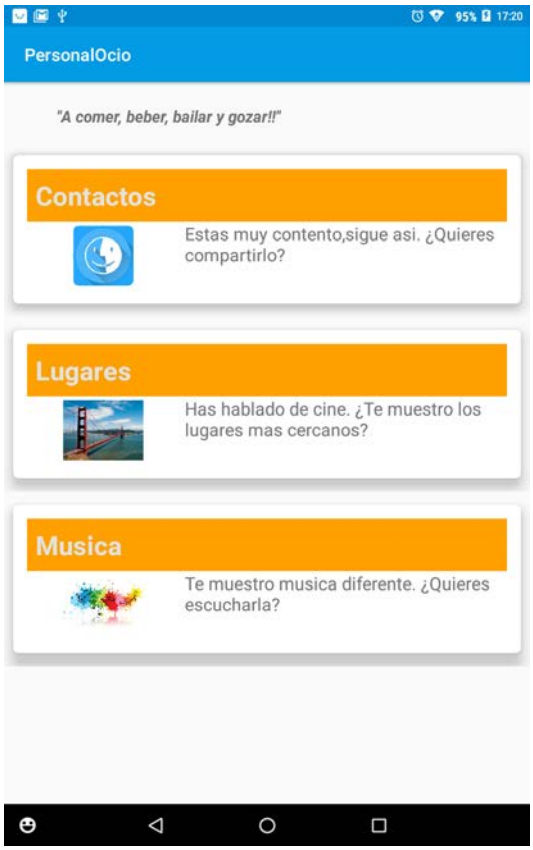


Figura 63: Results Activity

Las posibles actividades para mostrar al usuario se centran en tres grupos: que puede hacer, con quien puede realizarlo y qué música se ajusta mejor a lo que necesita.

En los comienzos se realizó la incorporación manual de los lugares extraídos de la conversación del usuario a través de Google Maps y Google Place como se menciona en el capítulo anterior. El resultado de la localización de lugares cercanos al usuario se puede observar en la Figura 64.

En cuanto a la actividad de música, se le ofrece distintas alternativas. Si se ha detectado que en su conjunto no está bien, la aplicación intentara animarle con su música favorita. En cambio, si disfruta de buen ánimo se le ofrecerá una música nueva para él. Y si, por el contrario, la aplicación no ha podido detectar ningún sentimiento reseñable se solicitará al usuario el tipo de música que quiere escuchar. Esto último se encuentra representado en la Figura 65.

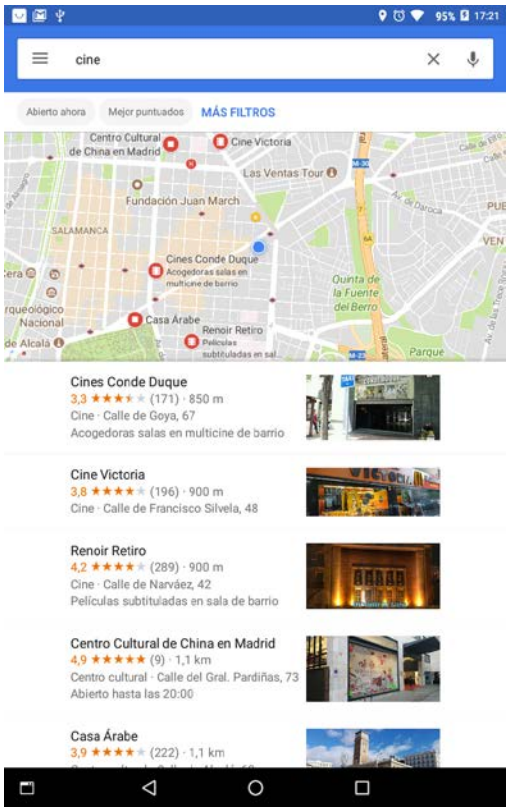


Figura 64: Resultado de la búsqueda en el Mapa

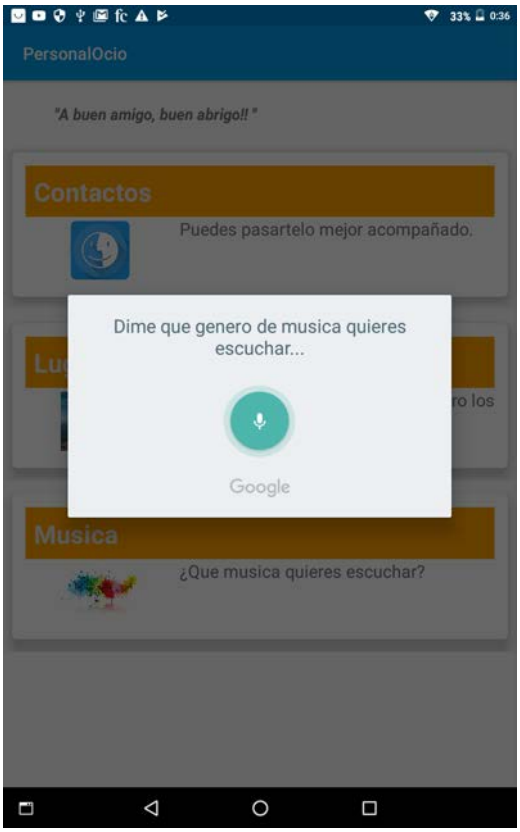


Figura 65: Solicita al usuario el género de música

Capítulo 5

5. Evaluación de la aplicación

En los puntos siguientes se describe la evaluación de la aplicación del proyecto fin de carrera. Para llevar a cabo la evolución se ha diseñado un cuestionario que recoge la impresión de los usuarios que la han utilizado, después se analizan los datos y finalmente se muestran las conclusiones.

5.1. Metodología de la evaluación

Para evaluar la aplicación Personal Ocio, se han diseñado un cuestionario que recoge las valoraciones subjetivas de los usuarios que interactúan con las dos aplicaciones Android desarrolladas en este proyecto.

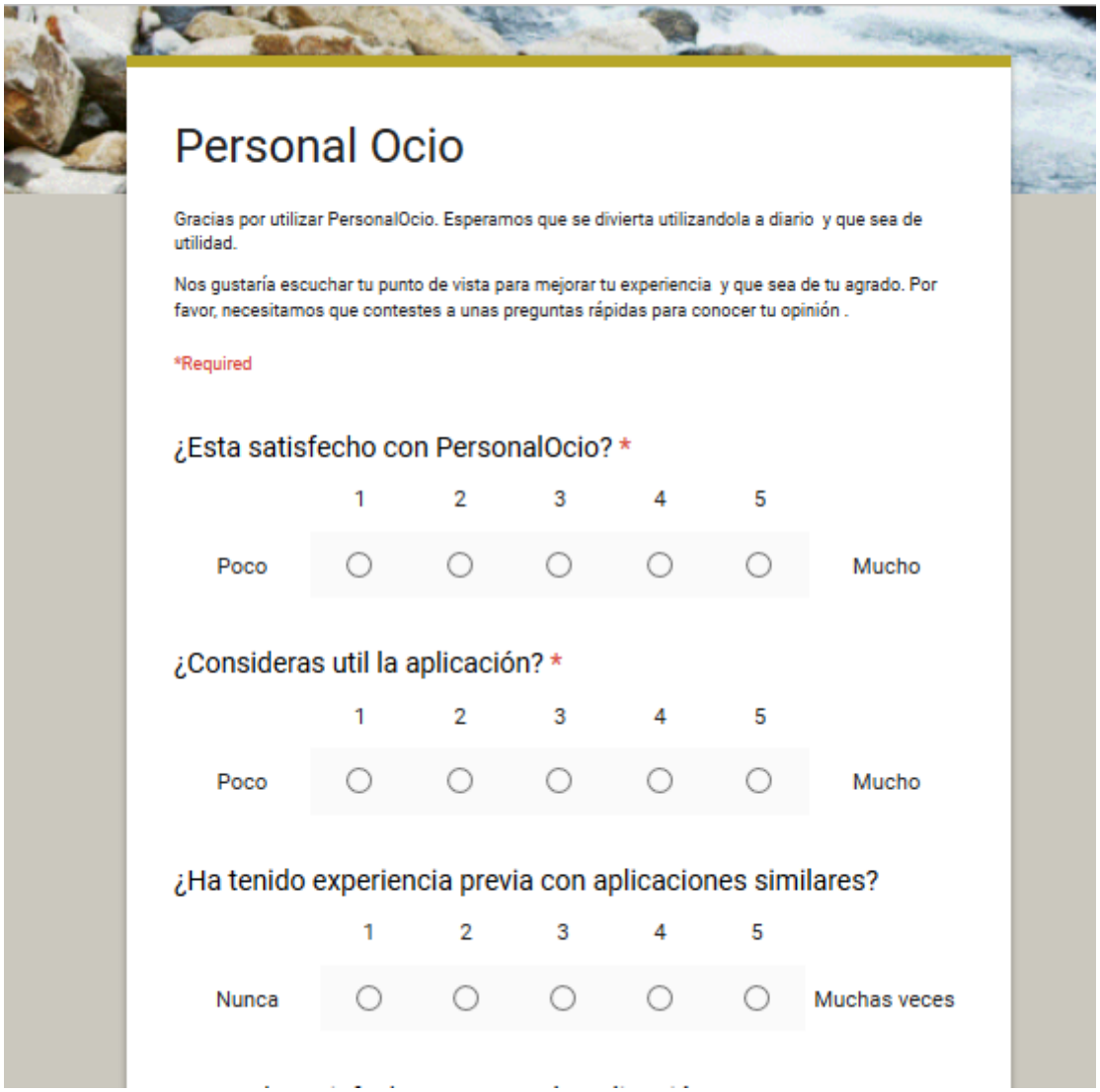
Las preguntas escogidas son acordes a los criterios de aceptación descritos en el análisis de la aplicación y a los requisitos funcionales localizados en el apartado 2.2 del presente documento. Estos requisitos cubren los aspectos principales de un sistema de recomendación, de análisis de sentimientos y reconocimiento de voz que se han descrito en la fase de análisis.

Además, se ha seguido el artículo “Evaluation of Mobile Interfaces as an Optimization Problem” [42] para incluir los puntos importantes de una aplicación

móvil y completar el cuestionario. Indica como puntos clave la importancia de la densidad de componentes (density), el espaciado y la correcta colocación de los elementos (regularity), orden según el patrón izquierda-derecha/arriba-abajo (sequence), similitud entre los elementos (grouping), sencillez (simplicity), buena distribución (homogeneity), simetría (symmetry) y coherencia(unity).

El cuestionario ha sido generado mediante la herramienta web Google Forms [87], proporcionada por Google de forma gratuita dentro de su suite ofimática en la nube, denominada Google Docs [88]. Una de sus ventajas principales es la posibilidad de generar las encuestas en la web y distribuir los enlaces de acceso público a los futuros usuarios encuestados.

Como estructura general, se han diseñado preguntas con cinco posibles respuestas para ambos cuestionarios, entre las cuáles, el usuario debe elegir la opción que más se ajuste a la realidad percibida.



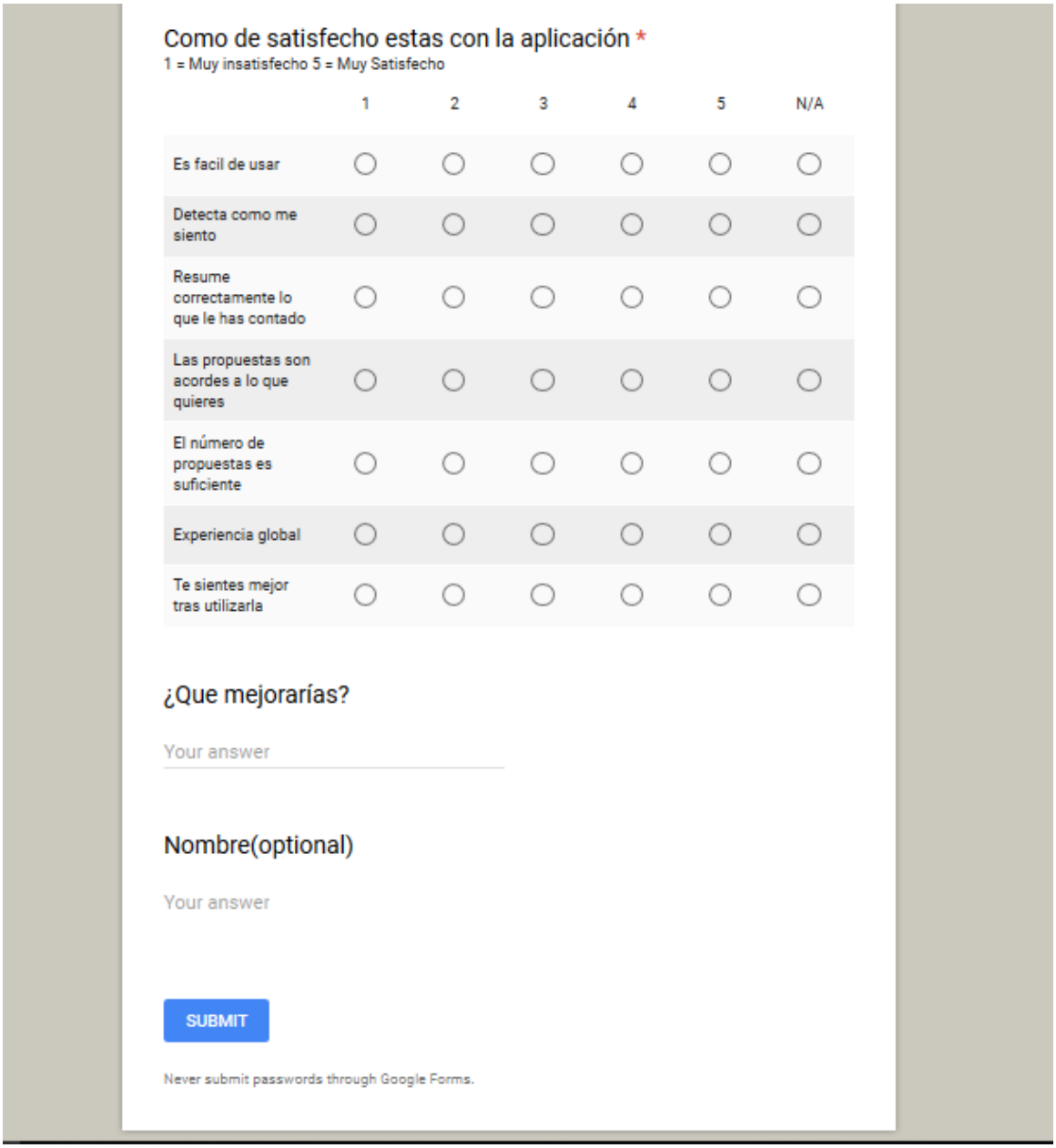


Figura 66: Formulario de evaluación [51]

5.2. Resultados

A continuación, se muestran los resultados de las 30 personas que han realizado el cuestionario. Google Form facilita esta labor, realizando un resumen por preguntas a modo de gráficos.

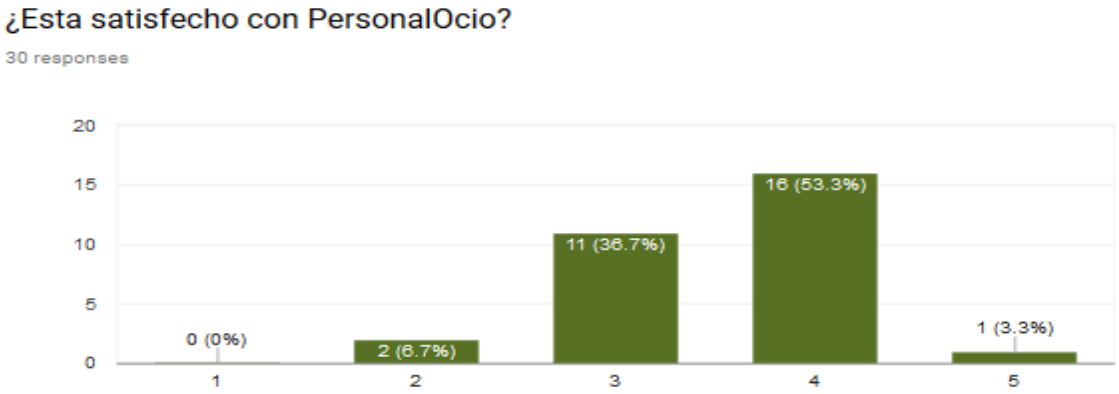


Figura 67: Resultado Pregunta 1

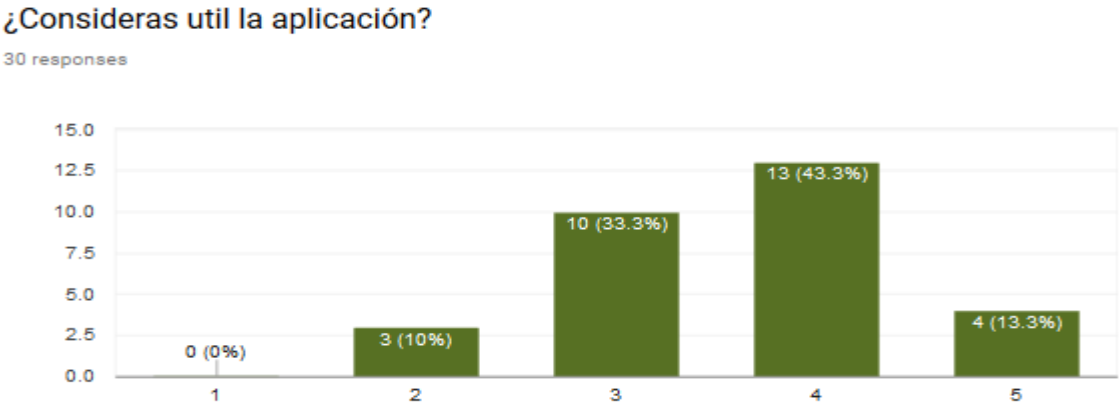


Figura 68: Resultado pregunte 2

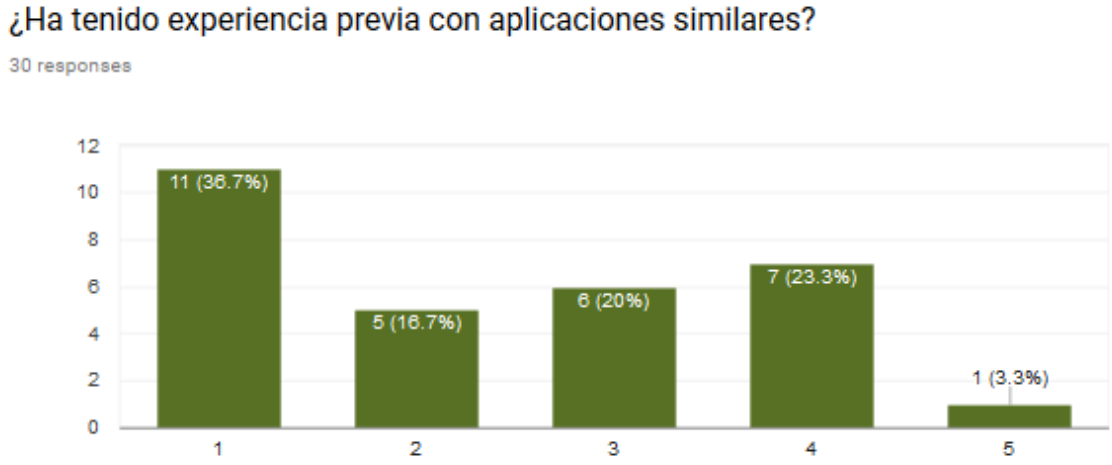


Figura 69: Resultado pregunte 3

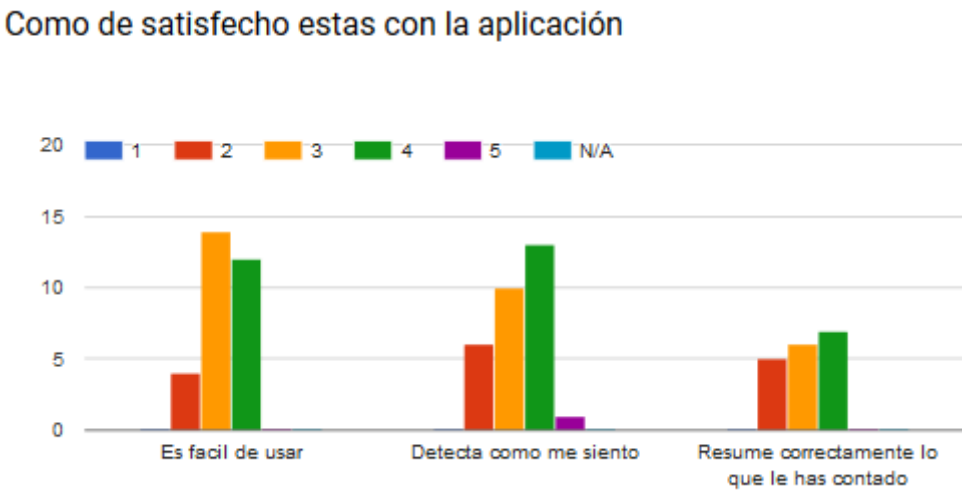


Figura 70: Pregunta 4,5 y 6

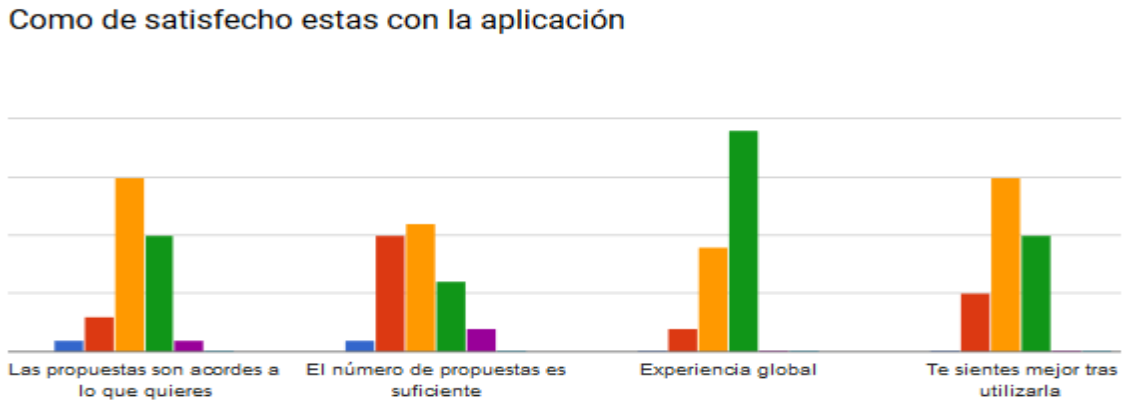


Figura 71: Pregunta 7,8,9 y 10

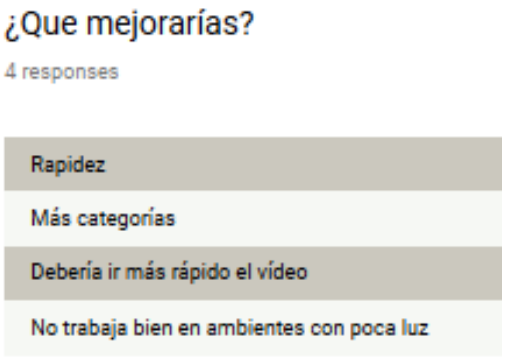


Figura 72: Pregunta abierta número 11

5.3. Conclusiones

Después de realizar los cuestionarios a usuarios muy dispares, comprendidos entre 20 a 65 años y obtener los resultados ofrecidos por Google Forms, se analizan los mismos. Las conclusiones obtenidas se detallan a continuación.

Por lo general, las personas que han probado la aplicación están contentas con el trabajo obtenido y les parece útil la aplicación.

En cuanto a la experiencia previa, hay menos homogeneidad en los resultados. La mayoría de los usuarios no conocen aplicaciones similares, sin embargo, hay un 23% que tiene un amplio conocimiento de aplicaciones similares.

Consideran que es fácil de usar la aplicación salvo un 13% que habría que analizar con más detenimiento. Pero no es crítico por no tener un valor significativo.

Cabe destacar que la detección de las emociones y el análisis de sentimientos, tienen unos resultados más igualados. Las respuestas no son tan claras como en otras preguntas. Este resultado cobra sentido en el marco de este proyecto ya que es una versión inicial y se utiliza un número pequeño de detectores faciales junto con un diccionario de texto limitado a tres campos de ocio únicamente (restaurantes, deportes y cine). Si el usuario ha utilizado temas diferentes a los manejados por la presente versión de la aplicación, se entiende que los resultados no son los más favorables posible. Sin embargo, el predominante, por poco, dé ambas preguntas es el nivel 4 de satisfacción.

La pregunta de si las propuestas son acordes a lo que quiere, indica que son positivas mientras que el número de propuesta es considerado pobre. Se muestras 3 propuestas de ocio y el resultado es acorde con el análisis preliminar del apartado 2.1 que indicaba como muestra de recomendación ideal a 5 y como comenta Ben Simón [36] en su artículo. Este punto es un factor crítico a la hora de considerar las mejoras de la aplicación.

Sin embargo, el ranking de la experiencia global es positiva y un 33 % considera que se siente mejor tras utilizar la aplicación.

En cuanto, a la última pregunta abierta algunos encuestados indicaron lo que mejorarían de la aplicación. Una de las respuestas es “más categorías” que es acorde a la limitación de los 3 tipos de ocio de este prototipo. Y otro punto que puede dar una pista al motivo obtenido de los resultados de las preguntas 5 y 6 es el del procesado del audio, cuando el usuario finaliza de grabar su video y espera los primeros resultados. Por lo tanto, este es otro punto crítico a la hora de plantear evolucionar la aplicación.

Otro usuario indica el problema de iluminación para el reconocimiento facial, siendo este punto otro motivo por el cual la pregunta 5 tiene un resultado más bajo. Sin embargo, este problema es uno de los encontrados en la fase de análisis y una de las limitaciones conocidas de estos sistemas de reconocimiento.

En resumen, el resultado global de la aplicación es positivo sin embargo se han detectado un par de puntos a mejorar.

Capítulo 6

6. Conclusiones y trabajo futuro

6.1. Conclusiones

Una vez concluida la revisión completa de la aplicación PersonalOcio, cuyo desarrollo constituía el objetivo principal del presente Proyecto Final de Carrera, se puede hacer un balance de los resultados obtenidos a partir de los objetivos parciales inicialmente marcados, que han ido necesarios para alcanzar el objetivo principal.

- Estudio de los sistemas de recomendación

En primer lugar, se ha realizado un estudio de los sistemas de recomendación. Se han visto las distintas formas en las que se puede destacar un elemento frente al resto, en función de la navegación y contenido visitado, de las compras realizadas por los usuarios o de las votaciones. Además, ha surgido la importancia del conocer las características del usuario (edad, sexo, situación geográfica, profesión, etc.). Este estudio ha sido necesario para incorporarlo a la aplicación desarrollada.

- Estudio de los sistemas de análisis de sentimientos

Una forma de obtener más información del usuario es conocer los sentimientos y emociones de la persona que está utilizando la aplicación. Se ha descrito en qué consiste el análisis de sentimientos, la arquitectura y su complejidad en conocer el estado de ánimo del usuario. Esto ha llevado a la necesidad de analizar los sistemas de dialogo y el reconocimiento facial.

- Estudio de los sistemas de reconocimiento facial

En cuanto a los sistemas de reconocimiento facial se han descrito las fases por las que pasa la imagen hasta que es conocida por una tipología concreta a partir de un cierto umbral. Después, se han comentado los principales algoritmos utilizados y por último las limitaciones sufren.

- Estudio de los sistemas de diálogo.

Se ha realizado un estudio de los sistemas de diálogo que incluye una descripción de la funcionalidad, la arquitectura y de los módulos que los conforman; un análisis de los requisitos que debe cumplir un sistema de diálogo ideal y de las limitaciones a las que están sujetos. Este estudio ha sido necesario para poder incorporar un reconocimiento de voz a la aplicación desarrollada, obtener toda la potencia de estos sistemas para su posterior análisis en cuanto a contenido sentimental y como principal característica del usuario a la hora de hacer recomendaciones de ocio.

- Estudio del mercado

De la teoría de los sistemas de recomendación, de dialogo, de reconocimiento facial y de análisis de sentimientos se ha pasado a mostrar algunas aplicaciones actuales que ponen en práctica la arquitectura de estos sistemas.

- Estudio de la plataforma Android.

En cuanto a la plataforma Android, se ha realizado un estudio exhaustivo de su arquitectura, componentes, características y funcionamiento ya que se desconocía todo lo relacionado con dicha plataforma antes de realizar el presente Proyecto Final de Carrera. Este objetivo se ha cumplido gracias a la completa documentación que ofrece Google a los desarrolladores consistente mayoritariamente en un desglose detallado de los paquetes, clases e interfaces que componen el SDK de Android. Además, se han consultado foros, blogs y otras publicaciones en Internet proporcionadas por desarrolladores de este sistema. En base al estudio realizado de las características de Android y a la experiencia adquirida tras el desarrollo de la aplicación, se exponen algunos aspectos que resultan ventajosos en su elección como plataforma:

Al ser un sistema de código abierto cualquiera puede estudiar, modificar, mejorar y distribuir el sistema Android sin ningún tipo de restricción. La comunidad de desarrolladores es muy activa y está creando continuamente soluciones para Android. Además, Android ofrece la opción al desarrollo privado mediante la publicación comercial de aplicaciones, permitiendo a cada desarrollador decidir cómo quiere distribuir su propio trabajo.

A pesar de que Google evita utilizar el término demasiado, el que se utilice el lenguaje de programación Java para desarrollar las aplicaciones para Android ayuda a que cualquier programador que tenga una mínima experiencia con el lenguaje de

programación Java pueda comenzar a programar aplicaciones para Android sin demasiada dificultad.

Cualquier aplicación Android es una combinación de componentes lo que ayuda a modularizar funcionalmente las aplicaciones. Android facilita el diseño de interfaces de usuario ya que incorpora una amplia variedad de elementos y ofrece la posibilidad de definir la interfaz tanto en el código como a través de documentos XML externos. Declarar el diseño de interfaz de usuario en XML en lugar de utilizar código en tiempo de ejecución es útil ya que permite crear diferentes diseños para diferentes tamaños u orientaciones de la pantalla.

El acceso a los recursos del dispositivo (Ej. Wii-Fi, servicio de reconocimiento de voz, motor de síntesis, cámara, etc.) es una tarea sencilla gracias a las bibliotecas API que Android ofrece en su SDK ya que ayudan a que el desarrollador no tenga que programar a bajo nivel para que una aplicación pueda acceder a los componentes de hardware de los dispositivos.

La plataforma Android ofrece un entorno de desarrollo completo que incluye un emulador de dispositivos, herramientas para la depuración, la memoria y perfiles de rendimiento, y Android Studio facilita enormemente la tarea de programación en este sistema. Así, pueden crearse proyectos completos para Android, incluyendo el manifiesto y la declaración de recursos externos.

Análisis de las posibilidades que ofrece Android para el desarrollo de aplicaciones que permitan la interacción con el usuario. Antes de comenzar con el desarrollo de la aplicación multimodal para Android, ha sido necesario realizar un análisis detallado de las posibilidades que ofrece la plataforma Android para integrar el reconocimiento facial, análisis de sentimientos y reconocimiento de voz en una aplicación. Ha sido necesario el conocimiento exhaustivo de las librerías ofrecidas por Affective y Meaning Cloud para llevar a cabo la aplicación

Tal y como se ha visto, Google con Android ha potenciado desde sus inicios la forma de interactuar con el sistema ofreciendo multitud de posibilidades al desarrollador para integrar la gestión de usuarios, análisis de sentimientos y reconocimiento de voz en una aplicación. Incluso se ha tenido que analizar el uso de mapas para mostrar las actividades recomendadas. Para realizar un análisis completo de estas posibilidades se ha partido de la documentación oficial ofrecida por Google sobre los paquetes android.speech, Google Speech, Google Natural Language, Google Storage, Google Database. Además, se han consultado libros, foros, blogs y otras publicaciones en Internet proporcionadas por desarrolladores de este sistema.

Partiendo del análisis efectuado, se ha aplicado la mejor de todas las posibilidades en cada caso.

• Estudio de otras tecnologías necesarias en el desarrollo de la aplicación para Android.

Además del estudio realizado de la plataforma Android y de la metodología de integración del reconocimiento facial, reconocimiento de voz y análisis de sentimientos en una aplicación Android, ha sido necesario realizar un estudio sobre

la integración de las siguientes funcionalidades en una aplicación Android: uso de tareas en segundo plano (AsyncTask), manejo de conexiones http y paseo de las respuestas, utilización de vista por tarjetas para la muestra de actividades, gestión de la actividad de preferencias en Android.

Cabe destacar algunas de las consideraciones que se han tenido en cuenta a la hora de diseñar la aplicación de recomendación de ocio a través de los sentimientos del usuario para Android con el objetivo de mejorar la interacción entre el usuario y la aplicación. En primer lugar y con el objetivo de que la interacción oral entre el usuario y la aplicación sea lo más simple posible, se ha evitado utilizar locuciones demasiado largas, se han utilizado ventanas con pocas opciones y un diseño muy visual. Además, la aplicación ofrece mecanismos de ayuda y sistemas de realimentación, que indican al usuario lo que debe hacer y que está ocurriendo a medida que avanza por los diferentes módulos de la aplicación. Por otro lado, para que el usuario se habitúe al modo de funcionamiento de la aplicación, se ha mantenido la coherencia y homogeneidad a lo largo de los diálogos.

La complejidad de la aplicación es elevada debido a la gran variedad de tecnologías y librerías utilizadas, la mayoría desconocidas con anterioridad a la realización del presente proyecto. A continuación, se explican las principales dificultades que han sido las responsables de añadir complejidad a la realización de la aplicación para Android.

Las librerías Emotion y Face 1.0 son librerías de pago, con una versión de prueba de pocos meses, con la ventaja de ofrecer procesamiento de video. Como consecuencia, se tuvo que descartar este módulo de la aplicación para utilizar la librería Affective como plan alternativo a pesar del tiempo empleado y el sobresfuerzo requerido.

RecognizerIntent, inicialmente, era la opción más factible y rápida para el reconocimiento de voz. Sin embargo, la limitación de utilizar el hilo principal de la actividad fue difícil de compaginar con la previsualización de la cámara y el reconocimiento facial. Por lo tanto, se optó por guardar el audio durante la captación facial y procesarlo a través de Google Speech posteriormente. Al tener que enviar el video al servidor conlleva tiempo e implica limitaciones en cuanto a tamaño de audio. Una de las principales críticas de la aplicación durante la fase de evaluación fue precisamente este punto, la lentitud en el procesado del audio.

En un principio, la vista de las actividades se iba a incluir en un mapa propio de la aplicación, pero una vez realizado, el resultado visual era muy pobre. Se tomó la decisión de descartarlo y utilizar Google Maps a pesar del esfuerzo dedicado.

El balance del proyecto es aceptable, se ha logrado realizar todos los requisitos planteados inicialmente. Y el resultado de la evaluación a través de los cuestionarios es positiva y se han alcanzado casi todos los criterios de aceptación planteado en la fase de análisis del proyecto.

6.2. Trabajo futuro

Finalizada la aplicación y el estudio del trabajo PFC se indican algunas mejoras obtenidas tras la evaluación de la misma:

- Incorporar nuevas categorías de ocio, ampliando el diccionario de texto.
- Añadir más tipos de recomendaciones a las ya incluidas de contactos, lugares cercanos y música. Incluir más propuestas, siendo como mínimo 5 el número total.
- Buscar alternativas a al procesado de audio, que no ralentice tanto la aplicación y mejorar la rapidez de respuesta para que este proceso sea transparente para el usuario.

Además, se sugieren una serie de nuevas ideas:

- Anadir variantes a la geolocalización: Ampliar la casuística de recomendación de actividades según el estado de ánimo del usuario y la geolocalización no solo a puntos cercanos. Por ejemplo, si está cansado proponerle actividades cercanas para que no tenga que moverse. Si dispone de más tiempo, incluso días y dentro de sus preferencias está el mar proponerle hoteles cercanos a la playa.
- Añadir TTS (Text to Speech): PersonalOcio podría leer las actividades obtenidas a través del asistente de voz. Proporcionaría una gran ayuda para personas con discapacidad visual.

- Añadir más tipos de actividades: En esta primera versión, solo se manejan actividades de ponerte en contacto con amigos, escuchar música o salir a comer, ir al cine o ver un partido. Otras actividades interesantes que puede ofrecer es animarle a hacer deporte, indicarle las rutas cercanas según las preferencias de campo o ciudad.

- Ampliar la aplicación a otros idiomas

- Permitir vídeos más largos que mejoren la rapidez

- Realizar un sistema de realimentación diario del perfil del usuario: Personal Ocio podría lanzar una o dos preguntas diarias sobre las preferencias del usuario, que haría en diferentes situaciones, escoger entre varias alternativas… para añadir información del usuario y permitir que la aplicación aprenda.

- Mostrar el histórico emocional del usuario; La aplicación almacena los datos recogidos en cada procesado del video, pero no se muestra ningún grafico o histórico de la evolución del mismo

Capítulo 7

7. Gestión del proyecto

El presente capítulo tiene como objetivo detallar la planificación del proyecto. Sirve como anexo al punto 2.3 de la Planificación

Se muestra a continuación, en la Figura 73, el diagrama de Gantt generado para la planificación. Esta herramienta permite realizar una estimación inicial y a medida que se van realizando las tareas permite reajustar tiempo y/o recurso. Se puede visualizar el estado de las tareas, completadas del 0% al 100%, calcular el coste por persona, visualizar tareas. Se puede observar los recursos incluidos calendario en una línea temporal en la Figura 74.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesor	Nombres d recursos
	Personal Ocio	**104 días**	**mar 09/05/17**	**vie 29/09/17**		
✓	**Analisis del proyecto**	**23 días**	**mar 09/05/17**	**jue 08/06/17**		
✓	Estudio de plataforma android	5 días	mar 09/05/17	lun 15/05/17		Cristina
✓	Estudio de analisis de sentimientos	4 días	mar 16/05/17	vie 19/05/17	3	Cristina
✓	Estudio de reconocimiento facial	3 días	lun 22/05/17	mié 24/05/17	4	Cristina
✓	Estudio de sistemas de recomendación	4 días	jue 25/05/17	mar 30/05/17	5	Cristina
✓	Estudio de sistemas de dialogo	3 días	mié 31/05/17	vie 02/06/17	6	Cristina
✓	Estudio de reconocimiento de voz	3 días	lun 05/06/17	mié 07/06/17	7	Cristina
✓	Analisis de requisitos	3 días	mié 31/05/17	vie 02/06/17	6	Cristina
✓	Planificación	2 días	lun 05/06/17	mar 06/06/17	9	Cristina
✓	Estimacion de costes	2 días	mié 07/06/17	jue 08/06/17	10	Cristina
	Seguimiento y control	**71 días**	**vie 12/05/17**	**vie 18/08/17**		
	Seguimiento y control 1	1 día	vie 12/05/17	vie 12/05/17		
	Seguimiento y control 2	1 día	vie 26/05/17	vie 26/05/17		
	Seguimiento y control 3	1 día	vie 09/06/17	vie 09/06/17		
	Seguimiento y control 4	1 día	vie 23/06/17	vie 23/06/17		
	Seguimiento y control 5	1 día	vie 07/07/17	vie 07/07/17		
	Seguimiento y control 6	1 día	vie 21/07/17	vie 21/07/17		
	Seguimiento y control 7	1 día	vie 04/08/17	vie 04/08/17		
	Seguimiento y control 8	1 día	vie 18/08/17	vie 18/08/17		
✓	**Solución Técnica**	**56 días**	**vie 09/06/17**	**vie 25/08/17**		
✓	**Diseño**	**14 días**	**vie 09/06/17**	**mié 28/06/17**		
✓	Diseño Funcional	7 días	vie 09/06/17	lun 19/06/17	11	Cristina
✓	Diseño Tecnico	7 días	mar 20/06/17	mié 28/06/17	23	Cristina
✓	**Desarrollo de la solución**	**42 días**	**jue 29/06/17**	**vie 25/08/17**		
✓	Desarrollo login	5 días	jue 29/06/17	mié 05/07/17	24	
✓	Desarrollo reconocimiento facial	7 días	jue 06/07/17	vie 14/07/17	26	Cristina
✓	Desarrollo analisis de sentimientos	7 días	lun 17/07/17	mar 25/07/17	27	Cristina
✓	Desarrollo reconocimiento de voz	7 días	mié 26/07/17	jue 03/08/17	28	Cristina
✓	Desarrollo Gramatica	2 días	vie 04/08/17	lun 07/08/17	29	Cristina
✓	Procesamiento de actividades	7 días	mar 08/08/17	mié 16/08/17	30	Cristina
✓	Desarrollo de presentacion de actividades	7 días	jue 17/08/17	vie 25/08/17	31	Cristina
✓	**Certificación- Ciclo de pruebas**	**10 días**	**lun 28/08/17**	**vie 08/09/17**		
✓	Pruebas unitarias	5 días	lun 28/08/17	vie 01/09/17	32	Cristina
✓	Pruebas integaración	5 días	lun 04/09/17	vie 08/09/17	34	
	Validación Final	**13 días**	**lun 11/09/17**	**mié 27/09/17**		
	Documentación Memoria	7 días	lun 11/09/17	mar 19/09/17	35	Cristina
	Presentación	5 días	mié 20/09/17	mar 26/09/17	37	
	Feedback cliente	1 día	mié 27/09/17	mié 27/09/17	38	Cristina
	Cierre del Proyecto	1 día	jue 28/09/17	jue 28/09/17	39	Cristina

Figura 73: Diagrama de Gantt del proyecto

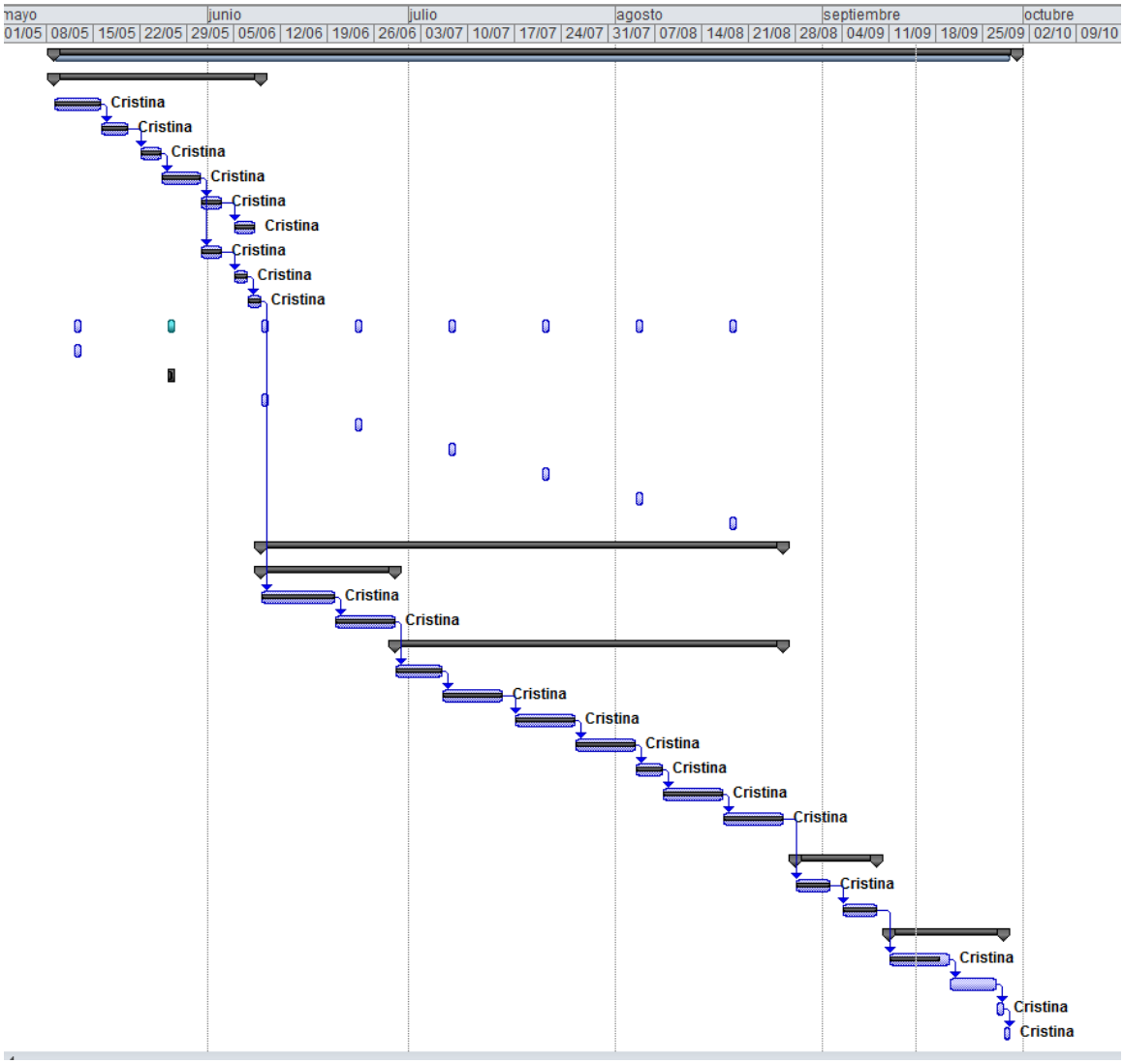


Figura 74: Planificación del proyecto en calendario

Capítulo 8

8. Presupuesto

En este capítulo se realiza un pequeño desglose de costes de personal, costes del material y costes totales. La obtención del presupuesto sigue la plantilla de "Presupuesto de Proyecto" proporcionada por la Universidad Carlos III de Madrid [35].

8.1. Costes de personal

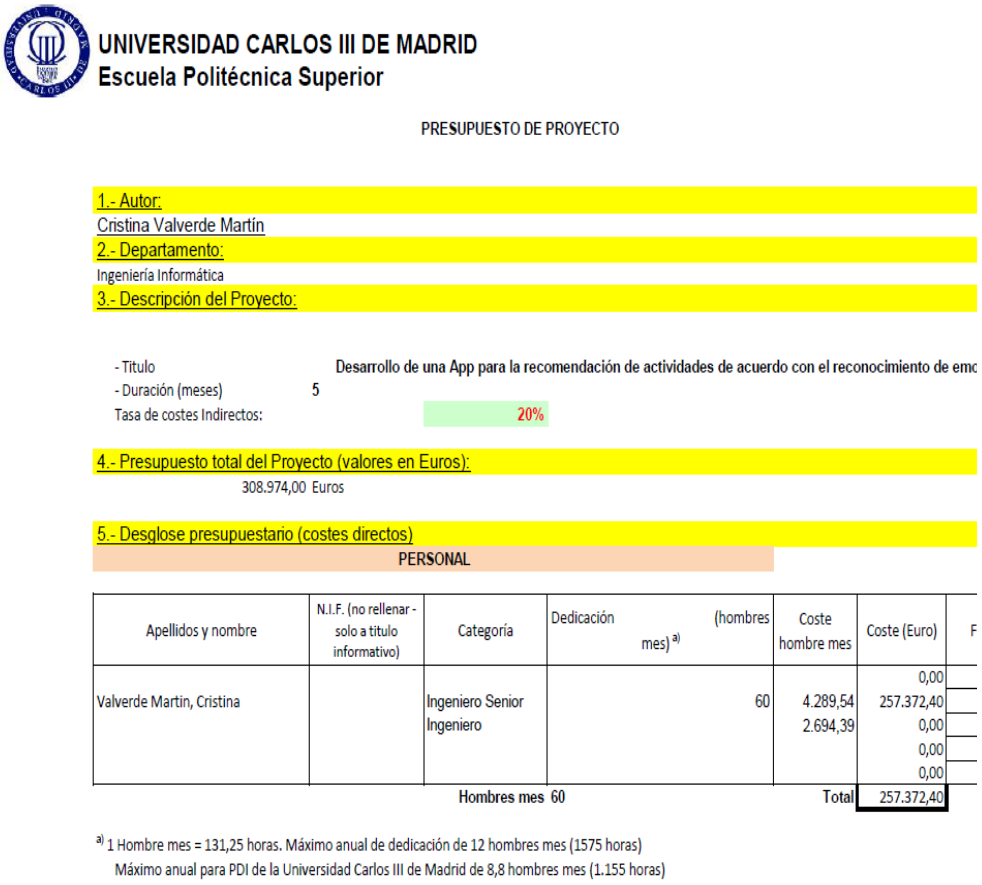
A partir de la planificación temporal, que aporta la información relativa a la duración total del proyecto, así como la dedicación por jornada de trabajo fijada en 3 horas diarias, de lunes a viernes puede calcularse el coste de personal resultante para un único ingeniero gracias al uso de la fórmula:

*Coste personal = (duración total*horas diarias) /dedicación mensual) *coste mensual por hombre*

Sustituyendo los siguientes valores:

- Duración total = 104 días
- Horas diarias = 3 horas
- Dedicación mensual = 60 horas
- Coste mensual por hombre (ingeniero) = 4.289,54 €

Se obtiene unos costes de personal de **257.372,40 €** como se observa en la Figura 48.



UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Cristina Valverde Martín
2.- Departamento:
Ingeniería Informática
3.- Descripción del Proyecto:

- Titulo: Desarrollo de una App para la recomendación de actividades de acuerdo con el reconocimiento de emc
- Duración (meses): 5
Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):
308.974,00 Euros

5.- Desglose presupuestario (costes directos)
PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	F
					0,00	
Valverde Martin, Cristina		Ingeniero Senior	60	4.289,54	257.372,40	
		Ingeniero		2.694,39	0,00	
					0,00	
					0,00	
		Hombres mes	60	Total	257.372,40	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

Figura 75: Coste de personal del proyecto

8.2. Coste de recursos

8.2.1. Costes Hardware

Los recursos mostrados en el apartado de recursos tienen los siguientes costes de adquisición hardware:

- Ordenador portátil → 1100€
- Tablet Android → 150€
- Cable USB. --> 4€

8.2.2. Coste Software

Los recursos mostrados en el apartado de recursos tienen los siguientes costes de adquisición software:

- Entorno de desarrollo Android Studio.
- JDK (Java Development kit): kit de desarrollo de Java.
- Servicio de alojamiento de archivos multiplataforma Dropbox.
- Microsoft Office 2010→ 85,99 € todo el paquete office

Una vez descritos los recursos necesarios para llevar a cabo el proyecto. Se realiza un cálculo de la amortización del equipo utilizado siguiendo la fórmula:

$$(A/B)\ \text{x C x D}$$

Dónde:
A = Número de meses desde la fecha de facturación en que el equipo es utilizado.
B = Periodo de depreciación.
C = Coste del equipo sin IVA.
D = % del uso que se dedica al proyecto (habitualmente 100%).

Se obtiene unos costes de recursos de **105,93€** como se puede ver en la Figura 62.

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Ordenador AsusTek	1.100,00	100	5	60	91,67
Tablet Android	150,00	100	5	60	12,50
Cable USB	4,00	100	5	60	0,33
Microsoft Office 2018	85,99	100	1	60	1,43
		100		60	0,00
					0,00
				Total	105,93

Figura 76: Coste de recursos del proyecto

8.3. Coste total

Para finalizar, además de los costes calculados anteriormente se incluyen en el presupuesto final los costes indirectos del proyecto, calculados sobre un 20% así como un 21% de I.V.A aplicado al total. Estos costes indirectos ascienden a **51.496 €**

En la Figura 77, se muestra el resumen de los costes descritos en los apartados anteriores y el coste final de todo el proyecto.

6.- Resumen de costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	257.372
Amortización	106
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	51.496
Total	308.974

Figura 77: Coste total del proyecto

El presupuesto total de este proyecto asciende a la cantidad de **TRESCIENTOS OCHO MIL NOVECIENTOS SETENTA Y CUATRO** EUROS.

Leganés a 20 de septiembre de 2017

Fdo. David López Muñoz

9. Glosario

SA	*Sentiment Analysis*
WIFI	*Wireless Fidelity*
XML	*Extensible Markup Language*
ADSL	*Asymmetric Digital Suscriben Line*
API	*Application Programming Interface*
ASCII	*American Standard Code for Information Interchange*
ASP	*Application Service Provider*
DT	*Decision Tree*
NN	*Neural Networking*
FCE	*Frequently Co-occurring Entropy*
TF-IDF	*Term frequency – Inverse document frequency*
SDK	*Software Development Kit*
RAM	*Random Access Memory*
RAH	*Reconocedor Automático del Habla*
JSON	*JavaScript Object Notation*
JDK	*Java Development Kit*
JAR	*Java ARchive*
IDE	*Integrated Development Environment*
HTML	*HyperText Markup Language*
APK	*Application PacKage File*
ASR	*Automatic Speech Recognition*
ART	*Android Runtime*
BBDD	*Bases de Datos*
RS	*Recommender System*

Referencias

[1] López Cózar Delgado, Ramón, “Sistemas de diálogo hablado y multimodal” Disponible en http://www.ugr.es/~rlopezc/sistemas_dialogo.htm [Septiembre 2017]

[2] “JAVA y Android, Una relación de amor-odio”, Artículo de opinión. Disponible en https://elandroidelibre.elespanol.com/2013/03/java-y-android-una-relacion-de-amor-odio.html [Septiembre 2017]

[3] Microsoft. “Conoce a Cortana”, microsoft.com, 2014. Disponible en: https://www.windowsphone.com/es-ES/How-to/wp8/cortana/meet-cortana [Septiembre 2017]

[4] Ahmad Ramsés Barragán Estrada y Cinthya Itzel Morales Martínez “PSICOLOGÍA DE LAS EMOCIONES POSITIVAS: GENERALIDADES Y BENEFICIOS”. Disponible en http://www.redalyc.org/articulo.oa?id=29232614006 [2014]

[5] Enrique García Fernández Abascal, “Disfrutar de las emociones positivas: Psicología” .Año 2015. ISBN: 9788494398025

[6] James Revelo, “Operaciones HTTP En Android Con El Cliente HttpURLConnection”. Disponible en http://www.hermosaprogramacion.com/2015/01/android-httpurlconnection/ [Septiembre 2017]

[7] Anupam Chugh, “OkHttp Android Example Tutorial”. Disponible en http://www.journaldev.com/13629/okhttp-android-example-tutorial. [Septiembre 2017]

[8] Miguel Catalan Bañuls, “Trabajando con JSON en Android”. Disponible en https://developer.android.com/reference/org/json/JSONObject.html. [Septiembre 2017]

[9] Andrés González, “Sistemas de recomendación de contenido con Machine Learning”. Disponible en http://cleverdata.io/sistemas-recomendacion-machine-learning/. [Septiembre 2017]

[10] SgOlier, “Android: Thread y AsyncTask”. Disponible en

http://www.sgoliver.net/blog/tareas-en-segundo-plano-en-android-i-thread-y-asynctask/ [Septiembre 2017]

[11] Kumar Ravi, Vadlamani Ravi, “A survey on opinion mining and sentiment analysis: Tasks, approaches and applications”. Artículo de investigación. Disponible en http://www.sciencedirect.com/science/article/pii/S0950705115002336 [Noviembre 2015, Pages 14-46]

[12] Jean Éthier, Pierre Hadaya, Jean Talbot y Jean Cadieux, “Interface design and emotions experienced on B2C Web sites: Empirical testing of a research model”. Disponible en http://www.sciencedirect.com/science/article/pii/S074756320800071X [17 September 2008, Pages 2771-2791]

[13] Android. “Guía de referencia de desarrolladores”. Disponible en https://developer.android.com/index.html

[14] Alla Yankouskaya, Moritz Stolte, Zargol Moradi, Pia Rotshtein y Glyn Humphreys. “Integration of identity and emotion information in faces: fMRI evidence”. Disponible en http://www.sciencedirect.com/science/article/pii/S0278262616303098 [August 2017, Pages 29-39]

[15] Konstantinos Kafetsios, Despoina Chatzakou ,Nikolaos Tsigilis y AthenaVakali. “Experience of emotion in face to face and computer-mediated social interactions: An event sampling study”. Disponible en http://www.sciencedirect.com/science/article/pii/S0747563217304557 [Noviembre 2017, Pages 287-293]

[16] Lars Vogel, “Getting started with Android development – Tutorial”. Disponible en http://www.vogella.com/tutorials/Android/article.html[Septiembre 2017]

[17] José Mendiola Zuriarrain, “Android ya es el sistema operativo más usado del mundo”. Disponible en https://elpais.com/tecnologia/2017/04/04/actualidad/1491296467_396232.html [Septiembre 2017]

[18] IAvilaE, “Programación Android, ciclo de vida de una app”. Disponible en http://www.proyectosimio.com/es/programacion-android-ciclo-de-vida-de-una-app/ [Septiembre 2017]

[19] Google. “Face Detection Concepts Overview”. Disponible en https://developers.google.com/vision/face-detection-concepts#top_of_page [Septiembre 2017]

[20] Rosalind Picard, “Los ordenadores emocionales”. ISBN: 9788434411845 [1998]

[21] Google. “Google Cloud Vision API Documentation”. Disponible en https://cloud.google.com/vision/ [Septiembre 2017]

[22] Microsoft. “Emotion Recognition API”. Disponible en https://azure.microsoft.com/es-es/services/cognitive-services/emotion/ [Septiembre 2017]

[23] Microsoft. “Face API - V1.0”. Disponible en https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f30395236

[24] Affectiva. “SDK & API”. Disponible en https://www.affectiva.com/product/emotion-sdk/ [Septiembre 2017]

[25] Google, “Firebase Authentication”. Disponible en https://firebase.google.com/docs/auth/?hl=es-419 [Septiembre 2017]

[26] Meaning cloud, “Sentiment analysis response”. Disponible en https://www.meaningcloud.com/developer/sentiment-analysis/doc/2.1/response [Septiembre 2017]

[27] Meaning cloud, “Sentiment aAnalysys rRequest”. Disponible en https://www.meaningcloud.com/developer/sentiment-analysis/doc/2.1/request [Septiembre 2017]

[28] Meaning cloud, “Text classification request”. Disponible en https://www.meaningcloud.com/developer/text-classification/doc/1.1/request [Septiembre 2017]

[29] Meaning cloud, “Text classification response”. Disponible en https://www.meaningcloud.com/developer/text-classification/doc/1.1/response [Septiembre 2017]

[30] Tutorial point. “Android - Location Based Services”. Disponible en https://www.tutorialspoint.com/android/android_location_based_services.htm [Septiembre 2017]

[31]Google, “Intenciones de Google Maps”. Disponible en https://developers.google.com/maps/documentation/android-api/intents?hl=es-419. [Septiembre 2017]

[32] Google, “Seleccionador de sitios”. Disponible en https://developers.google.com/places/android-api/placepicker?hl=es-419. [Septiembre 2017]

[33] Android, “Actividad de Configuración”. Disponible en https://developer.android.com/guide/topics/ui/settings.html [Septiembre 2017]

[34] Google, “Google Place API”. Disponible en https://developers.google.com/places/?hl=es-419 [Septiembre 2017]

[35] Universidad Carlos III de Madrid , “Presupuesto de Proyecto” . Disponible en http://portal.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20(3)_1.xlsx. [Septiembre 2017]

[36] David Ben-Shimon, Lior Rokach y Bracha Shapira, “An ensemble method for top-N recommendations from the SVD”. Disponible en http://www.sciencedirect.com/science/article/pii/S095741741630375X [1 Diciembre 2016, Pages 84-92]

[37] Kostas Kolomvatsos, Christos Anagnostopoulos y Stathes Hadjiefthymiades. “An efficient Recommendation System based on the Optimal StoppingTheory” Disponible en http://www.sciencedirect.com/science/article/pii/S0957417414002620 [Noviembre 2014, Pages 6796-6806]

[38] David Griol, Zoraida Callejas, Ramón López-Cózar y Giuseppe Riccardic “A domain-independent statistical methodology for dialog management in spoken dialog systems”. Disponible en http://www.sciencedirect.com/science/article/pii/S0885230813000612. [Mayo 2014, Pages 743-768]

[39] Joaquim Llisterri, “Los sistemas de diálogo”. Disponible en http://liceu.uab.cat/~joaquim/speech_technology/tecnol_parla/dialogue/dialogo_general/sistemas_dialogo.html#Un_sistema_de_di_logo_ideal [Septiembre 2017]

[40] Wikipedia, “Modelo oculto de Márkov”. Disponible en http://es.wikipedia.org/wiki/Modelo_oculto_de_M%C3%A1rkov[Septiembre 2017]

[41] Haşim Sak, Andrew Senior, Kanishka Rao, Françoise Beaufays y Johan Schalkwyk .”Google voice search: faster and more accurate “. Disponible en https://research.googleblog.com/2015/09/google-voice-search-faster-and-more.html [Septiembre 2017]

[42] Gasmi Ines, Soui Makram, Chouchane Mabrouka y Abed Mourad “Evaluation of Mobile Interfaces as an Optimization Problem”. Disponible en http://ac.els-cdn.com/S1877050917316393/1-s2.0-S1877050917316393-main.pdf?_tid=8c35f3f6-957a-11e7-9b15-00000aab0f6c&acdnat=1504974102_a9c8bac43f9e78f161e8786b780f8e73 [2017, Pages 235-248]

[43] Fabiola Becerra-Riera, Annette Morales-González y Heydi Méndez-Vázquez, “Facial marks for improving face recognition”. Disponible en http://www.sciencedirect.com/science/article/pii/S0167865517301423 [Mayo 2017]

[44] Guangtao Cheng y Zhanjie Song, “Robust face recognition based on sparse representation in 2D Fisherface pace”. Disponible en http://www.sciencedirect.com/science/article/pii/S0030402614000126 [June 2014, Pages 2804-2808]

[45] Müge Çarıkç, Figen Özen “A Face Recognition System Based on Eigenfaces Method”. Disponible en http://www.sciencedirect.com/science/article/pii/S2212017312000242 [2012, Pages 118-123]

[46] Muhammad Aurangze Khan, Costas Xydeas y Hassan Ahmed, "On the estimation of face recognition system performance using image variability information". Disponible en http://www.sciencedirect.com/science/article/pii/S0030402617302061 [Mayo 2017, Pages 619-632]

[47] Google. Consola de administración de PersonalOcio. Disponible en https://console.firebase.google.com/project/personalocio-174116/overview?hl=es-419

[48] Google,” Firebase almacenamiento”. Disponible en https://firebase.google.com/docs/storage/android/start?hl=es-419 [Septiembre 2017]

[49] Google, “Firebase realtime database”. Disponible en https://firebase.google.com/docs/database/android/start/?hl=es-419 [Septiembre 2017]

[50] Almudena Ruiz Iniesta. Sistemas de recomendación. Disponible en: http://gaia.fdi.ucm.es/files/people/almudena/seminario/recsys-dia1.pdf [Septiembre 2017]

[51] Google Form, cuestionario de Personal Ocio. Disponible en https://docs.google.com/forms/d/e/1FAIpQLScd8nukICZd3k2J_TJhuYXEUE9lyBYDjh2lXtpMpRY1Qb2Fig/viewform [Septiembre 2017]

[52] Portal Administración Electrónica- Ministerio de Hacienda y Administraciones Públicas, “Métrica v.3” Disponible en https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html [Septiembre 2017]

GLOSARIO