CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco

Departamento de Computación

Activación multimodal de servicios en dispositivos móviles

Tesis

Que presenta la

Ing. Viridiana Ponce Angulo

Para obtener el grado de

Maestra en Ciencias en Computación

Director de Tesis:

Dr. José Guadalupe Rodríguez García

México,Distrito Federal **Septiembre, 2012**

Resumen

Los dispositivos móviles están convirtiéndose en una herramienta importante de comunicación y de trabajo en la vidas de las personas. El número de usuarios de los denominados *smartphones* va en aumento, al igual que las capacidades y los recursos que integran estos dispositivos. Esta realidad plantea grandes desafíos a las áreas de cómputo móvil (CM) e interfaces hombre computadora (HCI por sus siglas en inglés). En CM el principal desafío se tiene en el diseño de dispositivos que mejoren la capacidad de cómputo y optimización de los recursos, por otra parte, en HCI el principal desafío se presenta en el diseño de interfaces de usuario multimodales para estos dispositivos. Este trabajo de tesis plantea la necesidad de contar con mecanismos multimodales proactivos de activación de servicios para dispositivos móviles, que proporcionen una interacción más natural con el usuario aún cuando éste tenga algún tipo de impedimento y que, al mismo tiempo, explote las capacidades de cómputo a través de los diversos sensores disponibles en esta clase de dispositivos.

Para dar respuesta a esta necesidad, se desarrollo un sistema multimodal de activación de servicios para dispositivos móviles basados en Android, que permite al usuario definir comandos en las modalidades de audio, gesto físico, texto y gesto táctil, y asociarlos a una aplicación o contacto disponible en el dispositivo. El usuario puede utilizar los comandos definidos, ya sea de forma aislada para activar los servicios de ejecutar un aplicación o mostrar la información de un contacto o de forma combinada para llamar o enviar un mensaje a un contacto específico. Además el sistema incorpora la cualidad de proactividad en dos maneras, la primera de ellas, al proponer al usuario los servicios que pueden coincider con un comando de texto, conforme se encuentra ingresando los caracteres que lo conforman; la segunda, al realizar el análisis de conectividad del dispositivo a la red celular e inalámbrica, con el propósito de determinar si es posible activar el servicio en el momento que el usuario lo desea.

Los resultados obtenidos muestran que el sistema facilita el uso del dispositivo móvil y el acceso a los servicios que proporciona, permitiendo al usuario seleccionar la modalidad con la que desea interactuar con el dispositivo de acuerdo a sus preferencias y las condiciones del ambiente de uso.

Abstract

In the last years mobile devices are becoming a fundamental communication and work tool for many people. The number of users of the so-called smartphones and the capabilities of such devices are growing everyday. This fact arises several interesting challenges in the mobile computing (MC) and the human-computer interaction (HCI) research areas. In MC the main challenge is the design of devices with improved computing capacities and the optimization on the use of resources. In HCI the main challenge is the desing of multimodal interfaces for mobile devices.

In this work, first it is shown the necessity of counting with a mechanism for the multimodal and proactive activation of services, the aim is to supply a more natural interaction with the user, even if he has some kind of phycisical handicap, exploiting at the same time the computing capacities of the mobile device through the use of the sensors installed in this kind of devices.

To deal to this necessity, this work proposes a system for the multimodal activation of services on Android-based devices. This proposal allows the user to define commands in several modalities (audio, physical gestures, text and tactile gestures) and associate them to some application or contact available on the device. The user can use the commands so defined in two ways: the first in an isolated way for activating a service or show a contact, in the second way it is possible to combine two commands for calling or sending a message to some contact on the device.

Moreover the proposal incorporates the quality of proactivity in two ways: the first is by proposing to the user the services that match up the command being typed by the user, the second is through the analysis of the connectivity (cellular or wi-fi) in order to determine if the service requested by the user can be activated or not.

The results show that this proposal makes easier the use of the device and its services, allowing to select the modality of activation in function of user's preferences or environment conditions.

Agradecimiento

Gracias a DIOS por permitirme culminar una etapa más en mi vida y ser mi guía invisible en todo momento.

A mis padres, Ernesto Ponce y Ma. Guadalupe Angulo, por impulsarme a emprender esta etapa aún cuando significó salir de casa. Gracias por su apoyo incondicional y su amor infinito.

A mis queridos hermanos, Fabiola, Ma. Guadalupe, Ramón Ernesto, Marisol y Ernesto por sus muestras de cariño y estar siempre al pendiente de mi.

A mi asesor, el Dr. José Guadalupe Rodríguez García por la confianza brindada y por la oportunidad de llevar a cabo este proyecto de tesis. Gracias por sus enseñanzas, orientación, observaciones y el tiempo que dedicó a este trabajo.

A mis revisores, el Dr. Amilcar Meneses Viveros y el Dr. Jorge Buenabad Chavez por el tiempo que dedicaron a la revisión de este documento y por sus valiosos comentarios.

A mis amigos, gracias por su apoyo a lo largo de este tiempo, por sus palabras de ánimo y brindarme momentos de esparcimiento. Especialmente gracias a Alexis de la Cruz por ser un excelente amigo, persona y equipo de trabajo.

A una gran persona, Sofia Reza por su amabilidad en todo momento y sus consejos.

A mi amado esposo, Jesús Eduardo Urías por ser la persona que me ha acompañado cada día de esta etapa, gracias por motivarme y convertir cada momento en algo especial.

A CONACYT, por la beca otorgada a lo largo de mis estudios de maestría y por fomentar la formación de nuevos investigadores para México.

Al Centro de Investigación y de Estudios Avanzados del IPN (CINVESTAV-IPN), por permitirme ser parte de esta reconocida institución y por el apoyo económico otorgado en la finalización de mis estudios. Agradezco de forma especial a los Doctores del Departamento de Computación que me compartieron parte de sus conocimientos y experiencias, contribuyendo así a mi formación profesional.

Dedicatoria

Esta tesis la dedico especialmente a mis padres, hermanos y esposo, a quienes agradezco de todo corazón su amor, cariño, comprensión y apoyo incondicional.

Ustedes son el principal pilar en mi vida, quienes me motivan a ser mejor persona cada día.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Planteamiento del problema

El ser humano siempre busca interactuar con los dispositivos electrónicos de la manera más natural posible, tal como lo hace con otras personas, utilizando sus diferentes capacidades motrices, de voz y audiovisuales. Es por esta razón que han surgido diversas áreas de investigación como la de interfaces hombre computadora (*HCI* por sus siglas en inglés) y que se han logrado grandes avances tecnológicos en la creación de dispositivos que faciliten esta tarea.

El diseño de interfaces de usuario para dispositivos móviles proponen nuevos retos, debido a la gran variedad de dispositivos y de sus características. Este reto se vuelve mucho mayor cuando los usuarios son personas con capacidades diferentes, ya sea por algún impedimento físico o por algún problema degenerativo de su capacidad motriz.

Por otra parte, cada vez los dispositivos móviles cuentan con una mayor capacidad de cómputo y conectividad, de la misma forma, el número de sensores que se encuentran embebidos a éstos va en aumento proporcionándoles una mayor funcionalidad. Si se compara un dispositivo móvil actual con una computadora personal de hace algunos años, la capacidad de cómputo del primero supera a la segunda. Sin embargo no se ha explotado completamente dicha capacidad en los dispositivos móviles, se siguen utilizando para realizar tareas simples que requieren bajo poder de procesamiento, como son realizar una llamada, enviar mensajes SMS o visitar alguna página en Internet.

Existe un elevado número de aplicaciones disponibles para este tipo de dispositivos, tomando en cuenta las que se encuentran incorporadas por omisión en el mismo y las disponibles en el mercado, con la característica de que la mayoría de estas aplicaciones integran una o dos formas de interacción con el usuario (entrada por texto, voz, *touch*, etc).

Con todo este panorama, se plantea el problema de contar con interfaces de usuario proactivas que asistan al usuario proporcionando opciones de servicios y que incorporen más de una modalidad de entrada/salida a la vez, y que proporcionen una interacción más natural con el usuario, aún cuando éste tenga algún tipo de impedimento y al mismo tiempo, explote las capacidades de cómputo a través de los sensores disponibles en este tipo de dispositivos.

1.2 Objetivos

1.2.1 Objetivo general

El objetivo principal del presente trabajo de tesis es diseñar e implementar mecanismos de activación multimodal de servicios, haciendo uso de los diversos sensores e interfaces embebidos en los dispositivos móviles.

1.2.2 Objetivos específicos

Para lograr el objetivo principal se establecieron los siguientes objetivos particulares:

- Diseñar mecanismos de interacción-activación multimodal de servicios.
- Facilitar la interacción con el usuario, brindando mecanismos complementarios de activación de servicios.
- Proponer una solución proactiva, que brinde opciones para la activación de servicios que mejor se puedan ajustar al contexto del usuario.
- Implementar mecanismos de análisis de contexto que asistan al sistema.
- Implementar mecanismos proactivos que brinden al usuario opciones de comunicación en función del tipo de redes disponibles.

1.3 Metodología

Para el logro de los objetivos planteados en la sección anterior, se realizaron un conjunto de tareas definidas para el desarrollo de proyectos de software, ya que como resultado final se espera obtener un conjunto de componentes que se ejecuten en dispositivos móviles, que permitan la activación multimodal de los servicios.

La metodología a seguir se divide en diferentes etapas y éstas, a su vez, en una serie de actividades, a continuación se presenta una descripción general de cada etapa:

- Documentación y análisis: Esta etapa tiene como objetivo obtener una visión general de los temas relacionados al proyecto de investigación para poder establecer y delimitar su alcance. Para ello se requiere una revisión exhaustiva de la bibliografía existente y el estado del arte relacionado.
- Propuesta de solución: Durante esta etapa se espera lograr la abstracción de la solución que permitirá lograr los objetivos establecidos para el proyecto. Requiere la revisión y análisis de los diferentes algoritmos que pueden ser útiles para la implementación de los objetivos y así poder definir un diseño arquitectural que modele los diferentes componentes de software que ofrezcan una solución inicial.
- Desarrollo: El objetivo principal de esta etapa es la implementación en software de la propuesta de solución, de cada uno de los componentes modelados. Para ello se requiere realizar primero un estudio de las herramientas disponibles que nos permitan llevar a cabo el proceso de implementación de la forma más adecuada. Cada componente debe ser codificado e integrado. Adicionalmente, en esta etapa se debe desarrollar una aplicación de servicios de telefonía para dispositivos móviles que haga uso de los componentes implementados.
- Pruebas: En esta etapa se evalúa el funcionamiento de cada uno de los componentes software desarrollados. Se requiere al final realizar diferentes pruebas de funcionalidad, rendimiento y usabilidad global de la propuesta.
- Resultados: Durante esta etapa se requiere analizar los resultados obtenidos en el proceso de pruebas y realizar una evaluación cuantitativa de los mismos para determinar si los objetivos establecidos inicialmente fueron logrados.

1.4 Organización del documento

La tesis que se presenta en este documento está estructurada en seis capítulos que se describen a continuación:

- **Capítulo 2. Estado del arte y trabajo relacionado:** este capítulo presenta el panorama actual de las áreas de investigación relacionadas a este proyecto de tesis, además de una descripción de los principales proyectos y aplicaciones actuales en el campo de interfaces multimodales.
- **Capítulo 3. Propuesta de solución:** presenta una descripción detallada de las etapas de análisis y diseño del sistema multimodal. En este capítulo se plantea la arquitectura física y lógica de la aplicación, así como los requisitos funcionales que debe cumplir.

- **Capítulo 4. Desarrollo del proyecto:** describe la forma en que se lleva a cabo la implementación del sistema multimodal utilizando la plataforma Android para dispositivos móviles.
- **Capítulo 5. Pruebas y resultados:** en este capítulo se detallan diversas pruebas que permiten verificar el correcto funcionamiento de la aplicación implementada.
- **Capítulo 6. Conclusiones y trabajo a futuro:** el objetivo de este capítulo es presentar una valoración cualitativa del trabajo realizado y plantear actividades que se podrían realizar para mejorar los resultados obtenidos.

Capítulo 2

Estado del arte y trabajos relacionados

Hoy en día, los dispositivos móviles forman parte de la vida diaria de las personas, de manera frecuente se introducen en el mercado nuevos dispositivos de este tipo con mejores características y funcionalidades.

Esta realidad es posible gracias a los avances tecnológicos en las áreas de cómputo móvil, comunicaciones inalámbricas e interfaces humano computadora.

El presente proyecto involucra estas áreas de investigación, es por ello que el contenido de este capítulo está conformado por diferentes secciones, que presentan el panorama actual de las áreas de investigación mencionadas y de los dispositivos móviles *smartphones* y sus características.

En la última sección se presentan los diferentes proyectos y aplicaciones relacionadas al tema de interfaces móviles multimodales.

2.1 Cómputo móvil

Los avances tecnológicos que ha vivido la humanidad en las últimas décadas han venido a revolucionar la forma en que se llevan a cabo las diferentes actividades diarias de los individuos, la manera en que éstos se relacionan y además, han propiciado el surgimiento de nuevas áreas de investigación.

El cómputo móvil es una área que surge gracias a las contribuciones que se han hecho en las tecnologías de comunicación inalámbricas y el área de los sistemas distribuidos. Esta área ha tenido gran auge debido a la necesidad de tener acceso a la información "en cualquier momento y en cualquier lugar, desde cualquier dispositivo"[1] .

El área de cómputo móvil hace referencia a un amplio conjunto de operaciones que permiten al usuario acceder a la información desde dispositivos portátiles, tales como

laptops, PDAs, teléfonos celulares, computadoras de bolsillo, reproductores de música, dispositivos de juegos portátiles, entre otros.

El uso de las tecnologías inalámbricas permiten que los usuarios puedan disfrutar de la movilidad de manera tal que su ubicación y la del dispositivo móvil no tenga que estar fija para mantener una conexión a la red; de esta manera el usuario puede realizar cualquier tarea mientras se encuentra en movimiento [1].

La investigación y desarrollo del cómputo móvil está en una etapa crucial. Muchos temas difíciles en este ámbito aún no se han abordado y se plantean líneas de investigación de vanguardia y el desarrollo en varias direcciones, principalmente en:

- Integración de redes inalámbricas y cableadas
- Seguridad y privacidad móvil
- Cómputo móvil conciente de la ubicación
- Interfaces humano computadora para aplicaciones móviles
- Diseño de software conciente del contexto
- Cómputo móvil con bajo consumo de energía

Evolución de las tecnologías de telecomunicación móvil

Los sistemas de telecomunicación móvil se pueden clasificar en generaciones, que si bien han aparecido en un corto espacio de tiempo haciendo que coexistan en la práctica unas con otras, éstas están basadas en diferentes tecnologías y van dirigidas a diferentes tipos de aplicaciones. A continuación se presenta un panorama general de la evolución de estos sistemas.

Durante los años 60's los laboratorios Bell desarrollaron el concepto de tecnología celular. Un gran avance tuvo lugar con la introducción del mecanismo de reutilización de frecuencia en el funcionamiento celular. Simultáneamente, los avances en la tecnología de los circuitos integrados, el procesamiento de señales y la tecnología de baterías, ayudaron al crecimiento de los servicios de comunicación personal y móvil. Tales sistemas fueron primeramente analógicos, mejor conocidos como sistemas móviles de primera generación (1G).

En 1990, la primera especificación de celular digital se publicó para el sistema global de comunicaciones móviles (GSM). Los sistemas que utilizaron la tecnología celular digital se consideran como la segunda generación de los sistemas móviles (2G) o sistemas de servicios de comunicación personal (PCS). El enfoque principal de los sistemas 2G fue la conmutación de circuitos de voz y los servicios de datos con una baja tasa de bits.

A lo largo de los años, la demanda de tener tasas de datos mayores se incrementó debido a las nuevas aplicaciones que requerían acceso a Internet y por los nuevos estándares como el General Packet Radio Service (GPRS) para GSM, Cellular Digital Packet Data (CDPC), High Speed Circuit-Switched Data (HSCSD), etc. Esto se considera el inicio de la siguiente generación de acceso inalámbrico, conocido como estándares de la generación 2.5 (2.5G).

A pesar de que el primer fin para el que se concibieron las redes inalámbricas fue para la transmisión de voz, con la revolución que trajo consigo el surgimiento de Internet, se prestó mayor atención a los servicios que demandaban su uso. Dos estándares, Mobile IP y Wireless Application Protocol (WAP), se propusieron para proveer soporte a nivel de aplicación para las redes inalámbricas.

Para enfrentar los desafíos de los sistemas 2G, la Unión Internacional de Telecomunicaciones (ITU) desarrolló el estándar IMT-2000 e inició el desarrollo de la tercera generación de los sistemas móviles (3G). El Universal Mobile Telecommunication System (UMTS) propuesto por el Instituto Europeo de Estándares en Telecomunicaciones (ETSI), se desarrolló a la par con el IMT-2000, soportando el *roamming* global y facilitando los servicios de datos a alta velocidad para las diferentes clases de QoS que van desde simples datos (ej. correo electrónico) hasta los datos más esenciales de conversación (ej. voz) o incluso los servicios multimedia altamente priorizados (ej. video bajo demanda).

Las organizaciones internacionales de regularización y estandarización de las telecomunicaciones están trabajando para que los sistemas móviles de cuarta generación (4G) estén disponibles de manera comercial alrededor de los años 2012-2015. Los sistemas 4G se proyectan con mayor tasa de transferencia de datos. Estos sistemas estarán disponibles para todas las redes IP y se proporcionará a los usuarios servicios de voz, datos y *streaming* de multimedia en cualquier momento y lugar [2, 1].

2.2 Dispositivos móviles: teléfono inteligente (*smartphone*)

Los dispositivos móviles están convirtiéndose en una parte importante en la vida diaria de las personas, actuando como una herramienta de comunicación, trabajo y entretenimiento [3].

Si bien, los primeros teléfonos móviles fueron diseñados con el objetivo principal de realizar llamadas por voz, los avances tecnológicos han permitido reducir la diferencia existente entre lo que hoy se considera un teléfono celular convencional y una computadora. Conforme esta diferencia tecnológica se reduce y se introducen nuevas generaciones de dispositivos móviles al mercado, un nuevo paradigma está emergiendo: la gente está reemplazando sus computadoras personales por los teléfonos inteligentes (*smartphones*) [4]. Esto acerca cada día más a la realidad la visión compartida por un gran número

de personas en la industria de la computación:

> "Un día, la computadora principal, el dispositivo que utilizas para realizar la mayoría de las tareas, será móvil, facíl de llevar en el bosillo y estará completamente integrado con el teléfono móvil" [5].

El término teléfono inteligente (*smartphone*) fue acuñado inicialmente por estrategas de mercado, para referirse en ese entonces a una nueva clase de teléfonos celulares con poder significativo de cómputo, que podían facilitar el acceso y procesamiento de datos. Estos dispositivos, surgen debido a la necesidad de tener acceso a datos en el teléfono celular [1].

Un *smartphone* normalmente soporta tecnologías inalámbricas además de las conexiones celulares de datos. Provee capacidad de cómputo móvil, acceso ubicuo a datos e inteligencia pervasiva para casi todos los aspectos de proceso de negocios y de la vida cotidiana de las personas. La movilidad y las características que ofrecen los *smartphones* permiten a los usuarios interacturar de forma más directa y continua con estos dispositivos. Los *smartphones* representan el primer dispositivo de computación móvil verdaderamente ubicuo [4, 6].

Además de las tradicionales aplicaciones de un teléfono celular (directorio, organizador, calculadora, calendario), los *smartphones* incluyen otro tipo de aplicaciones como juegos, cámara, reproducción y grabación de audio/video, mensajería instantanea, correo electrónico, buscadores de Internet, entre muchas otras que se encuentran disponibles en los mercados de aplicaciones de cada proveedor [7].

Otra característica que distingue a los teléfonos inteligentes actuales es que están equipados con diferentes tipos de sensores (GPS, micrófono, cámara, acelerómetros, giroscopios, sensores de proximidad, sensores de luz, entre otros) y mejorados por sistemas operativos móviles, como Android OS, iOS, Windows Mobile, Symbian OS y Blackberry [8].

Cada día aumenta el número de usuario de este tipo de dispositivos y se espera que esta cifra se incremente considerablemente en los próximos años. Así como el número de usuarios, aumentan también las características y capacidades de estos dispositivos para volverse más avanzados.

De acuerdo con diversos estudios realizados por la *International Data Corporation (IDC) Worldwide Quarterly Mobile Phone Tracker*[1], el mercado de ventas de dispositivos móviles y en especial de smartphones, presentan las siguientes estadísticas:

[1] *www.idc.com*

- Sólo en el año pasado 2011, el mercado de teléfonos inteligentes creció un 61.3 % a nivel mundial, con respecto al año anterior, al alcanzar 491.4 millones de unidades en comparación de las 304.7 millones de unidades del 2010 [9].

- Con respecto a lo que va del año 2012, reporta que en el primer trimestre, el mercado de teléfonos móviles a nivel mundial se redujó 1.5 % al vender 398.4 millones de unidades con respecto a 404.3 millones de unidades en este mismo periodo del año 2011; por otra parte, en ese mismo período de tiempo el mercado mundial de telefonos *smartphones* creció 42.5 % al alcanzar 144.9 millones de ventas en estos dispositivos, en comparación a 101.7 millones de unidades ese mismo periodo del año 2011 [10].

- En el segundo trimestre del presente año, el mercado mundial de teléfonos móviles creció 1 %, al alcanzar los 406.0 millones de unidades, con respecto a ese mismo periodo del año 2011 que llegó a los 401.8 millones de unidades. El mercado mundial de *smartphones* creció 42.1 durante el segundo trimestre de este año llegar a 153.9 millones de unidades, comparado con 108.3 millones de unidades en este periodo del 2011. Sin embargo, esta cifra se encuentra un punto porcentual por debajo del pronostico de *IDC* [11].

La *IDC* espera que aún cuando el espectro de nuevos problemas económicos pone en riesgo las perspectivas de crecimiento para el mercado de telefonía móvil, la demanda a largo plazo de teléfonos móviles y smartphones crezca de manera sostenida en el 2012 y en los años venideros, debido al papel que juegan estos dispositivos en la vida de las personas, tal como lo expresa Ramón Llamas, analista de esta corporación:

> "Para muchos usuarios, el teléfono móvil se ha convertido en el eslabon esencial de comunicación con los demás y con el mundo" [11].

2.2.1 Sistemas operativos y herramientas de desarrollo

Los teléfonos inteligentes se definen principalmente por el sistema operativo que utilizan [12]. Existen en la actualidad varias opciones de sistemas operativos y herramientas disponibles para el desarrollo de aplicaciones para dispositivos móviles [13]. A continuación se presentan algunos de ellos y sus principales características.

Android

Android[2] es el sistema operativo de Google, presentado en el año 2007 para promover estándares abiertos para dispositivos móviles. Android es una plataforma de software

[2] http://www.android.com

libre *Apache*, basada en Linux. La pila de software que lo conforma incluye un sistema operativo, un *middleware* y las aplicaciones principales.

- Las aplicaciones Android son escritas en el lenguaje de programación Java y compiladas en la máquina virtual denominada *Dalvik*.
- Cada aplicación es ejecutada en su propio proceso, con una instancia propia de la máquina virtual *Dalvik*.
- Dalvik ejecuta archivos DEX (formato ejecutable), que en tiempo de compilación son convertidos de clases estándares a archivos JAR.
- Los desarrolladores tienen acceso completo a todos los *frameworks* y APIs que utilizan las aplicaciones Android y a las bibliotecas de software desarrolladas por Google.
- La arquitectura de software de Android está diseñada para facilitar el reuso.
- El kit de desarrollo (SDK) de Android permite el desarrollo de aplicaciones con una gran variedad de funcionalidades.
- Puede manejar pantallas táctiles, acelerómetros, gráficos 3D, GPS, así como colaboración entre aplicaciones como correo electrónico, mensajería, calendarios, redes sociales, servicios basados en localización, por mencionar algunos.

Apple iOS

Es un sistema operativo propietario de Apple[3] que se ejecuta en sus dispositivos móviles *(iPhone, iPod touch e iPad*).

- Este sistema operativo maneja los componentes hardware del dispositivo y provee las tecnologías necesarias para implementar aplicaciones nativas.
- Provee su propio ambiente de desarrollo (*iOS SDK*), que contiene las herramientas e interfaces necesarias para desarrollar, instalar, ejecutar y probar aplicaciones nativas.
- Utiliza el lenguaje de programación *Objective-C* que es ejecutado directamente en *iOS*.
- La arquitectura de *iOS* es similar a la arquitectura básica de *Mac OS X*. A un nivel mayor de abstracción, *iOS* actúa como intermediario entre el hardware y la aplicación que se muestra en la pantalla del dispositivo.

[3] http://www.apple.com/ios/

Microsoft .NET Compact Framework (CF)

Diseñado para aplicaciones *Windows Mobile*[4], *.NET CF* es un subconjunto de la plataforma *.NET* de Microsoft. . *NET CF* carga previamente el motor del lenguaje común de ejecución (*CLR*) en la memoria del dispositivo para facilitar el despliegue de la aplicación móvil.

- El ambiente de ejecución de *.NET CF* es análogo a la Java Virtual Machine (JVM).
- La herramienta de desarrollo de *.NET CF* (*VS.NET*), sólo soporta dos principales lenguajes *.NET: C#* y *Visual Basic* (*VB.NET*). Por lo tanto, las aplicaciones están restringidas a plataformas *Windows*.
- Los componentes principales son un subconjunto del *framework .NET*, aproximadamente el 30 % de las clases y funcionalidad.
- El diseño de interfaces de *.NET CF* esta basado en un subconjunto de *.NET Windows Forms*.

Windows Phone

Windows Phone[5] es el sistema operativo móvil desarrollado por Microsoft como sucesor de Windows Mobile. En el 2010 se presentó oficialmente este producto y el año pasado se anunció la asociación entre las compañias Microsoft y Nokia, la cual consiste en que Windows Phone será el principal sistema operativo para los dispositivo móviles de esta última compañía.

El desarrollo de aplicaciones en Windows Phone se basan en dos modelos de programación:

- *Microsoft Silverlight*, que utiliza *XML Aplication Markup Language (XAML)*. Incluye el *Microsoft .NET Compact Framework*, que hereda de la arquitectura *.NET Framework.*

  La interfaz gráfica para *Silverlight* puede ser creada tanto en *Visual Studio* y *Expression Blend 4* para Windows Phone.

- *Microsoft XNA Framework*, en este modelo las aplicaciones son desarrolladas completamente en código *C#* o *VB.NET* utilizando *Visual Studio*. Es una implementación nativa de *.NET Compact Framework* que incluye un amplio conjunto de bibliotecas de clases, específicas para el desarrollo de juegos.

  Además permite desarrollar juegos para Xbox 360, Zune y Windows 7.

[4]http://msdn.microsoft.com/en-us/windowsmobile/
[5]https://dev.windowsphone.com

Tanto *Silverlight* y *XNA Framework* comparten el mismo modelo de aplicación construido sobre el *.NET Common Language Runtime (CLR)*, permitiendo el acceso a las bibliotecas de todos los frameworks. Por lo tanto, una aplicación de *Silverlight* puede acceder a las *APIs* de audio dipsonibles en el *XNA Framework*. Del mismo modo, una aplicación *XNA Framework* puede acceder al servidor de acceso remoto de *APIs* disponible en *Silverlight* [14] .

Symbian

Es un sistema operativo que tiene sus orígenes en un acuerdo de colaboración de diferentes empresas manufactureras de dispositivos móviles, entre ellas Nokia, Ericsson, Panasonic y Samsung, para crear un sistema operativo en común [15, 16].

- Es un sistema operativo diseñado específicamente para dispositivos móviles.
- Maneja un diseño orientado a objetos.
- El kernel de este sistema operativo tiene un diseño microkernel. Los sistemas operativos basados en microkernel normalmente ocupan mucho menos memoria durante el arranque y su estructura es más dinámica
- Utiliza el modelo cliente/servidor para acceder a los recursos del sistema. Las aplicaciones que necesitan acceder a recursos del sistema son los clientes; los servidores son programas que el sistema operativo ejecuta para coordinar el acceso a dichos recursos.
- Symbian presenta la desventaja de proveer un entorno de desarrollo de aplicaciones difícil de manejar.

Java Micro Edition (ME)

Java Micro Edition (ME)[6], es un subconjunto de la plataforma *Java* que provee una colección certificada de interfaces de programación de aplicaciones o *APIs* (del inglés *Application Programming Interface*) para el desarrollo de software para dispositivos con recursos restringidos como teléfonos celulares y *PDAs*.

- *Java ME* se ejecuta en el tope de una máquina virtual basada en kernel (*KVM)*, que permite un acceso razonable pero no completo a la funcionalidad del dispositivo.
- Soporta el desarrollo multi-plataforma a través de configuraciones y perfiles.

[6]http://www.oracle.com/technetwork/java/javame

- Una configuración define la características mínimas de la *Java VM* y el conjunto de bibliotecas para una familia horizontal de dispositivos.
- Un perfil comprende bibliotecas especializadas en características únicas de una clase en particular de dispositivos.

De acuerdo al análisis de las ventas de *smartphones* realizado por la *International Data Corporation (IDC) Worldwide Quarterly Mobile Phone Tracker* en el segundo trimestre del 2012 [17], los sistemas operativos Android e iOS establecieron un nuevo récord combinado al estar presentes en el 85 % de todos los *smartphones* vendidos en ese trimestre. Mientras tanto, BlackBerry y Symbian, los dos pioneros y líderes anteriores del mercado de teléfonos inteligentes, vieron caer sus ventas por debajo del cinco por ciento. De acuerdo con la *IDC*, Android domina el mercado con el 68.1 % de las ventas totales en *smartphones*, seguido de iOS con el 16.9 %, mientras que BlackBerry y Symbian llegaron al 4.8 % y 4.4 % respectivamente.

La figura 2.1 muestra la gráfica representativa de estos datos.

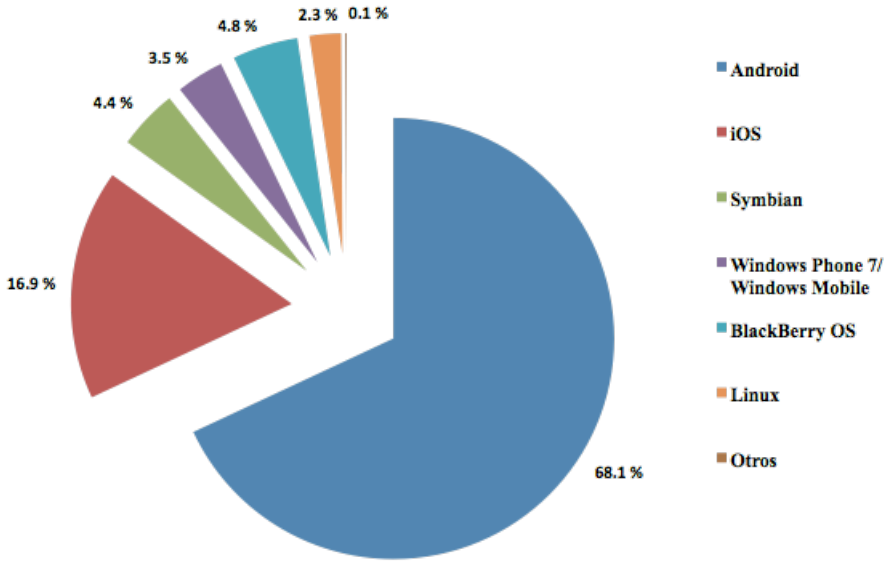


Figura 2.1: Sistemas operativos en smartphones

2.2.2 Capacidad de cómputo y memoria

Los aspectos de capacidad de cómputo, tomando como referencia la frecuencia de reloj del procesador y la memoria de almacenamiento disponible en los dispositivos móviles, también se ha incrementado en los últimos años.

De acuerdo a un estudio presentado por Xun Li et. al. [18], donde analiza diferentes características de las generaciones de los dispositivos *smartphones HTC* y *Apple*, entre ellas la capacidad de procesador y memoria, el incremento de estas capacidades siguen la ley de Moore [19] y se espera que los nuevos *smartphones* cuenten con capacidades de cómputo y memoria equivalentes o comparables entre fabricantes, a no ser que el consumo de energía que esta situación implique llegue a un nivel en que se tenga que realizar una compensación entre estas características y el consumo de energía.

La tabla 2.1 presenta las generaciones de *smartphones* analizados por Xun Li y las figuras 2.2 y 2.3 muestran gráficamente la evolución de las capacidades de cómputo y memoria de estos dispositivos.

Tabla 2.1: Lista de las diferentes generaciones de smartphones del caso de estudio presentado por Xun Li et. al.

Marca		Fecha de lanzamiento	Frecuencia de reloj del procesador	Memoria RAM
HTC	HTC Dream	Octubre 2008	528 MHz	192 MB
	HTC Magic	Marzo 2009	528 MHz	192 MB
	HTC Hero	Octubre 2009	528 MHz	288 MB
	HTC Nexus One	Enero 2010	1 GHz	512 MB
	HTC Desire	Marzo 2010	1 GHz	576 MB
	Droid Incredible	Abril 2010	1 GHz	512 MB
	HTC Evo 4G	Junio 2010	1 GHz	512 MB
Apple	iPhone Original	Junio 2007	412 MHz	128 MB
	iPhone 3G	Julio 2008	412 MHz	128 MB
	iPhone 3GS	Junio 2009	600 MHz	256 MB
	iPhone 4G	Junio 2010	1 GHz	512 MB

En cuanto a la frecuencia de reloj que emplean los *smartphones* analizados, es posible observar en la figura 2.2 que *Apple* lanza sus primeros modelos de *iPhone* con un procesador de 412 MHz, en el año 2009 utiliza uno de 600 MHz en el *iPhone 3GS* y el modelo *4G* de esta misma gama utiliza un procesador de 1 GHz. Por otra parte, los productos de *HTC*, catalogados como teléfonos inteligentes, utilizan procesadores con una frecuencia de reloj de 528 MHz durante los años 2008-2009 y en el año 2010 estos dispositivos incorporan procesadores de 1 GHz.

Con respecto a la cantidad de memoria RAM que incorporan estos mismos dispositivos, se puede apreciar en la figura 2.3 que practicamente se ha duplicado de un año a otro. El dispositivo *HTC Dream* utiliza una memoria RAM de 192 MB en el año 2008; para el año 2009 en el *HTC Hero* incorpora una memoria de 288 MB y en la mayoria de los dispositivos del año 2010 utiliza memorias RAM de 512 MB, con excepción del *HTC Desire*

en el incorpora una de 576 MB. El *iPhone Original* de *Apple* se presenta en el 2007 con una memoria RAM de 128 MB, al igual su sucesor el *iPhone 3G* en el 2008, en el año 2009 el *iPhone 3GS* incorpora una memoria con el doble de capacidad con 256 MB y el *iPhone 4G* en el año 2010 vuelve a doblar el tamaño de la memoria RAM con respecto a su sucesor al incorporar 512 MB.

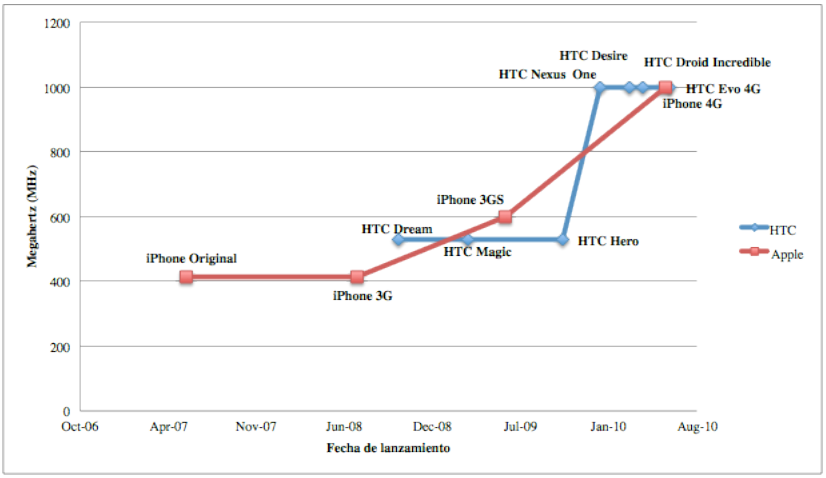


Figura 2.2: Gráfica de la evolución en la capacidad de cómputo en las diferentes generaciones de HTC y Apple

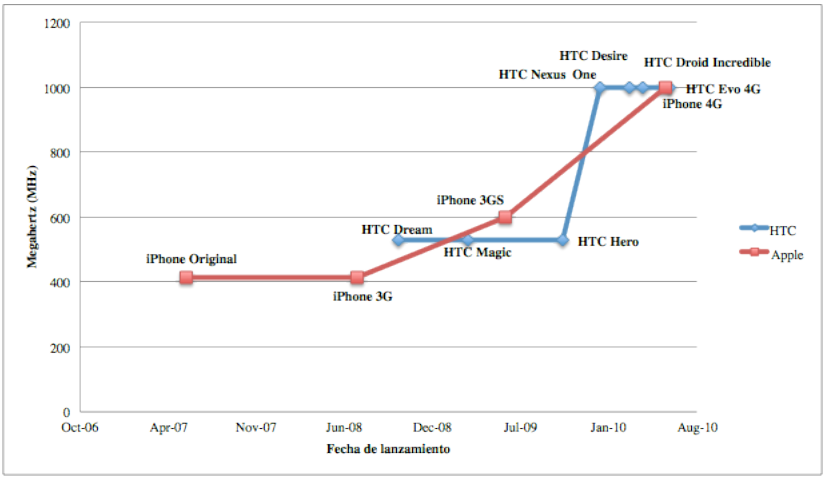


Figura 2.3: Gráfica de la evolución en la capacidad de memoria RAM en las diferentes generaciones de HTC y Apple

2.2.3 Sensores

Hoy en día los teléfonos inteligentes integran en su estructura un conjunto de sensores que recaban diferentes tipos de información del contexto en el que se encuentra el dispositivo. A continuación se listan los principales sensores emebebidos en los *smartphones* actuales [3, 20].

Global Positioning System (GPS)

Permite al dispositivo autolocalizarse, apoya a las aplicaciones basadas en localización tales como búsqueda local, redes sociales y navegación.

El GPS fue pensado originalmente para aplicaciones militares, pero en la década de 1980, el gobierno hizó el sistema diponible para uso civil. Es un sistema que sigue la navegación del usuario, generalmente con una imagen de un mapa de fondo, mostrando donde ha estado. Permite la programación de rutas, proporcionando una línea que representa la ruta y que el usuario puede seguir en la pantalla del teléfono inteligente.

Los satélites GPS rodean la Tierra dos veces al día en una órbita muy precisa y transmiten señales de información a la Tierra. Los receptores GPS toman esta información y utilizan la triangulación para calcular la posición exacta del usuario.

Acelerómetro

El acelerómetro permite que los dispositivos *smartphones* detecten la orientación del dispositivo. Se aplica en la estabilización de imágenes y video, control de gestos, orientación de la interfaz gráfica de usuario, caracterización de movimientos del usuario del dispositivo, entre otras.

Por ejemplo, cuando se gira el dispositivo hacia un lado, éste automáticamente cambia la pantalla al modo horizontal, para que el usuario tenga un espacio de visión más amplio. Del mismo modo, la cámara se basa en el acelerómetro para saber si el usuario está tomando una fotográfia en modo horizontal o vertical.

El acelerómetro en los *smartphones* mide la aceleración del dispositivo en relación con la caída libre. Un valor de 1 indica que el dispositivo tiene 1g de fuerza ejercida sobre él (1g de fuerza es la atracción gravitacional de la tierra, que el dispositvo experimenta cuando está en estado estacionario). el acelerómetro mide la aceleración del dispositivo en tres ejes: x, y, z.

Brújulas

Las brújulas son atraídas a los polos terrestres usando imanes. Sin embargo los teléfonos inteligentes modernos no pueden utilizarlos debido a la interferencia magnética haría

que las capacidades de telefonía móvil del dispositivo inutilizables.

Un ejemplo, el iPhone 3GS utiliza el chip AK8973, el cual es un pequeño sensor que está a la "escucha" de una señal de ultra baja frecuencia . Si dicha señal proviene de un lugar específico como el norte, se combina con el acelerómetro para que el dispositivo pueda calcular la orientación y la dirección. La brújula en el iPhone 4 es la AK8975, que es muy similar a la descrita anteriormente. Esta brújula detecta la orientación relativa al campo magnético de la Tierra utilizando el *efecto Hall*. El *efecto Hall* se produce cuando un campo magnético se aplica transversalmente a un flujo de corriente. El campo magnético desvía las cargas en movimiento que conforman la corriente, induciendo una tensión (tensión de Hall) que es transversal a la corriente. La tensión de Hall puede ser medida y utilizada para determinar la intensidad del componente del campo magnético que era transversal a esta corriente.

Mediante el uso de múltiples sensores orientados en diferentes direcciones y el uso de un disco de material de alta permeabilidad, denominado concentrador magnético, para doblar las líneas del campo magnético que son paralelas al plano del sensor, de manera que tengan una componente perpendicular al plano del sensor que se puedan detectar, el dispositivo puede medir el vector del campo magnético total y por lo tanto, determinar la orientación del dispositivo relativa a dicho campo magnético.

Giroscopio

Un giroscopio es un dispositivo para medir o mantener la orientación, basado en los principios de momento angular. Los *smartphones* y *tablets* utilizan los sensores giroscópicos en sistemas de navegación y sistemas de reconocimiento de gestos para encontrar la posición y orientación del dispositivo.

La combinación de un giroscopio con un acelerómetro permite que el dispositivo detecte el movimiento en seis ejes: izquierda, derecha, arriba, abajo, adelante y atrás, así como las rotaciones *yaw*, *roll* y *pitch*.

Cámara

Puede ser utilizada para realizar varias cosas, incluyendo desde la tradicional tarea de tomar una foto hasta realizar un seguimiento de los movimientos del ojo del usuario para activar alguna aplicación.

Sensor de luz ambiental

El uso de un sensor de luz ambiental en dispositivos móviles como *smartphones*, *tablets* y *laptops*, extieden el tiempo de vida de la batería y permite que las pantallas sean

fáciles de ver al estar optimizadas para el medio ambiente. Los sensores de luz se utilizan normalmente para ajustar el brillo de la pantalla.

En lugar de utiliar el principio de superposición para calcular el brillo de la luz ambiental, los sensores más actuales en el mercado, utilizan dos o más tipos diferentes de fotodiodos, cada uno sensible a una porción diferente del espectro de luz. Mediante la combinación matemática de las salidas de emiten estos fotodiodos, cada uno con una ganacia ajustada adecuadamente, el sensor puede dar una salida con una medición bastante exacta de la luminosidad del entorno para las fuentes de luz disponibles comúnmente. Sin embargo, básicamente, un sensor de luz ajusta el brillo de la pantalla, que a su vez ahorra energía de la batería del *smartphone*; este ahorro de energía se da por el ajuste del brillo de la pantalla con base a la cantidad de luz ambiental que está presente.

Sensor de proximidad

Un sensor de proximidad es muy útil en un *smartphone*. Detecta la proximidad de la pantalla del teléfono al cuerpo. Esto permite que el dispositivo detecte, por ejemplo, cuando el usuario acerca el teléfono al oído para realizar una llamada. En este caso se desactivan la pantalla táctil y el teclado, previniendo así que sean presionados accidentalmente; de esta forma también se ahorra energía ya que la pantalla es apagada.

En resumen, el sensor de proximidad en un *smartphone* detecta qué tan cerca está el teléfono a la mejilla o a la cara del usuario, de forma que el dispositivo pueda detener cualquier actividad que está ejecutando (ej. la reproducción de música o navegar por la Web) para que el usuario pueda realizar o tomar una llamada telefónica. Cuando el teléfono se aleja de la oreja después de la llamada, el teléfono vuelve a su actividad anterior.

Micrófono

Obtener información continua de este sensor permite, por ejemplo, clasificar los diferentes sonidos distintivos asociados a contextos particulares de las actividades de las personas. Ademas es un sensor indispensable en un teléfono ya que permite la captura de audio.

2.3 Interfaces Multimodales Humano - Computadora (HCI)

Como se mencionó en la sección 2.1, los avances en cómputo móvil imponen grandes retos en otra área de investigación denominada interfaces hombre computadora (HCIs), esto debido a que las aplicaciones y los servicios son utilizados principalmente cuando el usuario está en movimiento. La interacción entre el usuario, el dispositivo móvil y el sistema, debe ser lo suficientemente simple, conveniente, intuitiva, flexible y eficiente. Adi-

cionalmente, las interfaces de usuario deben ser diseñadas para ahorrar la mayor energía posible.

A diferencia de las computadoras de escritorio que utilizan un teclado y un ratón como principales dispositivos de entrada y un monitor como dispositivo de salida, las interfaces de usuario de los dispositivos móviles pueden soportar más de un método de entrada y salida, por ejemplo, teclado, reconocimiento de voz, sonido, alertas luminosas, vibraciones, además de texto e imágenes.

Las investigaciones recientes en HCI se han enfocado en lo que se conoce como interfaces multimodales, las cuales combinan dos o más formas de entrada/salida en una misma interfaz de usuario, tales como la voz, teclado, *stylus*, gestos e incluso lenguajes corporales como expresiones faciales, movimientos del ojo y mímica [21, 22], y se caracterizan por reconocer de manera paralela múltiples entradas.

Se considera que estas interfaces pueden proveer una mayor usabilidad, en comparación a las interfaces de usuario tradicionales. Por ejemplo, este tipo de interfaces tienen el potencial de ser más intuitivas y fáciles de aprender debido a que implementan formas de interacción parecidas a las que se utilizan en la interacción humano-humano.

En particular, los dispositivos móviles, que generalmente presentan problemas de usabilidad debido a su tamaño reducido y su uso en ambientes que cambian constantemente, se pueden ver beneficiados por el uso de interfaces multimodales [23].

Según Sharon Oviatt y Philip Cohen [24], la razón más importante para el desarrollo de interfaces de este tipo, es su potencial para expander ampliamente la accesibilidad a diversos tipos de usuarios no especialistas y promover nuevas formas de llevar a cabo la interacción entre humano-computadora. Dado que existen una gran variedad de diferencias en las habilidades de las personas y preferencias para utilizar diferentes modos de comunicación, las interfaces multimodales pueden incrementar la accesibilidad a usuarios de diferentes edades, niveles de preparación, estilos cognitivos, lenguajes, impedidos de forma motriz o sensorial, o incluso con enfermedades temporales. Esto gracias a que una interfaz multimodal permite al usuario seleccionar la forma en que desea interactuar con el dispositivo.

Una gran variedad de tecnologías de reconocimientos son incorporadas a las interfaces multimodales, incluyendo voz, visión y biometría. Además, los investigadores en el área de HCI prevén sistemas futuristas multibiométricos-multimodales-multisensores (M3) que puedan interpretar y responder al lenguaje natural y comportamiento de los usuarios [25]. Como la investigación en interfaces multimodales sigue progresando y el costo computacional va disminuyendo, la aplicación de este tipo de enfoques para HCI móviles es sólo cuestión de tiempo [1].

2.4 Aplicaciones móviles multimodales

Durante la revisión bibliográfica de la literatura disponible relacionada a las áreas de cómputo móvil e interfaces multimodales, se encontraron los siguientes proyectos y publicaciones que son de interés para el tema de investigación:

Speak4it

Speak4it [26] es un proyecto de búsqueda orientada al consumidor que aprovecha la entrada/salida multimodal para permitir a los usuarios buscar información sobre comercios locales en dispositivos móviles. El usuario puede realizar consultas por voz en el sistema o combinar voz y gestos para realizar las búsquedas.

Cuando la aplicación se inicializa se muestra al usuario un mapa del área en la que se encuentra ubicado y puede seleccionar el botón "*Speak/Draw*" o posicionar el dispositivo a la altura del oído para iniciar una consulta por voz. Los usuarios pueden buscar comercios a partir de su nombre, categoría o palabras claves asociadas.

En esta aplicación es posible manipular la vista del mapa usando los gestos de arrastrar (*drag*) y pellizcar (*pinch*), y por medio de la voz.

Además de los comandos de voz unimodales para la búsqueda de comercios y la manipulación del mapa, el sistema soporta comandos multimodales en los que la ubicación se traza en el mapa. El sistema soporta los gestos de apuntar, delimitar un área y dibujar una línea. Por ejemplo, el usuario puede decir "restaurante francés aquí" en conjunto con el gesto de apuntar o delimitar un área. Para el gesto de apuntar, el sistema regresará como resultado los restaurantes cercanos al punto señalado. Para el gesto del área, regresa los resultados que se encuentran dentro de ella. Los usuarios pueden también buscar comercios a lo largo de una ruta específica a través del gesto de dibujar una línea. Por ejemplo, el usuario puede decir "gasolineras", dibujar una línea en el mapa, y el sistema regresará las gasolineras más cercanas a esa ruta.

Esto se lleva a cabo a través de una aplicación cliente en un iPhone o iPad, la cuál se comunica por HTTP con una plataforma de búsqueda multimodal que se encarga de realizar el reconocimiento de voz y gestos, el análisis de la consulta, la geodecodificación y la búsqueda.

LS4M: Live Search for Mobile

Live Search for Mobile [27] es un proyecto para celulares que permite a los usuarios interactuar con portales de información basados en Web. La implementación actual de la aplicación se enfoca a información basada en negocios locales: su número telefónico y dirección, cómo llegar a ellos, reseñas, mapas y tráfico. Presenta la interfaz de reconoci-

miento de voz que fue desarrollada para la aplicación y que permite al usuario interactuar con el dispositivo por medio de la voz.

Esta aplicación permite al usuario realizar la búsqueda de información de lugares proporcionando el nombre del lugar o su dirección, o bien ingresándola utilizando la interfaz gráfica.

La arquitectura de LS4M está organizada en torno a un conjunto de servidores de retransmisión que son mediadores entre un grupo de máquinas que realizan el reconocimiento de voz y los servidores de búsqueda local de MSN. La interacción es iniciada por la aplicación cliente la cuál, dependiendo de su estado, puede enviar a los servidores de retransmisión ya sea, una solicitud para ejecutar un servicio web específico o la transcripción de una expresión. El único cómputo realizado del lado del cliente es un procesamiento mínimo de la señal y la lógica del flujo de control. Las solicitudes del cliente son recibidas por un balanceador de cargas, el cual reenvía el mensaje a un servidor de retransmisión disponible. En este punto, el mensaje es reenviado a un servidor de búsqueda MSN o a un balanceador de cargas secundario que controla al conjunto de servidores de reconocimiento de voz.

TravelMan

TravelMan es un proyecto [28] que presenta una interfaz multimodal para guía de rutas. Las modalidades de entrada/salida incluyen síntesis de voz, reconocimiento de voz, tacto, texto, navegación física, gestos físicos, audio e información GPS.

TravelMan provee información del transporte público en Finlandia, como el metro, tranvía y autobús. Presenta dos funcionalidades principales: la planeación de trayectos y una guía interactiva durante el recorrido. En la fase de planeación de viajes, el usuario proporciona la dirección o el lugar de partida y destino, puede seleccionar preferencias para excluir o incluir ciertos medios de transporte, especificar la velocidad al caminar y el número de resultados en la búsqueda. Las rutas pueden ser almacenadas y compartidas entre los usuarios.

Cuando el usuario proporciona la dirección, el sistema busca los planes de viaje o rutas adecuados. Posteriormente, el sistema permite al usuario navegar a través de las rutas sugeridas. Cada ruta tiene una descripción que consiste de un número de subrutas con diferentes medios de transporte. Cada subruta contiene información detallada como las paradas del metro o autobús. Después de seleccionar una ruta, el usuario puede simplemente escuchar el progreso del recorrido o navegar interactivamente en la descripción de la ruta.

El diseño de la interfaz multimodal de la aplicación se basa en las siguientes modalidades:

Interfaz Gráfica de usuario por voz tipo *fisheye*: El uso de TravelMan está basado en menús multidimensionales que son manipulados por las teclas direccionales del teléfono. La interfaz se inspira en la técnica *fisheye*.

El elemento del menú se escucha en voz alta cuando es activado a través del sintetizador de voz. Una salida adicional de voz, es el título del menú y la funcionalidad actual de las teclas.

Gestos: Es posible navegar el menú utilizando gestos, inclinando el dispositivo hacia atrás, adelante, izquierda y derecha. La selección de una opción se puede realizar con los botones o tocando el dispositivo.

Sonidos y tacto: Adicionalmente la navegación en los menús se apoya en sonidos. Mientras el usuario se mueve entre las opciones se reproducen sonidos que indican la posición de la opción seleccionada en el menú (incrementa/decrementa el tono). Por otra parte se disponen de sonidos de una persona al caminar, trenes de metro, tranvías y autobuses, con diferentes entonaciones (velocidad constante, aceleración y decremento del sonido), para indicar el medio de transporte que se está utilizando a lo largo de la ruta.

Entrada de voz y texto: Utilizadas para proporcionar la dirección o nombres del lugar de partida y destino. En caso de que la entrada sea proporcionado por voz, el reconocimiento se realiza en el servidor, de obtenerse múltiples resultados se presenta un dialogo de aclaración. Si la entrada de la dirección se realiza por texto, cuando el usuario ingresa caracteres el sistema realiza una búsqueda de la dirección y "dice" en voz alta la secuencia de caracteres más probable de acuerdo al dominio actual. Esto se realiza en segundo plano, por lo que no interfiere con las acciones del usuario.

Posicionamiento y navegación física: Se utiliza el GPS para proporcionar la ubicación actual como dirección de partida y para obtener el contexto de ubicación durante el viaje. Durante el traslado, el usuario puede utilizar la cámara del teléfono, para leer matrices de datos que se integraron a algunas paradas de autobús y contienen la dirección de la parada con información adicional.

MOBILTEL: Mobile Multimodal Telecommunications Systems and Services

El proyecto MobilTel [29] establece actividades de investigación y desarrollo en el área de interfaces multimodales. El resultado es una arquitectura funcional para los sistemas móviles multimodales de telecomunicaciones. El comunicador MobilTel es una interfaz multimodal, gráfica y de voz en eslovaco. Las modalidades posibles son interacción por *stylus* y pantalla táctil, teclado y pantalla en la que la información es más fácil de presentar y proporciona la posibilidad de utilizar hipervinculos y menús desplegables.

En el artículo se describe el método de interacción entre el dispositivo móvil (PDA) y el comunicador MobilTel. Por parte del PDA existen principalmente dos servicios ejecu-

tándose durante la interacción multimodal:

- Navegador web, como la interfaz gráfica de usuario.
- Servidor de audio de voz a la par del módulo de control principal.

El módulo de control principal ejecuta la correspondiente *URL (Universal Resource Locator)* de la *GUI (Graphical User Interface)* en el navegador web en pantalla completa. La URL correspondiente se envía utilizando un canal de control TCP/IP.

La voz es transferida a través de canales TCP/IP *full duplex* en un formato de datos predefinido.

El servidor de GUI lo proporciona por omisión el navegador web en modo de pantalla completa. Si el usuario interactua con un *stylus* o el teclado, la pantalla es recargada con el contenido actualizado.

El dialogo de voz está escrito en el lenguaje VoiceXML y el dialogo GUI multimodal correspondiente está escrito en el lenguaje HTML utilizando PHP y JavaScript.

Al final del documento se presentan dos ejemplos de servicios multimodales, el primero de ellos es un servicio que provee información sobre los horarios del ferrocarril y sus conexiones. El segundo es un servicio del clima, proporciona información sobre el pronóstico del tiempo de las ciudades de Eslovaquia.

MMT: Multimodal Museum Tour

MMT [30] es un proyecto desarrollado con la finalidad de asistir a los usuarios finales durante una visita al museo, el cual puede mostrar el contenido de cada sala. La interfaz en esta fase se ve como un pizarrón donde se muestran las obras de arte que están en la sala. Cada imagen es un objeto *"point and speak"* o *"point and draw"*.

Cuando se inicia la visita el usuario abre la aplicación en su dispositivo móvil, el usuario puede acceder a la información usando interacción oral (ej. decir "sala uno"), entonces aparece en la pantalla un panel que contiene las obras de arte de la sala. Cada obra de arte es un panel de entrada de audio, así por ejemplo, el usuario puede:

- Apuntar y decir "zoom", de modo que se muestre una versión más grande de la imagen.
- Apuntar y decir "autor", para que se muestre un panel de salida de audio o video con la biografía del pintor.

Las obras de arte se pueden representar también como un panel de dibujo, donde el usuario puede escribir o dibujar y crear una representación simbólica de la obra que puede ser reconocida como un comando por el sistema.

Architecture for Multimodal Mobile Applications

Se trata de una arquitectura para aplicaciones móviles, que están distribuidas en una terminal (dispositivo móvil), la red y el *backbone*. Se incluye la descripción utilizando un prototipo [22].

El prototipo presentado se utiliza para brindar información a usuarios de T-Mobile sobre campañas y productos, existentes y nuevos que ofrece la compañía. Utiliza como modalidades de entrada la voz y *stylus*.

La implementación del prototipo requiere un servidor, una plataforma multimodal y un dispositivo móvil. En el prototipo descrito, el servidor ejecuta la versión de sistema operativo Microsoft Windows 2003 Enterprise, la aplicación de servidor Tomcat y la plataforma multimodal. Como dispositivo móvil se utiliza el T-Mobile MDA III, el cual es una PDA que funciona con el sistema operativo Mobile Windows 2003 y soporta como dispositivos de entrada un micrófono, una pantalla táctil, un teclado miniatura y una tecla de función *push-to-talk*. Para permitir entradas multimodales se instala el Microsoft Speech Plug-in para Pocket Internet Explorer.

El funcionamiento general de la aplicación es el siguiente: el usuario presiona la tecla *push-to-talk* del MDA III. Se inicia el reconocimiento automático de voz (ASR), se abre el micrófono y se graba la voz de entrada: el sonido con la voz del usuario es enviado al ASR de lado del servidor, donde es recibido como entrada y es analizado de acuerdo a la gramática predefinida, hasta que el botón deja de ser presionado. Una vez realizado el reconocimiento de voz, el archivo resultante es enviado en el lenguaje Extended Multimodal Annotation (EMMA), de regreso al procesador de integración del dispositivo. Si se llega a reconocer una nueva modalidad de entrada, en un intervalo menor a 0.75 segundos, las entradas actuales son fusionadas, en caso contrario son procesadas como múltiples entradas de forma secuencial. De esta tarea es responsable el procesador de integración implementado en Javascript en el dispositivo móvil.

MUMS: Multimodal Route Navigation System

MUMS [31] es un sistema que permite a los usuarios de dispositivos móviles realizar consultas de las rutas de transporte público, utilizando cualquier combinación de voz y *stylus*. El sistema proporciona información de navegación a través de voz y representaciones de mapas gráficos.

La interfaz de usuario está implementada en un PDA, pero básicamente se podría utilizar cualquier dispositivo móvil con pantalla táctil, grabación y reproducción de voz, y conexión inalámbrica.

Los usuarios pueden recorrer y ampliar el mapa en tiempo real, mientras el botón de grabación está presionado, se graba la voz y los gestos sobre el mapa. Las entradas pueden

ser unimodales ya sea por voz o por gestos de señalización, o multimodales con una combinación de ambos. Si se acepta la ruta sugerida por el sistema, se inicia la navegación y los usuarios son guiados a lo largo de la ruta paso a paso.

MAGA: Multimodal Archaeological Guide in Agrigento

MAGA es un proyecto con un enfoque multimodal y omnipresente para guías virtuales en sitios de patrimonio cultural, el cual integra agentes inteligentes de conversación y tecnologías de reconocimiento/síntesis de voz a un framework basado en localización RFID y a un framework basado en el intercambio de datos vía Wi-Fi [32]. La aplicación se puede acceder usando un PDA equipado con un módulo de auto localización basado en RFID, mientras que el servicio de recuperación de información es proporcionado por medio de una interacción oral y lenguaje natural con un agente conversacional, y reforzado por la integración de capacidades de razonamiento en su módulo de representación del conocimiento.

El sistema está compuesto de dos principales áreas: el área interactiva y el área racional. La primera provee un acceso multimodal a los servicios del sistema. Esta área incorpora la interacción por voz, la interfaz de *point-and-click* y la auto-ubicación. Esto permite adquirir las entradas del usuario en forma multimodal y alimentar el área racional con preguntas simples en forma de texto.

Después del proceso de recuperación de información (IR), el área racional regresa texto, imágenes y links al área interactiva para el proceso de salida.

TMAW: Telephone-Supported Multimodal Accessible Web System

TMAW [33] es creado de acuerdo con el concepto de accesibilidad Web y accesibilidad universal, cuyo objetivo es proporcionar acceso máximo a la Web para cualquier usuario, en cualquier momento, en cualquier lugar. Como resultado, TMAW crea un ambiente con una operación dualmodal (ratón/teclado) y una presentación dualmodal (visual/audio), y soporta el acceso telefónico. Las técnicas utilizadas en el desarrollo de TMAW incluyen mecanismos de interacción multimodal, síntesis de texto a voz (TTS) y la tecnología VoiceXML.

Los usuarios pueden combinar dos modalidades o seleccionar una en específico, dependiendo de cómo los usuarios deciden llevar a cabo la tarea o la situación del medio ambiente. Las dos formas de acceso son:

Modo de acceso desde la computadora: Esta es la forma tradicional de los usuarios para navegar en Internet. Sin embargo, TMAW proporciona una interfaz más flexible con una presentación y una operación dualmodal.

Modo de acceso desde el teléfono: Utilizando la tecnología VoiceXML, se crea un ambiente que permite al usuario navegar en la Web a través de PSTN (Public Switched Telephone Network) vía teléfono o teléfono celular.

Existen actualmente aplicaciones que permiten llevar a cabo ciertas tareas en los dispositivos móviles utilizando comandos de voz, por mencionar algunas de las más populares se tiene:

Siri

Siri[7] es una aplicación de asistente personal para iOS. La aplicación utiliza procesamiento del lenguaje natural para responder preguntas, hacer recomendaciones y realizar acciones mediante la delegación de las solicitudes a un conjunto de servicios web que va en aumento. Siri fue adquirida por Apple Inc. el 28 de abril de 2010.

Siri está disponible en el *smartpone* iPhone 4S, permite utilizar comandos de voz para enviar mensajes, agendar reuniones, realizar llamadas, entre otras tareas que tiene disponible este dispositivo.

Vlingo

Vlingo[8] es un asistente virtual para dispositivos móviles, se encuentra disponible para diferentes dispositivos con Android, iPhone, BlackBerry, Windows Mobile y Nokia.

El sistema cuenta con un sistema de reconocimiento de voz muy preciso. Utiliza la tecnología *voice-to-text* en combinación con un "motor de intentos" para ayudar al usuario a completar rápidamente la acción deseada.

Google Voice

Google Voice[9] es un sistema de búsqueda por voz disponible para dispositivos móviles Android e iPhone.

Al activar la aplicación y decir en voz alta las palabras que se desean encontrar, Google Voice lo transforma en resultados del buscador de google.

Los dispositivos que cuentan con la versión Android 2.1 o posterior proporcionan la opción de utilizar este sistema para brindar la funcionalidad de *voice-enabled keyboard*, la cual se activa al presionar el botón de micrófono que se muestra en el teclado, esta opción permite dictar mensajes de texto en lugar de escribirlos.

[7] http://www.apple.com/iphone/features/siri
[8] http://www.vlingo.com
[9] https://www.google.com/voice

Adicionalmente, el sistema Android proporciona el API `android.gesture`[10], para el manejo de gestos. La cual permite al desarrollador almacenar, cargar, dibujar y reconocer gestos que desea implementar en una aplicación.

La investigación realizada de este tipo de proyectos y aplicaciones existentes actualmente, ha permitido recabar información útil para este proyecto de tesis y delimitar el alcance del mismo. Como se puede observar en la primera parte de esta sección, las aplicaciones presentadas se consideran multimodales ya que incluyen en su funcionalidad más de una modalidad de entrada/salida, sin embargo ninguna de ellas realiza el procesamiento de modalidades en el dispositivo móvil y no todas realizan la integración de modalidades simultáneamente. Cabe mencionar que no se encontró ninguna aplicación multimodal para dispositivos móviles que permita ejecutar servicios de telefonía, ni proactivos.

[10]http://developer.android.com/resources/articles/ gestures

Capítulo 3

Propuesta de solución

El objetivo principal de este proyecto de tesis es diseñar e implementar mecanismos de activación multimodal de servicios en dispositivos móviles. Como resultado final se espera obtener un conjunto de componentes de software que sean integrados para su ejecución en dichos dispositivos. Este conjunto de componentes puede ser visto como un sistema multimodal, ya que su interacción tiene un fin común: la activación de un servicio en el dispositivo móvil, a través de diferentes modalidades.

Cuando se desea implementar un sistema de software, es de necesario llevar a cabo un conjunto de actividades establecidas en el proceso de desarrollo de software, que permitan la definición de los objetivos y de los elementos que intervienen para alcanzarlos. Estas actividades, de acuerdo a la disciplina de ingeniería de software, pueden ser vistas como la etapa de análisis y diseño.

En las siguientes secciones de este capítulo se presentan algunas de las actividades llevadas a cabo para modelar el sistema que se desea desarrollar.

3.1 Funcionalidad general del sistema

Se propone desarrollar un sistema para dispositivos móviles que permita a los usuarios interactuar con el dispositivo de la forma más natural posible, es decir, utilizando las capacidades con las que normalmente se comunica con otras personas (voz, tacto, gestos). De esta manera, se pretende hacer más accesible el uso de los dispositivos móviles actuales con avanzada tecnología (*smartphones*) a usuarios que sufren de algún impedimento que no les permita manipular estos dispositivos de la manera habitual.

Se plantea por tanto, la necesidad de desarrollar un conjunto de interfaces, adicionales a las que provee normalmente el dispositivo móvil, que permitan la interacción a través de comandos de voz, gestos que se dibujen sobre la pantalla táctil, gestos que se describan con movimientos del dispositivo móvil y comandos de texto. Se requiere para ello

que el dispositivo se encuentre a la espera de los posibles eventos (entradas), que deberá procesar para dar una respuesta al usuario.

Por otra parte, se desea que el sistema realice un análisis de contexto de conectividad del dispositivo, es decir, la intensidad de señal de redes inalámbricas y celulares disponibles para que, con base en esta información, determine si es posible concretar el servicio solicitado.

Los servicios que se desean activar por medio del sistema multimodal corresponden, por lo menos, a los utilizados con mayor frencuencia por los usuarios de *smartphones*: realizar llamadas, enviar mensajes, acceder al navegador de Internet, enviar un correo electrónico, tomar una fotografía, sin descartar a futuro otros servicios.

A continuación se describe un ejemplo del escenario general de uso del sistema:

- El usuario del dispositivo móvil desea realizar una tarea (ej. realizar una llamada a un contacto) por lo que utiliza una modalidad de interacción (comando de voz, gestos, texto) para activar el servicio disponible en el dispositivo móvil que le permita llevar a cabo dicha tarea (marcador telefónico) e ingresa en otra modalidad el comando asociado al contacto que desea llamar.

- El dispositivo móvil puede percibir, a través de los sensores con los que cuenta, el tipo de entradas que está recibiendo (voz, gesto, tacto) y las procesa, dando como respuesta la ejecución del servicio solicitado.

- De acuerdo al contexto (intensidad de la señal de celular o inalámbrica), el sistema activa el servicio solicitado si se cuenta con la conexión que requiere para funcionar correctamente. En caso contrario, el sistema informa al usuario que no es posible concretar el servicio solicitado. En el caso del marcador telefónico, revisa el estado de la conexión a la red celular.

3.2 Arquitectura general del sistema

Una tarea importante en el planteamiento de una propuesta de solución, que conlleve la implementación de un sistema, es la identificación de los diferentes elementos que lo modelan y sus relaciones. Esto permite obtener una vista abstracta de la funcionalidad y la manera en que los diferentes elementos contribuyen para lograrla.

En la figura 3.1 se pueden observar los diferentes elementos y sus relaciones, que logran modelar el sistema descrito en la sección anterior. A continuación se presenta una breve descripción de los elementos.

- **Usuario:** persona que desea activar algún servicio disponible en el dispositivo móvil, para lo que utiliza diferentes modalidades de interacción (voz, texto, gestos).

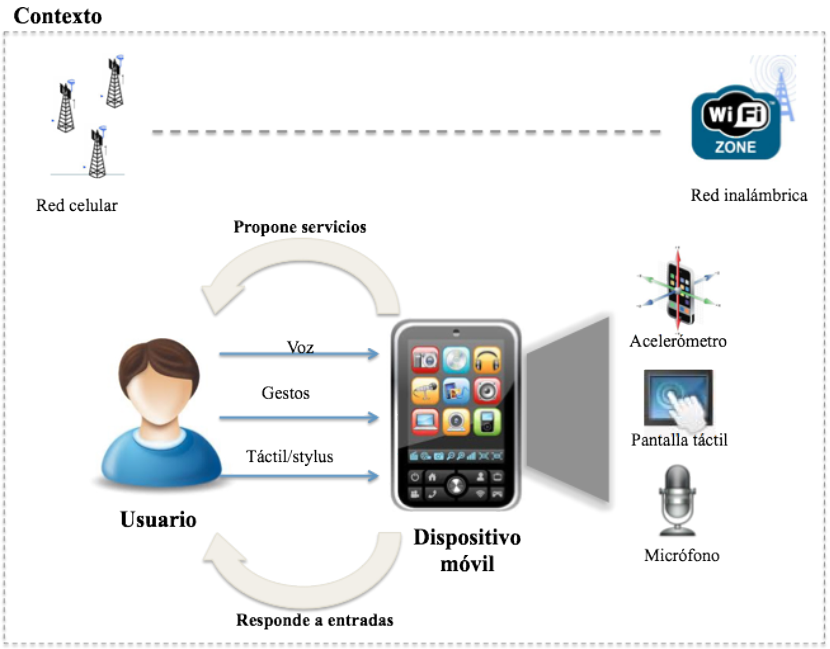


Figura 3.1: Arquitectura general del sistema

- **Dispositivo móvil:** equipo electrónico con capacidad de cómputo; para este caso, hace referencia a un *smartphone* que tiene disponible diferentes servicios (llamadas, mensajes, música, correo electrónico, etc). Adicionalmente se encuentra equipado con diferentes sensores como son acelerómetro, pantalla táctil y micrófono, que permiten la percepción de diferente modalidades de interacción.

- **Contexto:** entorno de conectividad disponible para el dispositivo en un momento determinado. En específico, está determinado por la intensidad de señal de la red celular e inalámbrica del dispositivo.

3.3 Especificación de requisitos

La actividad más importante dentro del análisis es la especificación de los requisitos, ya que permite identificar y establecer el comportamiento del sistema a desarrollar, así como sus restricciones.

Durante esta actividad se clasifican a los requisitos como funcionales y no funcionales. Los requisitos funcionales especifican las acciones que debe ser capaz de cumplir el sistema y los requisitos no funcionales son cualidades y restricciones propias del sistema. Además se establece la importancia de los requisitos para incorporarlos incrementalmente en el diseño.

A continuación se presentan los requisitos funcionales y no funcionales identificados en esta actividad:

- **Requisitos funcionales**

1. Agregar un comando de voz asociado a un servicio específico disponible en el dispositivo.
2. Determinar el gesto táctil asociado a un servicio en particular disponible en el dispositivo.
3. Añadir un comando de texto asociado a un servicio disponible en el dispositivo.
4. Describir el gesto físico asociado a un servicio específico disponible en el dispositivo.
5. Activar la ejecución del sistema multimodal en el dispositivo móvil.
6. Ejecutar los servicios disponibles en el dispositivo móvil por medio de comandos de voz.
7. Iniciar los servicios disponibles en el dispositivo móvil por medio de gestos dibujados en la pantalla táctil del dispositivo.
8. Activar los servicios disponibles en el dispositivo móvil por medio de comandos de texto.
9. Ejecutar los servicios disponibles en el dispositivo móvil por medio de movimientos específicos del dispositivo (gestos físicos).
10. Iniciar los servicios disponibles en el dispositivo móvil por medio una combinación de diferentes modalidades.
11. Monitorear el contexto del dispositivo en términos de intensidad de señal celular y redes inalámbricas.
12. Tratar de establecer una conexión a una red inalámbrica en caso de que no exista y sea requerida.
13. Determinar con base en el contexto, si el servicio que se desea activar se puede concretar.
14. Mostrar y seleccionar los servicios disponibles en el dispositivo móvil a los que es posible asociar un comando en alguna modalidad.
15. Visualizar los servicos a los que se les ha asociado un comando en las diferentes modalidades.

- **Requisitos no funcionales**

1. La activación de servicios del dispositivo móvil debe tomar a lo más 5 segundos.
2. La propuesta de opciones de servicio con base en el contexto debe tomar a lo más 3 segundos.
3. La interfaz gráfica debe ser intuitiva para el usuario.

3.4 Casos de uso

La técnica de casos de uso permite capturar los requisitos potenciales de un sistema de software. Cada caso de uso específica cómo debe efectuarse la interacción entre el actor y el sistema para cumplir un objetivo.

En la tabla 3.1 se muestran los casos de uso obtenidos a partir de los requisitos funcionales especificados anteriormente y el mapeo de cada uno de ellos.

En la figura 3.2 se representa de manera gráfica la comunicación y el comportamiento del sistema a través de un diagrama de casos de uso.

A continuación se presenta una breve descripción de los casos de uso identificados:

- **Agregar un comando de audio:** el usuario ingresa un comando de audio asociado a un servicio disponible.
- **Añadir un comando de gesto táctil:** el usuario dibuja un gesto táctil sobre la pantalla del dispositivo y lo asocia a un servicio.
- **Especificar un comando de texto:** el usuario a través del teclado del dispositivo móvil ingresa una secuencia de caracteres para definir un comando de texto y lo relaciona a un servicio.
- **Definir un comando de gesto físico:** utilizando un movimiento descrito por el dispositivo móvil, el usuario define un comando de gesto físico asociado a un servicio disponible.
- **Activar la ejecución del sistema multimodal:** el usuario activa la función de reconocimiento del sistema multimodal en el dispositivo para que esté a la escucha de las diferentes modalidades de entrada.
- **Ejecutar un servicio por medio de un comando de audio:** el usuario dice en voz alta un comando de audio asociado a un servicio en el dispositivo y dicho servicio se ejecuta.

Tabla 3.1: Casos de uso y mapeo de requisitos funcionales

		Requisitos funcionales														
Casos de uso	**Título**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	Agregar un comando de audio	x														
2	Añadir un comando de gesto táctil		x													
3	Especificar un comando de texto			x												
4	Definir un comando de gesto físico				x											
5	Activar la ejecución del sistema multimodal					x										
6	Ejecutar un servicio a través de un comando de audio						x									
7	Iniciar un servicio utilizando un comando de gesto táctil							x								
8	Activar un servicio usando un comando de texto								x							
9	Ejecutar un servicio mediante un comando de gesto físico										x					
10	Iniciar un servicio utilizando una combinación de modalidades											x				
11	Analizar el contexto de conectividad del dispositivo											x		x		
12	Establecer una conexión a la red que requiere el servicio												x			
13	Mostrar y seleccionar los servicios disponibles														x	
14	Presentar los servicios entrenados															x

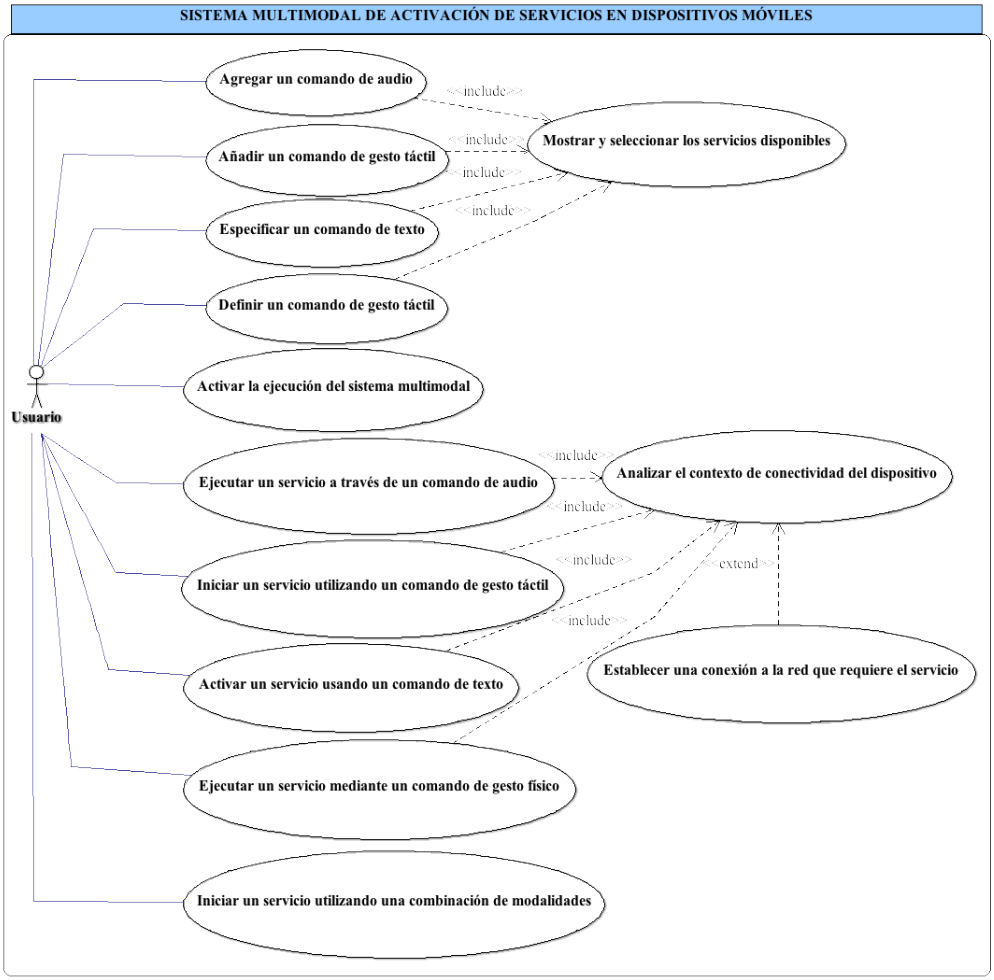


Figura 3.2: Diagrama de casos de uso

- **Iniciar un servicio utilizando un comando de gesto táctil:** el usuario dibuja en la pantalla táctil del dispositivo un gesto asociado a un servicio en el dispositivo y al hacerlo se inicia este servicio.

- **Activar un servicio usando un comando de texto:** el usuario introduce a través del teclado un comando de texto relacionado a un servicio y se presenta el nombre del servicio para que el usuario lo active.

- **Ejecutar un servicio mediante un comando de gesto físico:** el usuario realiza un movimiento del dispositivo (gesto físico), asociado a un servicio en el dispositivo y dicho servicio se ejecuta.

- **Iniciar un servicio utilizando una combinación de entradas:** el usuario ingresa de manera consecutiva dos comandos diferentes y el sistema los combina para activar el servicio al que están asociados.
- **Analizar el contexto de conectividad del dispositivo:** el sistema realiza un monitoreo del estado de conectividad del dispositivo móvil a las red celular e inalámbrica. Con base a la información que se obtiene, el sistema determina si es posible activar el servicio solicitado por el usuario.
- **Establecer una conexión a la red inalámbrica:** el sistema intenta establecer una conexión a alguna red inalámbrica que no tenga seguridad, en caso de que no cuente con una conexión establecida y el usuario desee ejecutar una aplicación que requiera conexión a Internet.
- **Mostrar y seleccionar los servicios disponibles:** el sistema presenta al usuario una lista con los nombres de los servicios disponibles, es decir, aquellos servicios a los que aún no se les ha asociado un comando en una modalidad específica.
- **Presentar los servicios entrenados:** el sistema muestra al usuario una relación con los nombres de los servicios a los que ya se les ha designado un comando en la modalidad seleccionada.

3.5 Arquitectura lógica

Durante la etapa de diseño se definió una arquitectura en tres capas (ver figura 3.3), para el sistema multimodal de activación de servicios en dispositivos móviles, con el objetivo de separar los diferentes módulos que lo conforman.

El diseño arquitectónico en capas permite abstraer los componentes del sistema en diferentes niveles y desarrollar de forma independiente cada uno de ellos. Por lo general se definen tres capas: capa de presentación, capa de lógica y capa de datos. La comunicación en este tipo de arquitectura se lleva a cabo sólo entre capas adayacentes.

A continuación se presenta una descripción general de cada una de las capas que conforman la arquitectura y sus módulos. En la figura 3.3 se puede apreciar una representación gráfica de la misma.

1. **Capa de presentación:** esta capa representa la interfaz gráfica de usuario y permite la interacción entre el usuario y el sistema. Además, despliega información y recibe las peticiones del usuario. Está compuesta principalmente por:
    - **Pantalla táctil:** presenta una interfaz activada por medio del tacto con la superficie de la pantalla del dispositivo móvil.

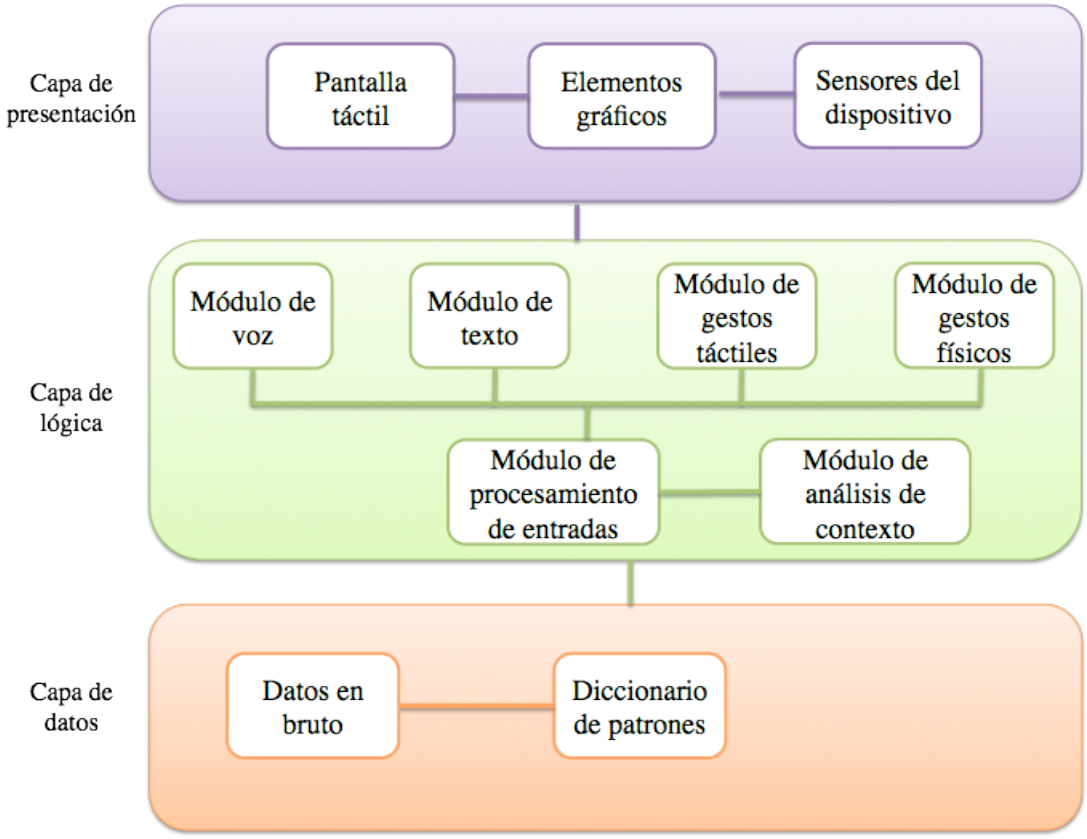


Figura 3.3: Arquitectura lógica del sistema multimodal

- **Elementos gráficos:** muestra al usuario los diferentes componentes que conforman la interfaz gráfica de usuario, como son botones, cuadros de texto, vistas, menú, etc.
- **Sensores del dispositivo:** representa los diferentes sensores que están integrados en el dispositivo móvil, los cuales permiten al usuario proporcionar una entrada en alguna de las modalidades disponibles. Por ejemplo, el micrófono se utiliza para definir un comando de audio y el acelerómetro para un comando de gesto físico.

2. **Capa de lógica:** a este nivel se encuentran los módulos encargados de procesar las entradas recibidas desde la capa de presentación y realizar las operaciones del sistema. Se conforma de siete módulos:

   - **Módulo de voz:** encargado de procesar una entrada de audio (voz). Graba el comando de voz recibido como una señal de audio, realiza un preprocesamiento para detectar en la señal, el inicio y fin del comando y elimina el posible

ruido presente; finalmente extrae las características significativas de la señal.

- **Módulo de texto:** recibe un secuencia de caracteres y conforme se ingresan, busca una posible coincidencia con algún patrón almacenado.
- **Módulo de gestos táctiles:** responsable de procesar los diferentes gestos que se dibujen sobre la pantalla táctil del dispositivo.
- **Módulo de gestos físicos:** encargado de procesar una entrada generada por un gesto físico, es decir, un movimiento en específico del dispositivo. Para realizar esta tarea obtiene los valores, registrados por el acelerómetro, correspondientes a los ejes inerciales x, y, z del dispositivo móvil.
- **Módulo de procesamiento de entradas:** dada una entrada de alguno de los módulos anteriores, es responsable de realizar la tarea de reconocimiento de patrones para determinar si la entrada corresponde a algún comando almacenado previamente (patrón) y de esta manera ejecutar el servicio correspondiente.
- **Módulo de análisis de contexto:** encargado de determinar, con base en el contexto de conectividad, si es posible activar el servicio asociado al comando que proporciona el usuario, por lo que una de sus tareas más importantes es supervisar el estado de conectividad del dispositivo a la red celular e inalámbrica.

3. **Capa de datos:** en esta capa se encuentran los módulos responsables del manejo de la información, los cuales reciben solicitudes de recuperación y almacenamiento de datos desde la capa lógica del negocio. Se conforma de tres módulos:

   - **Datos en bruto:** almacena en formato de archivo los diferentes valores que representan las características más importantes extraídas de las entradas (audio y gestos).
   - **Diccionario de patrones:** encargado de almacenar y recuperar la información relacionada a los diferentes patrones relacionados a las diferentes modalidades de comandos de entrada (audio, gesto táctil, gesto físico y texto).

3.6 Diagrama de clases

El diagrama de clases es una representación gráfica de las diferentes clases que serán utilizadas en el sistema que se desea desarrollar. Permite observar los diferentes atributos y métodos que están definidos para cada clase, así como sus relaciones con otras clases.

En la figura 3.4 se presenta el diagrama de las clases principales que modelan el sistema multimodal de activación de servicios en dispositivos móviles.

Las clases que se muestran en este diagrama representan las que abstraerán la funcionalidad que requiere para el sistema. A continuación se da una breve descripción de cada una de ellas:

- **Audio:** encargada de implementar las funciones que están relacionadas a la captura, almacenamiento y reconocimiento de un comando de audio.
- **TouchGesture:** contiene los métodos que se encargan de adquirir y procesar un gesto táctil a partir de un trazo realizado sobre la pantalla del dispositivo.
- **PhysicalGesture:** define las funciones que permiten grabar, almacenar y reconocer un comando de gesto físico.
- **Text:** establece los métodos que se requieren para procesar un comando de texto.
- **Preprocess:** define los procesos de preprocesamiento de una señal de audio y de las señales del acelerómetro que definen un gesto físico.
- **Recognition:** establece los métodos que se requieren para realizar el reconocimiento de un comando de gesto físico y de audio.
- **Algorithms:** contiene los algoritmos que permiten llevar a cabo los procesos de preprocesamiento y reconocimiento de comandos en las modalidades de audio y gesto físico.
- **Patterns:** abstrae los elementos comunes a un patrón en cualquier modalidad. De esta clase se hace una especialización de clases que definen los patrones de cada una de las diferentes modalidades: AudioPattern, TouchGesturePattern, TextPattern y PhysicalGesturePattern.
- **WiFiNetwork:** establece los métodos que permiten supervisar el estado de conexión a una red inalámbrica.
- **MobileNetwork:** define los métodos que se requieren para comprobar el estado de conexión a la red celular.
- **ContextManager:** provee los métodos que se utilizan para determinar si la el tipo de conexión que se requiere para activar un servicio está disponible.
- **InputsManager:** encargada de procesar las entradas que se reciben (comandos) para su reconocimiento y activar el servicio asociado a ellas.

La funcionalidad de cada uno de los objetos mostrados en el diagrama se traducirá en líneas de código durante el desarrollo, pudiendo ser necesario modelar nuevos objetos.

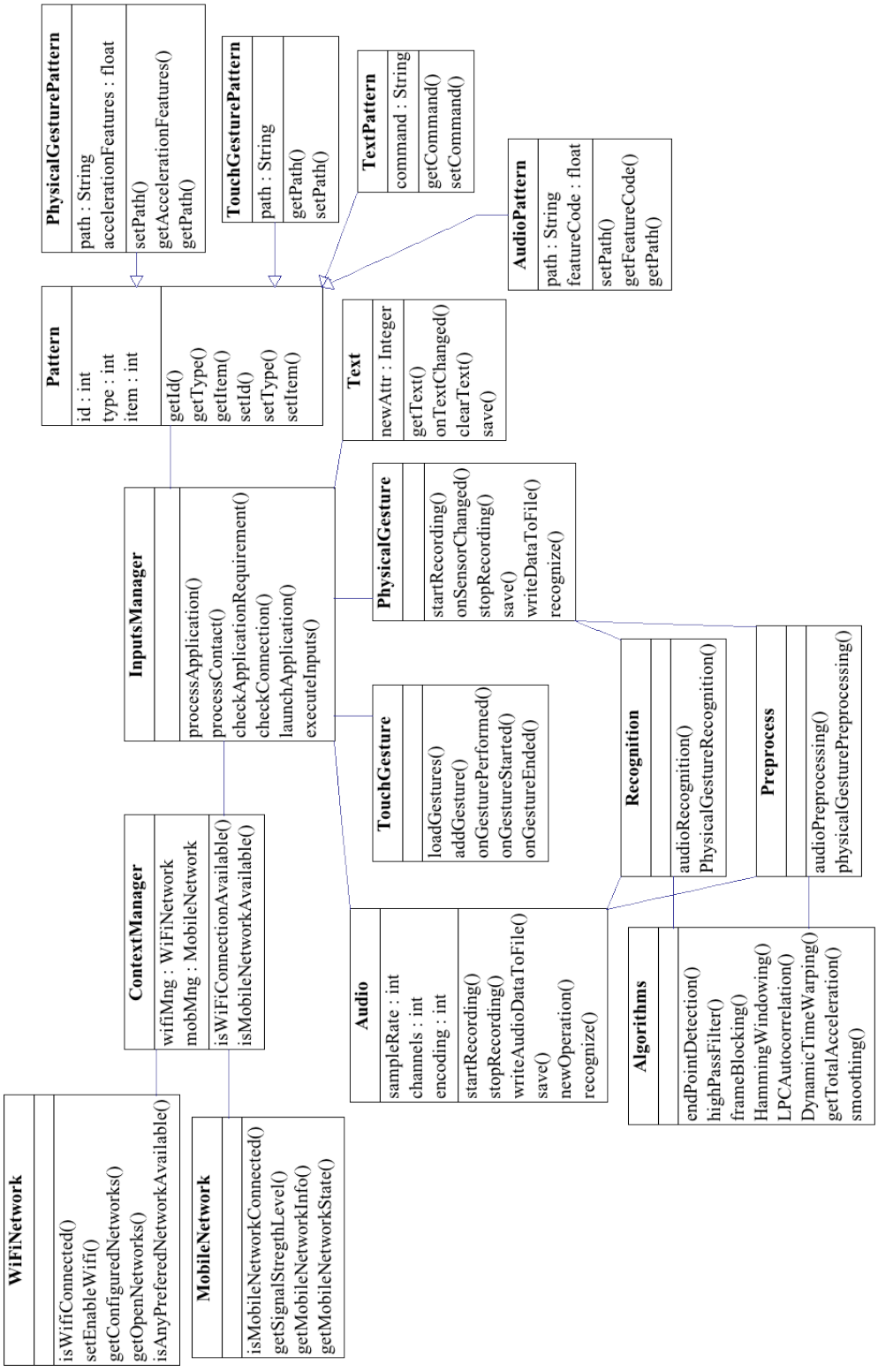


Figura 3.4: Diagrama de las clases principales del sistema multimodal

3.7 Diagramas de interacción

Durante el diseño se desarrolló una solución lógica basada en el paradigma orientado a objetos que cubre los requerimientos de la aplicación. Para ello se crearon diagramas de interacción que representan el modo en que los objetos colaboran entre sí para satisfacer los requisitos.

Existen dos tipos de diagrama de interacción, ambos basados en la misma información, pero cada uno enfatiza un aspecto particular: diagramas de secuencia y diagramas de colaboración.

Se seleccionaron los diagramas de secuencia debido a que muestran una interacción ordenada según la secuencia temporal de eventos, permitiendo interpretarlos de forma más clara.

3.7.1 Agregar en la configuración del sistema un comando de voz

La figura 3.5 muestra el diagrama de secuencia diseñado para representar el proceso de agregar en la configuración del sistema un comando de voz para una aplicación en específico, dicho de otra manera el almacenamiento de un patrón de comando de voz o proceso de entrenamiento del sistema.

El proceso inicia cuando el controlador *AudioController* recibe el mensaje *RecordAudio* desde la interfaz gráfica de usuario. Este controlador tiene la función de manejar todos los mensajes relacionados con el componente de audio. Este mensaje desencadena la creación de un objeto tipo *Audio*, invocando a la vez el método *startRecording* cuya principal función es indicar el inicio de la grabación de audio. Posteriormente se invocan una serie de mensajes de la clase *AudioRecord* para configurar los diferentes parámetros necesarios para llevar a cabo la grabación. La configuración se realiza por medio del envío de los mensajes:

- *getMinBufferSize*: se obtiene el tamaño mínimo del *buffer* requerido para la grabación del audio. Es necesario enviar como parámetros del mensaje la frecuencia de grabación, el tipo de canal y el formato de codificación.
- *createAudioRecord*: crea un objeto de la clase *AudioRecord* encargado de manejar las funciones asociadas a la grabación de audio. Para su creación, se pasa como primer parámetro la fuente de entrada del audio, seguido de los parámetros de frecuencia de grabación, el tipo de canal, el formato de codificación y por último el tamaño mínimo de *buffer* obtenido a través del mensaje *getMinBufferSize*.

Cuando se recibe el mensaje *StopRecording*, el objeto *Audio* invoca, a través del objeto *AudioRecord*, los mensajes *stop* y *release* encargados de indicar el fin de la grabación de audio y la liberación de los recursos adquiridos por este objeto respectivamente.

Si el usuario decide guardar la grabación, el controlador recibe el mensaje *SaveAudioCommand* y envía el mensaje *save* al objeto *Audio*, que a su vez se encarga de guardar los valores correspondientes a la grabación en un archivo, esto a través del método *saveAudioDataToFile*.

Dado que el objetivo final de este proceso es obtener un patrón correspondiente al comando de voz grabado, es necesario realizar las tareas de preprocesamiento y extracción de características representativas de la señal de audio almacenada en el archivo y de esta manera obtener el patrón para posteriormente guardarlo. Es por esto que se crea una instancia de la clase *Preprocess* encargada de enviar diferentes mensajes a la clase *Algorithms* para la ejecución de diferentes algoritmos que permiten realizar dichas tareas:

- *endPointDetection*: localiza y obtiene la sección de representa la voz dentro de la señal y elimina lo que no es. Recibe como parámetros la señal en bruto y retorna la señal que contiene la voz.

- *highPassFilter*: realiza un filtrado de la señal, recibe la señal que representa voz y regresa esta señal filtrada.

- *signalToFrames*: divide la señal en bloques que se traslapan entre sí, recibe la señal filtrada y regresa un arreglo que representa los bloques obtenidos.

- *hammingWindow*: a cada uno de los bloques obtenidos se aplica una función de ventaneo. Esta mensaje recibe como parámetro el bloque y devuelve el bloque al que se ha aplicado la función de ventaneo.

- *LPCAutocorrelation*: encargado de la extracción de las características principales de la señal para lo que realiza una codificación representativa de la misma. Recibe el bloque obtenido después de aplicar la función de ventaneo y retorna los coeficientes que representan la señal. Este mensaje se invoca por cada bloque al igual que el mensaje *hammingWindow*.

Posteriormente se crea un objeto tipo *Pattern* que se encarga de abstraer los atributos de un patrón y se realiza una instacia de la clase *DBPatterns* para su almacenamiento en la base de datos de patrones. Para esto se envían los mensajes *openDB*, *insertPattern* y *closeDB*, responsables de abrir la conexión a la base de datos, insertar los valores correspondientes al patrón y cerrar la conexión, respectivamente.

Al terminar este proceso se tiene almacenado en el sistema un patrón correspondiente a un comando de voz.

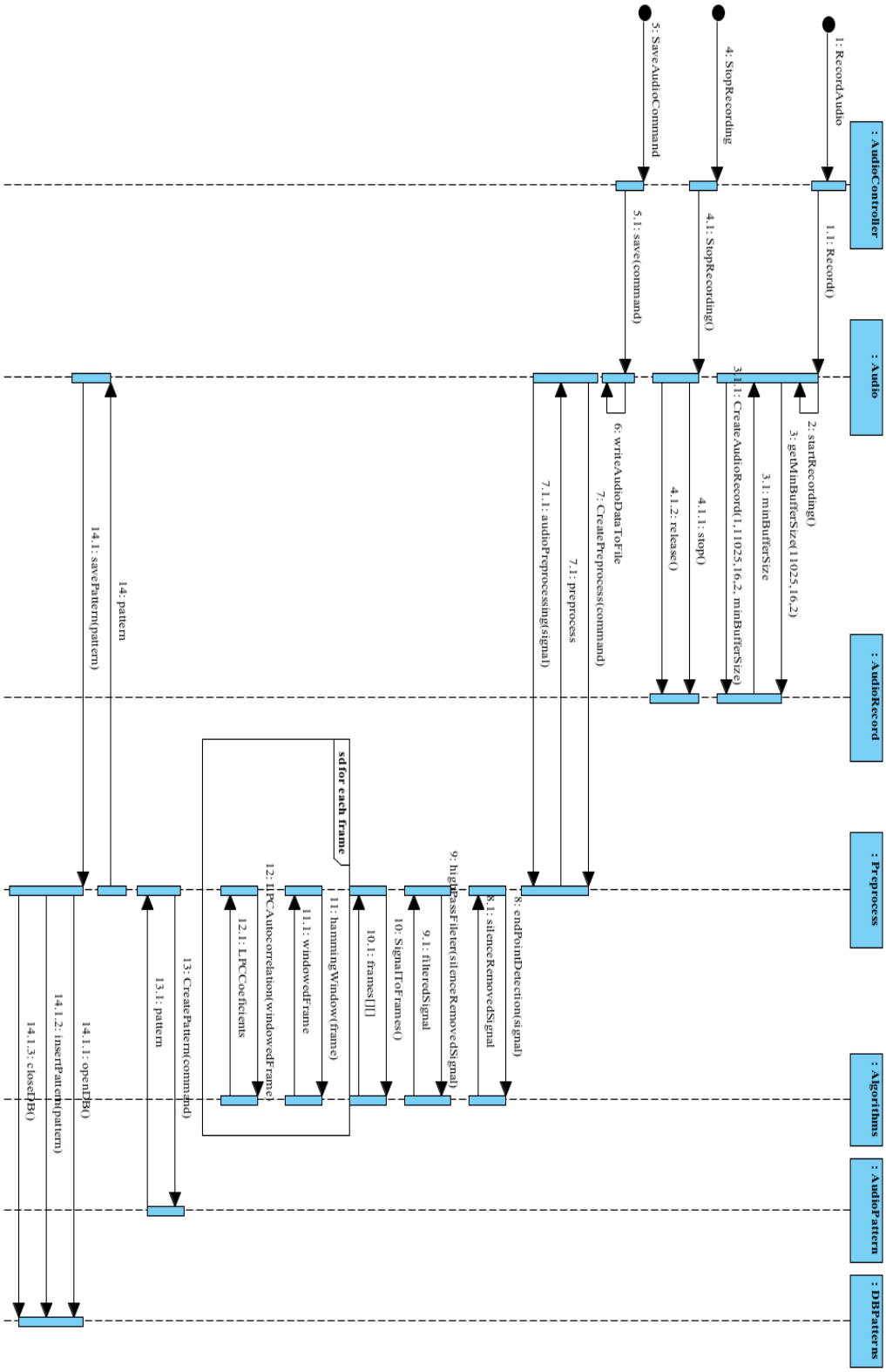


Figura 3.5: Diagrama de secuencia para almacenar un patrón de un comando de voz

3.7.2 Reconocer un comando de voz

La figura 3.6 muestra el diagrama de secuencia diseñado para representar el proceso de reconocer un comando de voz y ejecutar el servicio asociado a dicho comando, es decir, el proceso de prueba del sistema.

El proceso de reconocimiento inicia cuando se recibe desde la interfaz gráfica un evento que corresponde a la entrada de un comando de voz, esto a través del mensaje llamado *RecordAudio*.

Tal como se puede observar en la figura, cuando el controlador *AudioController* recibe el mensaje *RecordAudio* procede a realizar la misma secuencia de creación de objetos y envío de mensajes que se llevan a cabo en el proceso de entrenamiento del sistema, descritos anteriormente, hasta la obtención de un patrón correspondiente al comando que se está recibiendo como entrada.

De igual manera que en el proceso de entrenamiento se crea un objeto *Preprocess* al que se le delega la responsabilidad de enviar los mensajes para realizar el preprocesamiento y extracción de características de la señal de audio actual.

Una vez obtenido el patrón correspondiente al comando recibido (*pattern*), se crea un objeto *Recognition* y éste invoca el mensaje *audioRecognition*, enviando como parámetro dicho patrón y éste último a su vez utiliza una instancia de la clase *Algorithms* para la ejecución del algoritmo encargado de realizar la comparación entre patrón actual y los que se encuentran almacenados en el sistema. Esto se realiza a través del mensaje *DynamicTimeWarping*, el cual retorna el patrón reconocido como el más similar al actual.

Finalmente, se crea una instancia de la clase *InputsManager* y se envía el mensaje *launchService* para realizar la ejecución del servicio relacionado al patrón reconocido.

3.7.3 Agregar en la configuración del sistema un comando de gesto físico

En la figura 3.7 se presenta el diagrama de secuencia diseñado para representar el proceso de agregar en la configuración del sistema un comando de gesto físico para una aplicación en específico, equivalente al almacenamiento de un patrón de comando de gesto físico.

Este proceso inicia cuando el usuario envía por medio de la interfaz gráfica el mensaje *RecordGesture*, indicando que desea grabar el gesto que describa con el movimiento de su dispositivo móvil. Este mensaje lo recibe el controlador *PGController* encargado de manejar los mensajes relacionados al módulo de gestos físicos.

Cuando el controlador al recibe el mensaje crea un objeto de tipo *PhysicalGesture*, el cual a su vez invoca el mensaje *getSystemService* especificando el sensor del que se desea estar a la espera de valores. En el caso de gestos físicos, el sensor que se utiliza para describirlos es el acelerómetro.

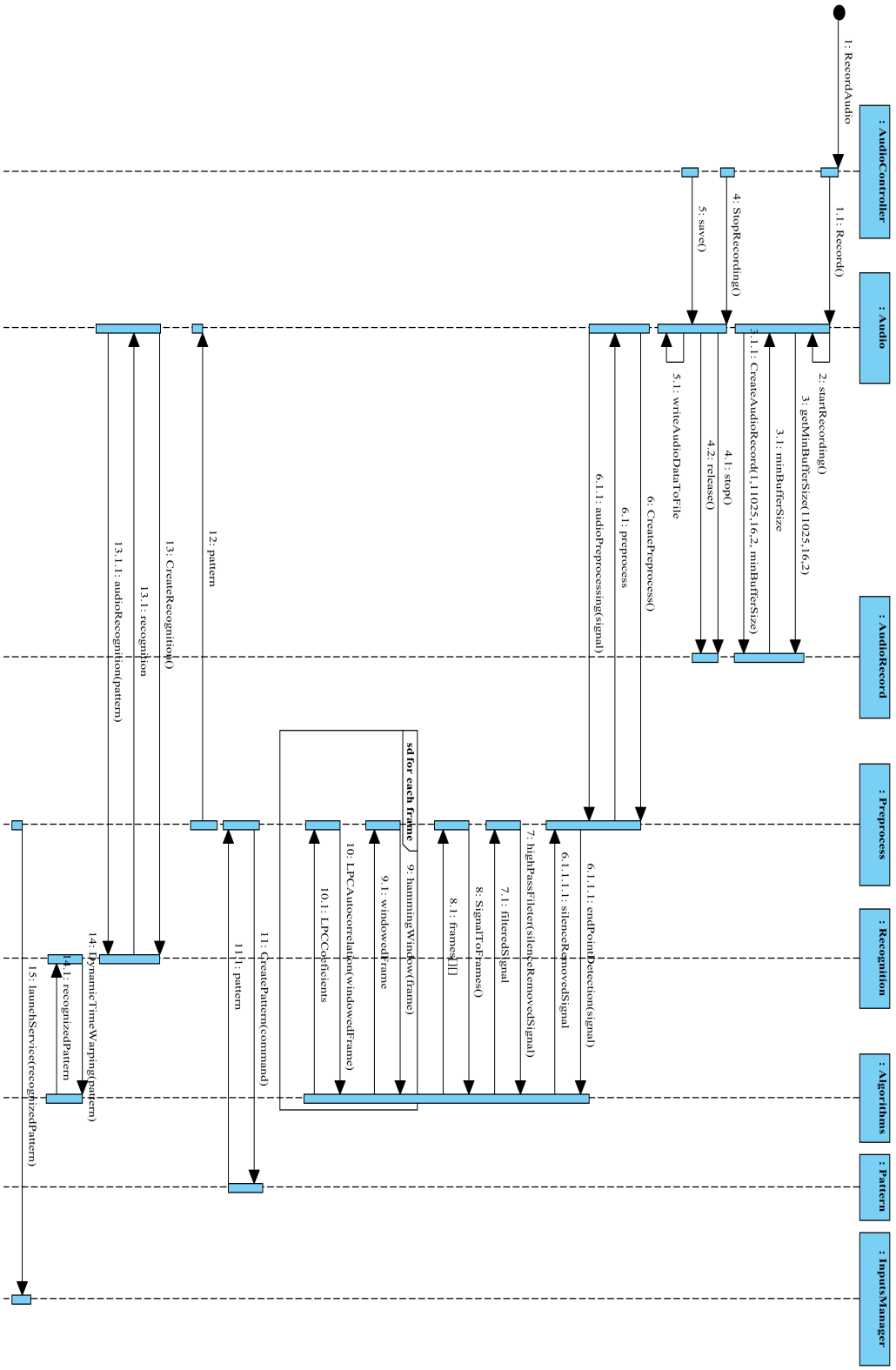


Figura 3.6: Diagrama de secuencia para almacenar un patrón de un comando de voz

Se crea un objeto *SensorManager* que nos permite tener acceso a los sensores. Este objeto envía los mensajes *getDefaultSensor* pasando como parámetros el tipo de sensor al que se desea acceder y regresa una instancia del sensor.

Para que el sistema esté a la espera de los datos registrados por el sensor se invoca el mensaje *registerListener,* indicando el sensor y la velocidad con la que se desea obtener los datos registrados por él mismo.

Al recibir un evento *SensorEvent*, es decir un cambio en los valores medidos por el sensor acelerómetro, se procede a obtener los valores registrados para cada uno de los ejes *x, y, z*, a través de envío de mensajes *getAxisValue* especificando cada uno de los ejes.

Cuando se recibe el mensaje *StopRecording*, el objeto *PhysicalGesture* deja de estar a la espera de cambios en los valores del sensor.

Si el usuario decide guardar el gesto físico, el controlador recibe el mensaje *SavePhysicalGesture* y envía el mensaje *save* al objeto *PhysicalGesture* que a su vez se encarga de guardar en archivo, los valores correspondientes a los registrados en los ejes *x, y, z* hasta el momento de recibir el mensaje *StopRecording*. Esto se realiza a través del método *saveDataToFile*.

A partir de los valores almacenados se espera obtener el patrón correspondiente al comando de gesto físico realizado, por lo que se requiere realizar una tarea de preprocesamiento y obtener el patrón representativo del gesto para posteriormente guardarlo.

Debido a lo anterior, se crea una instancia de la clase *Preprocess* encargada de enviar los mensajes *getTotalAcceleration* y *smoothing* a la clase *Algorithms* para la ejecución del algoritmo que obtiene la señal de aceleración total y el que realiza el proceso de suavizado de la señal resultante, respectivamente.

Posteriormente se crea un objeto tipo *Pattern* que se encarga de abstraer los atributos de un patrón y se realiza una instacia de la clase *DBPatterns* para su almacenamiento en la base de datos de patrones.

Los mensajes que permiten almacenar el patrón son *openDB*, *insertPattern* y *closeDB* que, como se mencionó con anterioridad, son responsables de abrir la conexión a la base de datos, insertar los valores correspondientes al patrón y cerrar la conexión, respectivamente.

Al terminar este proceso se tiene almacenado en el sistema un patrón correspondiente a un comando de gesto físico.

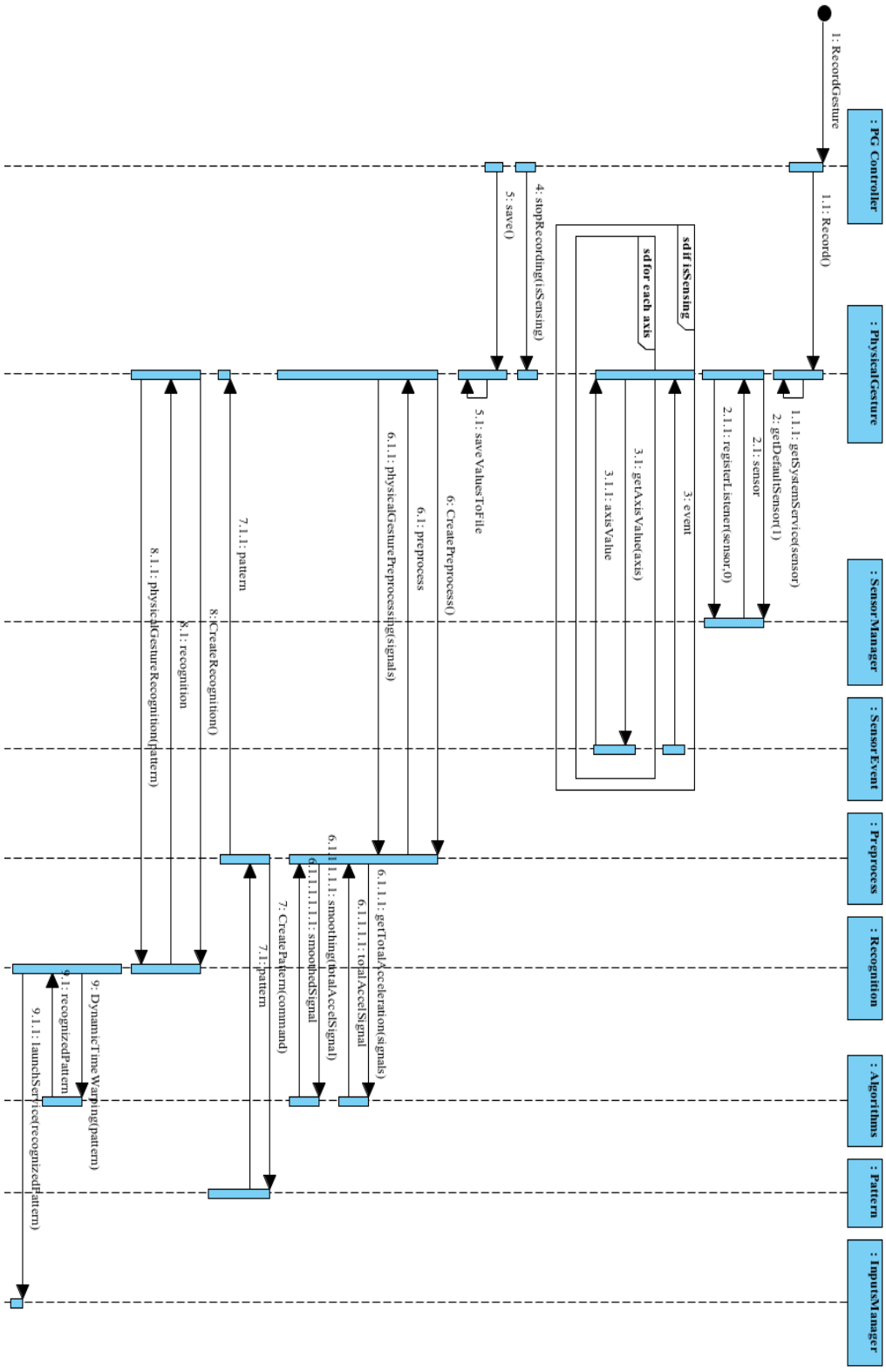


Figura 3.7: Diagrama de secuencia para almacenar un patrón de un gesto físico

3.7.4 Reconocer un comando de gesto físico

La figura 3.8 muestra el diagrama de secuencia diseñado para representar el proceso de reconocer un comando de gesto físico y ejecutar el servicio asociado a dicho comando.

El proceso de reconocimiento inicia cuando se recibe desde la interfaz gráfica un evento que corresponde a la entrada de un comando de gesto físico, es decir que el usuario describe un movimiento representativo con su dispositivo, esto a través del mensaje *RecordGesture*.

Al recibir el mensaje *RecordGesture*, el controlador *PGController* procede a realizar la misma secuencia de creación de objetos y envío de mensajes que se llevan a cabo en el proceso para agregar al sistema un nuevo patrón correspondiente a un gesto físico, descritos anteriormente, hasta la obtención de un patrón del comando físico que se está recibiendo como entrada.

De forma similar al proceso de entrenamiento de gestos físicos, se crea un objeto *Preprocess* que es responsable de enviar los mensajes para obtener la señal de aceleración total y para realizar el suavizado de la señal obtenida .

Cuando se obtiene el patrón correspondiente al comando físco recibido (*pattern*), se crea una instancia de la clase *Recognition* para invocar el mensaje *physicalGestureRecognition*, enviando como parámetro dicho patrón. Este último objeto a su vez utiliza la instancia de la clase *Algorithms* para la ejecución del algoritmo encargado de realizar la comparación entre el patrón actual y los que se encuentran almacenados en el sistema. Esto se realiza a través del mensaje *DynamicTimeWarping*, el cual recibe el patrón actual y retorna el patrón reconocido como el más similar al actual.

Finalmente, se crea una instancia de la clase *ServiceManager* y se envía el mensaje *launchService* para realizar la ejecución del servicio relacionado al patrón reconocido.

3.7.5 Agregar en la configuración del sistema un comando de gesto táctil

Por otra parte, la figura 3.9 muestra el diagrama de secuencia diseñado para representar el proceso de agregar en la configuración del sistema un comando de gesto táctil para una aplicación en específico, equivalente al almacenamiento de un patrón de comando de gesto táctil.

El diagrama representa este procedimiento de una manera más abstracta en comparación con los diagramas anteriores, debido a que se planea utilizar una interfaz de programación de aplicaciones (*API* por sus siglas en inglés) que proporciona el lenguaje de programación para Android, el cual se utilizará para el desarrollo del sistema multimodal.

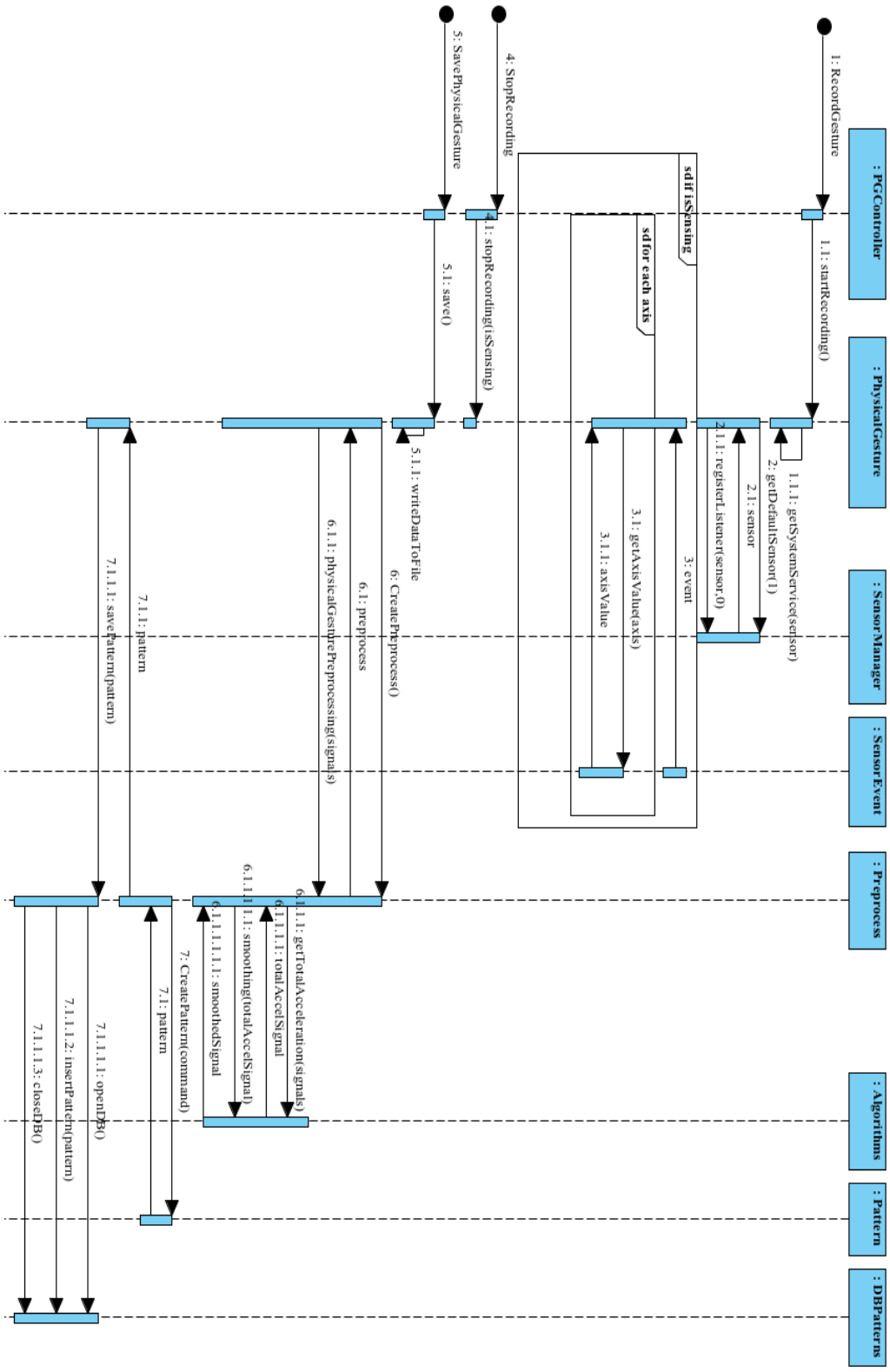


Figura 3.8: Diagrama de secuencia para reconocer un gesto físico

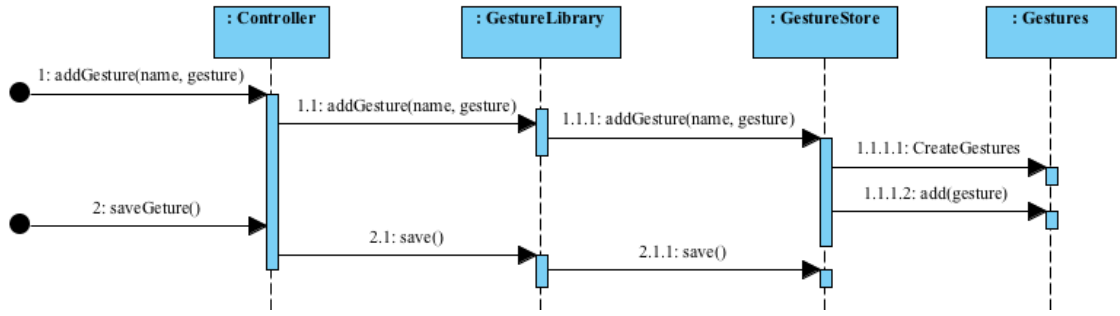


Figura 3.9: Diagrama de secuencia para almacenar un patrón de un gesto táctil

El proceso de agregar un gesto táctil al sistema, inicia con la recepción del mensaje *addGesture* en el objeto *Controller*, este mensaje se invoca cuando el usuario dibuja un gesto sobre la pantalla táctil del dispositivo móvil.

Al recibir el mensaje, el objeto *Controller* crea una instancia de la clase *GestureLibrary*, que a su vez delega la responsabilidad de agregar el gesto definido por el usario al objeto *GestureStore*.

Para agregar el gesto, el objeto *GestureStore* crea una colección de objetos *Gestures* a la cual agrega, a través del mensaje *add*, el gesto recibido (*gesture*).

En caso de que el usuario decida almacenar el gesto táctil, el controlador recibe el mensaje *saveGesture* y delega la responsabilidad de guardar el gesto al objeto *GestureLibrary* y éste a su vez al objeto *GestureStore*.

El objeto *GestureStore*, al recibir el mensaje *save* procede a almacenar en un archivo la definición del gesto táctil.

Cada una de las definiciones de gestos táctiles contenidas en el archivo corresponden a un patrón de gestos táctiles en el sistema multimodal.

3.7.6 Reconocer un comando de gesto táctil

La figura 3.10 presenta el diagrama de secuencia diseñado para ilustrar el proceso de reconocer un comando de gesto táctil.

Al igual que el proceso de agregar un comando de gesto táctil al sistema, el proceso de reconocimiento utilizará el *API* mencionada. Este proceso inicia cuando el controlador *Controller,* encargado de manejar los eventos relacionados con el módulo de gesto táctiles, recibe el mensaje *recognize* en el momento que el usuario dibuja sobre la pantalla táctil un gesto (*gesture*).

El objeto *Controller* crea una instacia de *GestureLibrary* y propaga el envío del mensaje *recognize* al objeto *GestureStore*.

Cuando el objeto *GestureStore* recibe como entrada el gesto a reconocer, a través del

mensaje *recognize*, crea una instancia de la clase *Learner* para invocar el mensaje *classify*, encargado de realizar una clasificación de esta entrada y predecir el gesto que sea más similar al gesto que se recibió, haciendo una comparación con los gestos que se encuentran almacenados en el sistema.

Para realizar lo anterior, la clase *Learner* crea la colección *Predictions* que contiene el resultado de la predicción, el cual es enviado a través de los diferentes objetos hasta el controlador.

Al recibir la predicción, el objeto *Controller* crea un objeto del tipo *ServiceManager* y envía el mensaje *launchService* para realizar la ejecución del servicio relacionado al gesto táctil reconocido.

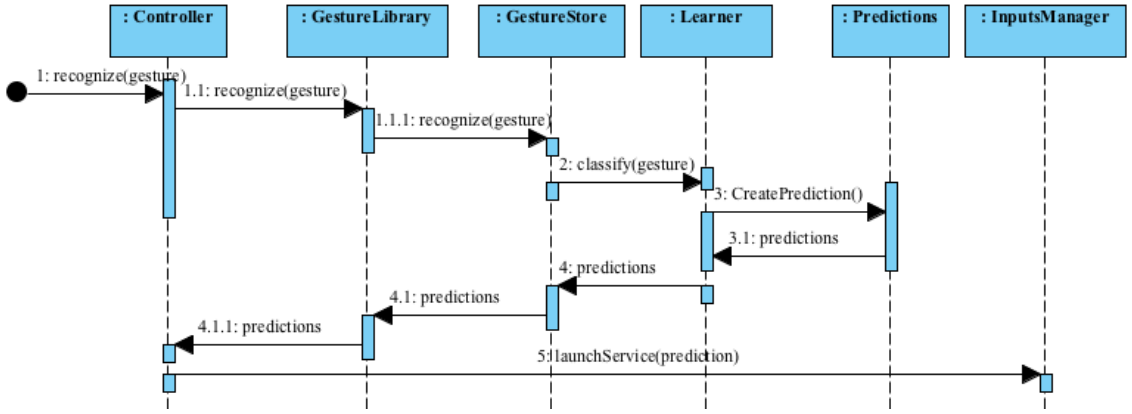


Figura 3.10: Diagrama de secuencia para reconocer un gesto táctil

Capítulo 4

Desarrollo del proyecto

El presente capítulo describe el proceso de desarrollo del proyecto. En las siguientes secciones se presenta la manera en que fueron implementados cada uno de los módulos y procesos descritos anteriormente, tomando en cuenta el diseño de los mismos.

El desarrollo del proyecto está pensado para dispositivos móviles con sistema operativo Android, por lo cual el lenguaje de programación utilizado en la implementación es Java.

Actualmente existen un gran número de dispositivos móviles en el mercado que incorporan estas tecnologías; este hecho y las características presentadas en la sección 2.2.1., son los principales motivos de su uso en el desarrollo de este proyecto.

4.1 Módulo de voz

La implementación de un módulo de reconocimiento de comandos de voz, implica el desarrollo de dos etapas muy bien diferencias: preprocesamiento y extracción de características de la señal de audio adquirida a través de un micrófono.

En el caso de que el reconocimiento sea dependiente del hablante, se requiere llevar a cabo una etapa previa de entrenamiento del sistema, que permita la adquisición de patrones representativos de los diferentes comandos (palabras) que se deseen reconocer.

Para la etapa de reconocimiento se utiliza, al final del proceso, algún algoritmo de reconocimiento de patrones que permita comparar la palabra que se desea reconocer con los patrones almacenados en el sistema durante la etapa de entrenamiento y de esta manera determinar el comando más similar.

Los procesos de entrenamiento y reconocimiento de voz se presenta en la figura 4.1. En las siguientes subsecciónes se describe a detalle cada una de las etapas de estos procesos.

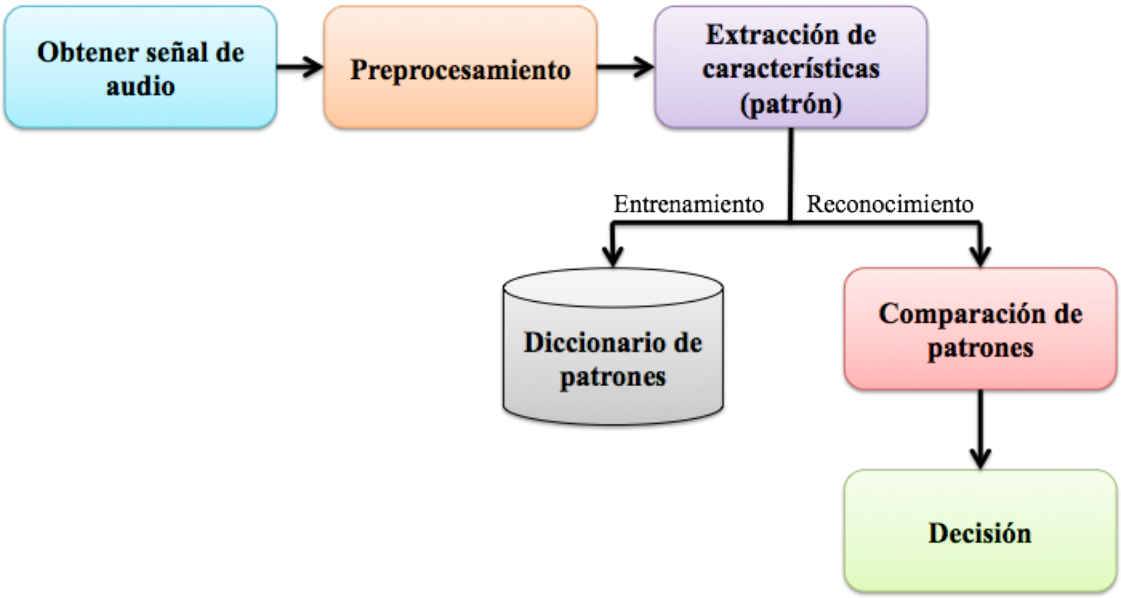


Figura 4.1: Procesos de entrenamiento y reconocimiento de comandos de voz

4.1.1 Adquisición de la señal de audio

La adquisición de la señal de audio se lleva a cabo utilizando el micrófono del dispositivo móvil. El micrófono es un sensor que se encarga de convertir el sonido a una señal eléctrica.

Para adquirir una señal digital de voz es necesario realizar la conversión de la señal analógica del micrófono a una señal digital, para esto se utiliza una tarjeta de sonido con conversor analógico a digital (A/D). Mientras el micrófono esté en operación, las ondas de sonido provocan una vibración en el elemento magnético que lo conforma, causando una corriente eléctrica que pasa a la tarjeta de sonido. El conversor A/D se encarga de grabar cada determinado tiempo los voltajes eléctricos producidos.

Android proporciona la clase `AudioRecord`, encargada de gestionar los recursos necesarios para la grabación de audio, utilizando el hadware de entrada de audio disponible en la plataforma.

Al crear un objeto de la clase `AudioRecord` es necesario especificar los siguientes parámetros:

- Fuente de entrada de audio: indica el origen del audio. La clase `MediaRecorder` establece diversas constantes asociadas a fuentes de entrada de audio que se pueden utilizar para este parámetro. Entre las constantes está la relacionada al micrófono, `MediaRecorder.AudioSource.MIC`, la cual se utilizó como parámetro.

- Frecuencia de muestreo: es un valor expresado en *Hertz*, éste señala qué tan seguido

se graban los valores de voltaje registrados por el micrófono. Android predefine los valores de 44100, 22050, 16000 y 11025 Hz, para ser utilizados como frecuencia de muestreo. Para este caso se seleccionó el valor de 16000 Hz, ya que de acuerdo a la literatura es la frecuencia que usan la mayoría de las aplicaciones de reconocimiento de voz, debido a que el habla está en un ancho de banda relativamente bajo (en su mayoría entre 100Hz-8kHz) [34, 35, 36].

- Configuración de canal: describe la configuración de los canales de audio. Para el fin de la aplicación, la configuración en un sólo canal es la adecuada. La clase `AudioFormat` declara un conjunto de constantes que definen diversos formatos de audio y configuraciones de canales para la grabación de audio. La declaración de una variable para configurar un canal de audio (mono) es la siguiente: `AudioFormat.CHANNEL_IN_MONO`.

- Formato de audio: representa el formato en el cual los datos de audio son representados. Comúnmente se utiliza una representación de 8 o 16 bits por muestra. Las contastes `ENCODING_PCM_16BIT` y `ENCODING_PCM_8BIT`, se encuentran definidas en la clase `AudioFormat` para especificar el tipo de representación. En esta implementación se utilizó una representación de 16 bits.

- Tamaño de búfer: es un valor expresado en bytes que representa el tamaño total del búfer donde los datos de audio son escritos durante la grabación. Estos datos se pueden leer por partes, accediendo a una cantidad de datos menor que el tamaño definido para el búfer. Para calcular el tamaño mínimo de búfer requerido se recomienda utilizar el método `getMinBufferSize` de la clase `AudioRecord`.

Una vez creado el objeto de tipo `AudioRecord` se invoca el método `startRecording` para iniciar la grabación de audio. Como se mencionó, los datos de audio son escritos en un búfer durante la grabación, por lo que, para conservarlos se requiere escribir estos datos a un archivo. Esta tarea se realiza utilizando un hilo de ejecución, que accede a los datos del búfer mientras ocurre la grabación, a través del método `read`.

Cuando se desea detener la grabación se llama el método `stop` y `release` del objeto `AudioRecord`, para terminar la grabación y liberar los recursos solicitados en un inicio, respectivamente.

En la figura 4.2 se puede observar una gráfica que representa una señal de audio adquirida utilizando el procedimiento descrito anteriormente.

4.1.2 Preprocesamiento

La etapa de preprocesamiento es realmente importante en el proceso de tratamiento de señales de audio, ya que por su naturaleza una señal de audio grabada presenta silencio y

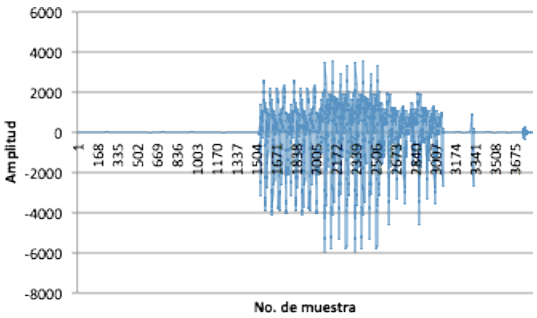


Figura 4.2: Señal de audio adquirida por el micrófono

ruido de fondo. Al final de esta etapa se obtiene una señal que permite realizar una mejor extracción de características.

El proceso que se estableció para esta etapa, consiste primeramente en remover de la señal de audio grabada aquellos valores que representan silencio, dicho de otra manera, obtener los valores que sólo representan voz. Posteriormente, aplicar un filtro preénfasis a la señal y finalmente, dividir la señal filtrada en intervalos que se translapen.

La figura 4.3 representa esta etapa y en la siguientes subsecciones se describe a detalle el procedimiento establecido.

Figura 4.3: Etapas del preprocesamiento de la señal de voz

4.1.2.1 Detección de la voz

Detectar apropiadamente las secciones de la señal de audio que representan voz puede reducir significativamente el procesamiento requerido posteriormente.

En la implementación se utilizó el algoritmo presentado por Saha et al., que utiliza propiedades estadísticas del ruido para detectar la voz en la señal de audio [37].

De acuerdo con los autores del algoritmo, si se toman entre 5 y 100 milisegundos (ms) de una señal de voz, se puede notar que sus características varían lentamente en el tiempo. También, usualmente debido a que transcurre cierto tiempo entre el momento en que se inicia la grabación y cuando el usuario pronuncia algo, los primeros 200 milisegundos o más de una grabación de voz, corresponden a silencio o ruido de fondo.

El algoritmo se basa en estas propiedades básicas de una señal de voz y en las de la distribución normal o Gaussiana. La función de densidad de la distribución normal, se describe con la curva que se conoce como la campana de Gauss. La distribución de

una variable normal está completamente determinada por dos parámetros, su media y su desviación estándar, denotadas generalmente por μ y σ^2.

Entre las propiedades de la distribución normal están el ser simétrica con respecto a su media μ, y normalmente, los valores tienden a agruparse alrededor de la media. Esta última propiedad tiende a obedecer las siguientes probabilidades (ver figura 4.4) :

$$Pr\left[|x-\mu| \leq \sigma\right] \approx 0.68$$

$$Pr\left[|x-\mu| \leq 2\sigma\right] \approx 0.95$$

$$Pr\left[|x-\mu| \leq 3\sigma\right] \approx 0.997$$

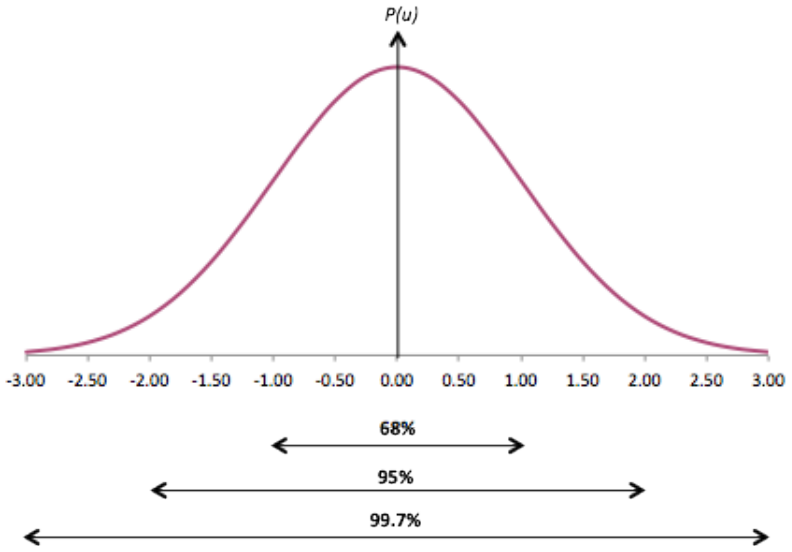


Figura 4.4: Distribución de probabilidad alrededor de la media

El algoritmo para la detección de voz implementado está descrito a continuación:

- Calcular la media y desviación estándar de las primeras muestras correspondientes a 100 ms.

$$\mu = \frac{1}{1600}\sum_{i=1}^{1600} x(i) \quad \sigma = \sqrt{\frac{1}{1600}\sum_{i=1}^{1600}(x(i)-\mu)^2}$$

  El algortimo original trabaja sobre los primeros 200 ms., pero de acuerdo a observaciones que realizamos durante el desarrollo de este proyecto, utilizando este valor en la aplicación, se determinó que 100 ms es el valor adecuado para nuestro caso.

- Para cada una de las muestras de la señal grabada, checar si la distancia Mahalanobis[1] [38] es mayor o igual a 3:

$$if, \frac{\mid x - \mu \mid}{\sigma} \geq 3$$

  Si sí, entonces la muestra es tratada como voz.

  Si no, la muestra es tratada como silencio o ruido de fondo.

- Marcar la muestra de voz como 1 y la muestra sin voz como 0.
- Dividir la señal completa en ventanas sin traslapes de 10 ms.
- Considere que existen M ceros y N unos en la ventana. Si $M \geq N$ entonces marcar como ventana sin voz, en caso contrario como ventana con voz.
- Unir las ventanas con voz correspondientes a las muestras etiquetadas con '1' .

En la figura 4.5 se puede observar de lado izquierdo una señal de voz grabada y de lado derecho la señal obtenida después de aplicar el algoritmo de detección de voz. Es posible apreciar a simple vista la sección de la señal donde se encuentra presente la voz.

El número de muestras se reduce significativamente al realizar este procedimiento.

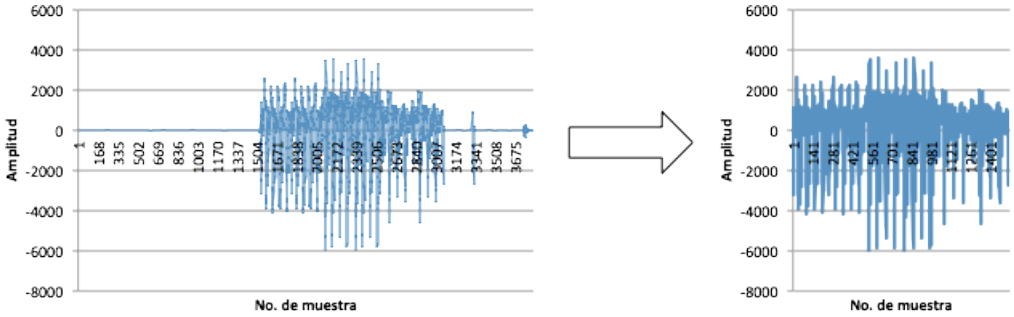


Figura 4.5: Señal obtenida despues de la detección de voz

4.1.2.2 Filtro preénfasis

El filtro preénfasis se aplica a una señal de voz dado que por naturaleza presenta una atenuación en las frecuencias altas, es por esto que se debe realizar un filtro que permita obtener información suficiente de esas frecuencias, para no concentrarse únicamente con

[1]La distancia de Mahalanobis es una medida de distancia desde x a la media μ, dada en unidades de desviación estándar.

la información de las frecuencias bajas. Además es una operación realizada sobre una señal para hacerla menos suceptible a truncamientos.

La ecuación del filtro implementado está dada por:

$$y[n] = x[n] - \alpha[n-1]$$

Donde:

- $x[n]$ es el vector de amplitud de la señal de voz de entrada
- $y[n]$ es la salida del filtro preénfasis
- Si $\alpha < 0$, tenemos un filtro pasa bajas
- Si $\alpha > 0$, tenemos un filtro pasa altas

  En nuestro caso $\alpha = 0.97$

La figura 4.6 muestra un ejemplo de señal antes y después de aplicar el filtro preénfasis.

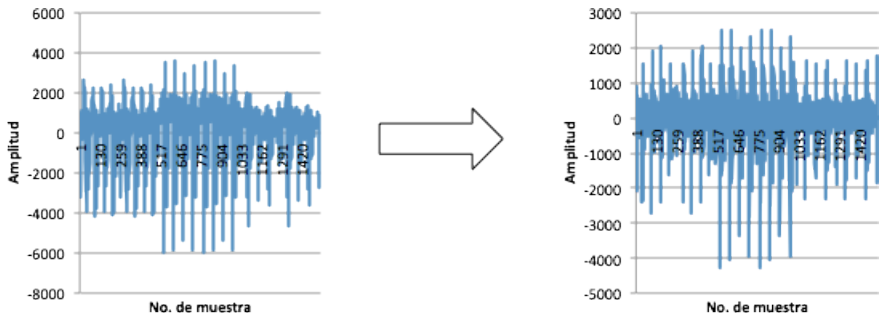


Figura 4.6: Señal obtenida después de aplicar el filtro preénfasis

4.1.2.3 Ventaneo

El proceso de extracción de características se realiza en intervalos cortos de tiempo, comúnmente los estudios de voz se realizan a intervalos de 20ms y 30 ms, que es donde se considera que la onda de voz no presenta demasiados cambios.

La señal completa de audio se divide en bloques, el tamaño de cada bloque se conoce como tamaño de ventana. Para realizar un análisis adecuado de la señal, es necesario que los bloques se traslapen, de aquí surgen los valores de incremento y solapamiento. La figura 4.7 representa gráficamente estos conceptos.

Los valores adoptados durante la implementación corresponden a los siguientes:

- Tamaño de ventana: 400 muestras = 25 ms
- Incremento: 50 %
- Solapamiento: 50 %

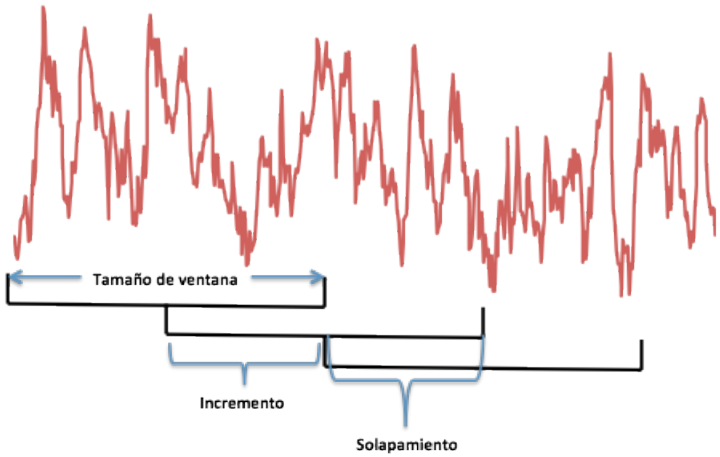


Figure 4.7: Ventaneo de una señal

A cada bloque se le asigna una función, conocida como función de ventana, con el fin de disminuir la importancia de los valores que se encuentran en los extremos de los bloques; esto se realiza para evitar que las características de estos valores alteren la interpretación de la parte central del bloque, que es donde se encuentran los valores más significativos.

La funcion de ventana más utilizada en aplicaciones de procesamiento de voz, es la ventana de Hamming [39] definida a continuación:

$$V(n) = 0.54 - 0.46Cos\frac{2\pi n}{N}$$

Donde:

- N es el número total de muestras de la ventana.
- n es la muestra actual.

La figura 4.8 muestra la gráfica asociada a la función de ventana de Hamming.

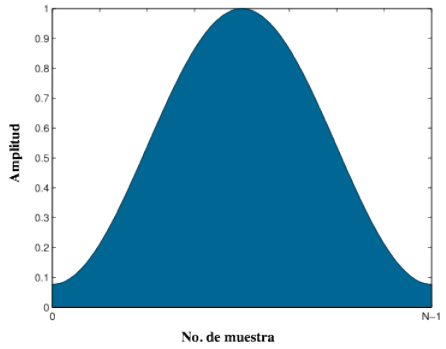


Figura 4.8: Ventana de Hamming

4.1.3 Extracción de características

En la etapa de preprocesamiento la señal de voz ha sido preparada para extraer sus características más representativas, es decir, obtener la información principal de cada bloque de la señal de voz.

Existen una gran variedad de algoritmos que se utilizan para la extracción de características de una señal de voz, algunos trabajan sobre la señal en dominio del tiempo y otros sobre el dominio de la frecuencia. En este caso, se requiere seleccionar algún algoritmo que trabaje sobre el dominio del tiempo, para que no sea necesario realizar procedimientos computacionales adicionales para convertir la señal obtenida (que está en el dominio de tiempo), al dominio de la frecuencia.

Se ha seleccionado el algoritmo de codificación por predicción lineal (LPC por sus siglas en inglés) [40], para su implementación por su sencillez y rapidez de ejecución. Además de ser una técnica que trabaja sobre el dominio del tiempo, se basa en la producción del habla y proporciona la información más representativa de la voz.

El concepto en el que se basa LPC es en el que una muestra de una señal de voz $x\,(n)$, puede ser predicha por la k muestras anteriores de la misma señal, generando una señal aproximada $\tilde{x}(n)$:

$$\tilde{x}(n) = \sum_{i=1}^{k} a_i * x(n-i)$$

La implementación del algoritmo LPC implica definir el parámetro de orden de la codificación; este valor indica al algoritmo cuantos coeficientes de codificación se desean obtener de cada bloque de la señal. El valor seleccionado para este parámetro corresponde a 10, dado que es un valor recomendado en la literatura.

El conjunto de valores obtenidos al final como la codificación completa de la señal, corresponde al patrón asociado al comando de voz adquirido a través del micrófono. Este patrón es almacenado en un archivo y su ubicación es registrada en una base de datos que puede ser vista como un diccionario de patrones.

4.1.4 Algoritmo de comparación de patrones

Se implemento el algoritmo DTW (*Dynamic Time Warping*) [41, 42], para realizar la tarea de comparación de patrones. El patrón obtenido de una señal de voz capturada, que corresponde a una palabra pronunciada por el usuario para su reconocimiento, se compara con los patrones previamente almacenados en el diccionario de patrones para determinar cual es el más similar al primero.

El algoritmo DTW permite medir la similaridad entre dos secuencias, en este caso entre las secuencias de valores obtenidas a través del algoritmo LPC para cada uno de los patrones.

La ventaja que presenta este algoritmo, es que las secuencias a comparar pueden varian en tiempo y velocidad. Por esto es conveniente utilizar DTW en el contexto de reconocimiento de voz, ya que pueden existir dos patrones que correspondan a la misma palabra pero con diferente longitud, debido a que el usuario pronunció la misma palabra a diferente velocidad.

El pseudocódigo del algoritmo es el que se presenta a continuación:

Algoritmo 4.1 Pseudocódigo del algoritmo DTW

```
Entradas: Secuencia del patrón almacenado A
          Secuencia del patrón a comparar C
Salida:   Distancia mínima entre A y C
DynamicTimeWarping(A, C){
  n = longitud de A;
  m = longitud de C;
  dtw [m+1][n+1];
  costo = 0;
  Desde i = 1 hasta i < m con pasos de 1 hágase{
    dtw [0][i] = infinito;}
  Desde j= 1 hasta j < n con pasos de 1 hágase{
    dtw [j][0] = infinito;}
  dtw[0][0]=0;
  Desde i = 1 hasta i <= n con pasos de 1 hágase{
    Desde j = 1 hasta j <= m con pasos de 1 hágase{
      costo = | C[i-1] - A[j-1] |;
      dtw [i][j] = costo +
      mínimo ( mínimo (dtw[i-1][j], dtw [i][j-1]),
               dtw [i-1][j-1]);}}
  dése como resultado dtw [n][m];
}
```

4.2 Módulo de gestos físicos

El módulo de gestos físicos es el encargado de procesar las entradas que se asocian a gestos definidos por movientos del dispositivo móvil.

Obtiene los valores registrados por los ejes inerciales (x, y, z) del acelerómetro durante un cierto período determinado por el usuario, al iniciar y terminar de grabar un gesto físico.

Posteriormente, procede a realizar un preprocesamiento de estos valores para obtener la señal de aceleración y después aplicar sobre ella una función de suavizado.

El resultado del procedimiento de suavizado representa el patrón asociado al gesto físico de entrada y es almacenado en una base de datos, en caso de tratarse del proceso de entrenamiento.

Si se desea reconocer algún gesto físico, se requiere utilizar un algortimo de comparación de patrones que permita determinar a qué patrón almacenado anteriormente en el sistema está asociado.

El proceso de entrenamiento y reconocimiento de un comado de gesto físico se presenta en la figura 4.9. En las siguientes subsecciónes se describe a detalle cada una de los procedimientos que permiten llevar a cabo estos procesos.

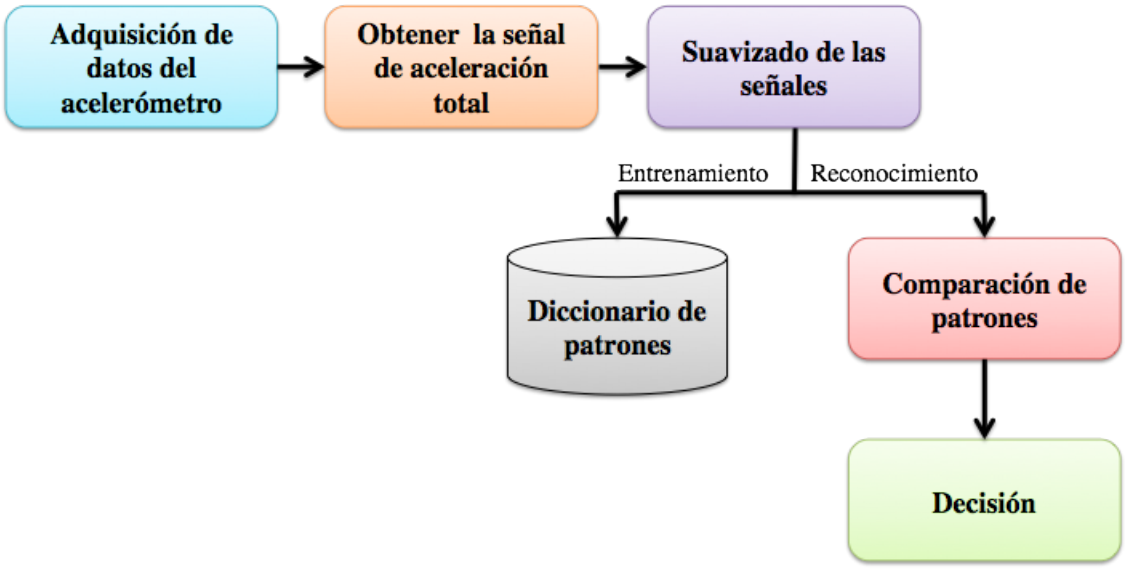


Figura 4.9: Proceso de entrenamiento y reconocimiento de comandos de gestos físicos

4.2.1 Adquisición de datos de acelerómetro

Un gesto físico está definido a través del movimiento descrito por el dispositivo móvil en un determinado tiempo. La aplicación utiliza el acelerómetro con el que cuenta el dispositivo móvil, para registrar los valores asociados al gesto físico.

El acelerómetro registra los valores en los ejes, x, y, z del dispositivo. En la figura 4.10 se puede observar los ejes con respecto al dispositivo móvil.

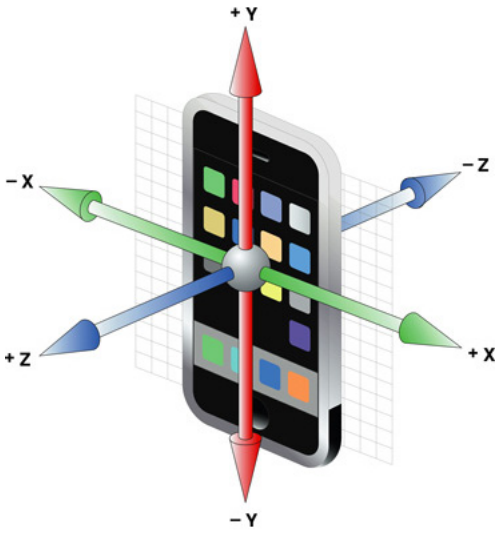


Figura 4.10: Ejes inerciales del acelerómetro con respecto al dispositivo móvil

Android proporciona la clase `SensorManager` para el manejo de los sensores disponibles en el dispositivo móvil. Entre los sensores se encuentra el acelerómetro, giroscopio, sensor de orientación y sensor de campo magnético.

Para adquirir los valores registrados por el acelerómetro en Android es necesario realizar los siguientes pasos:

- Crear una instancia de la clase `SensorManager`. Esto se realiza a través del envío del mensaje `getSystemService` para acceder al servicio del sistema que maneja los sensores. Utilizando la instrucción `getSystemService(SENSOR_SERVICE)` se obtiene la instancia de la clase `SensorManager`.

- Definir el sensor. Android dispone de la clase `Sensor` que define diferentes constantes asociadas a los sensores soportados. En este caso se requiere obtener la instancia del sensor acelerómetro, esto se realiza enviando el mensaje `getDefaultSensor` con el valor de la constante asociada al acelerómetro como parámetro, de esta manera: `getDefaultSensor(Sensor.TYPE_ACCELEROMETER)`.

- Registrar el evento que está a la escucha de cambios en el acelerómetro. Una vez adquirida la instancia del sensor, es necesario estar a la escucha de los cambios de valor que registre el acelerómetro a través del método `registerListener`. Este método recibe como parámetros el objeto tipo `Sensor` y la frecuencia de muestreo sugerida al sistema para obtener los valores registrados por el acelerómetro.

Cada cierto tiempo el sistema trata de acceder a los valores que registra el acelerómentro; cuando se produce un cambio en ellos, se crea un objeto de tipo `SensorEvent` que abstrae la información obtenida del sensor al momento de su creación, al cual se puede acceder por medio del método `onSensorChanged`.

El objeto `SensorEvent` encapsula la variable `values`, la cual es un arreglo de valores flotantes con longitud de 3 para el caso del sensor acelerómetro. Los valores registrados en los ejes x, y, z, al momento del evento, se encuentran almancenados en ese orden en el arreglo. Cada uno de ellos se accede y se escriben en un arreglo temporal para su uso posterior.

En la figura 4.11 se representa gráficamente las señales de los ejes x, y, z obtenidas del acelerómetro al dibujar un gesto físico en forma circular.

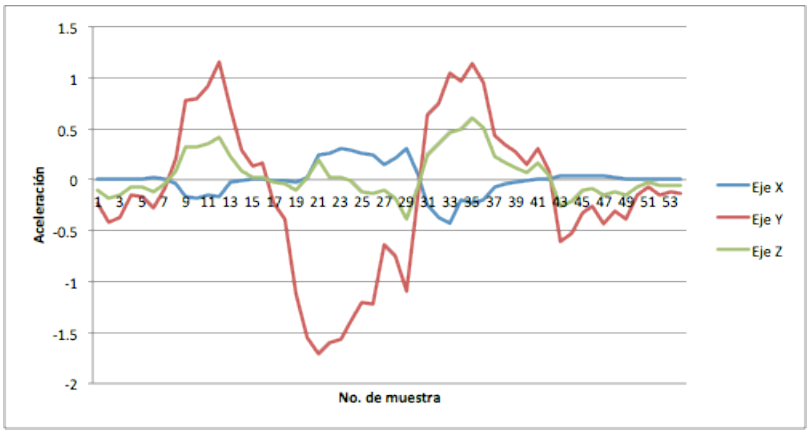


Figura 4.11: Señales de los ejes x, y, z obtenidas del acelerómetro

4.2.2 Obtener la señal de aceleración total

El conjunto de valores registrados conforman la señales representativas de los cambios de aceleración en los ejes x, y z del dispositivo; a partir de estos valores se procede a calcular la magnitud del vector de aceleración, utilizando la fórmula:

$$\bar{a} = \sqrt{x^2 + y^2 + z^2}$$

Este procedimiento nos permite obtener una sola señal que es representativa de los valores obtenidos. Realizar el cálculo de la señal de aceleración permite reducir cómputo requerido en etapas posteriores ya que se maneja sólo un conjunto de valores en lugar de tres diferentes.

4.2.3 Suavizado de señal

Como se mencionó con anterioridad, al capturar una señal es común que ésta contega ruido. Es por esto que se requiere aplicar un algoritmo de suavizado sobre la señal de aceleración calculada.

Se implementó el algoritmo de suavizado de promedio móvil. Este algoritmo reemplaza cada punto en la señal con el promedio de sus m puntos adyacentes, donde m es un entero impar positivo llamado ancho de suavizado.

Dicho de manera matemática, si tenemos un arreglo de datos $Y = [y_1, y_2, ..., y_n]$, puede ser convertido a un nuevo arreglo de datos suavizados. El punto suavizado $(y_k)_s$ es el promedio de un número impar, $2n + 1$, de puntos consecutivos del arreglo de datos Y $(y_{k-n}, y_{k-n+1}, ..., y_{k-1}, y_k, y_{k+1}, ..., y_{k+n-1}, y_{k+n})$.

$$(y_k)_s = \sum_{i=-n}^{i=n} y_{k+i} / \ (2n+1)$$

La implementación se realizó utilizando un ancho de suavizado con valor de 5.

Los valores que definen a la señal obtenida después del proceso de suavizado, representan el patrón del gesto físico adquirido en un inicio. Este patrón es almacenado en un archivo y la referencia de su ubicación se almacena en el diccionario de patrones para este tipo de comando.

4.2.4 Algoritmo de comparación de patrones

El algoritmo implementado para comparar los patrones de gestos físicos, corresponde esencialmente al mismo utilizado para reconocer comandos de voz, descrito en la sección 4.1.4, el algoritmo *Dynamic Time Warping (DTW)*.

La diferencia es que la comparación se realiza utilizando el patrón de gesto físico que se desea reconocer y el conjunto de patrones de gestos físicos que se encuentran almacenados en el diccionario de patrones de este tipo de comandos.

El algoritmo DTW sigue siendo útil para reconocer gestos físicos debido a que el usuario puede definir un mismo gesto físico a diferente velocidad.

4.3 Módulo de gestos táctiles

El módulo de gestos táctiles es el responsable de procesar los gestos dibujados sobre la pantalla táctil del dispositivo móvil.

En la implementación de este módulo se ha utilizado el API `android.gesture`, disponible desde la versión 1.6 de Android, para el manejo de comandos tipo gestos táctiles.

Este API permite almacenar, cargar, dibujar y reconocer gestos táctiles. En las siguientes subsecciones se explica detalladamente la forma en que se lleva a cabo cada uno de los procedimientos.

4.3.1 Dibujar un gesto táctil

Android proporciona una vista especial para poder dibujar gestos sobre la pantalla táctil. Esta vista es la definida por `GestureOverlayView`.

En el diseño de la interfaz gráfica, la `GestureOverlayView` puede ser utilizada como una vista normal, dentro de un layout o puede estar por encima de otra vistas.

La vista `GestureOverlayView` no forma parte del paquete usual de elementos gráficos, `android.widget`, por lo que es necesario utilizar su nombre completo al momento de su declaración: `android.gesture.GestureOverlayView`.

La `GestureOverlayView` puede ser vista como un pizarrón, donde el usuario puede dibujar sus propios gestos. Cuenta con diversas propiedades como son el color y ancho de los trazos que describen los gestos y el poder estar a la escucha de lo que el usuario hace sobre su superficie.

Un *listener* que se implementó para esta vista es `onGestureListener` que permite acceder a los métodos `onGestureStarted` y `onGestureEnded`, lanzados cuando el usuario inicia a dibujar un gesto sobre la vista y cuando termina de hacerlo, respectivamente. En específico, dentro del método `onGestureEnded` se obtiene el objeto de tipo `Gesture`, que define al gesto dibujado, al invocar el mensaje `getGesture`.

La figura 4.12 muestra un ejemplo de uso de la `GestureOverlayView`.

4.3.2 Crear y cargar una biblioteca de gestos táctiles

El API de gestos provee los métodos requeridos para crear una biblioteca de gestos táctiles, así mismo para cargarla dentro de la aplicación para su uso.

La clase `GestureLibrary` abstrae los elementos que conforman una biblioteca de gestos táctiles. La figura 4.13 muestra el diagrama de la estructura de una biblioteca de este tipo.

Para obtener una instancia de tipo `GestureLibrary` es necesario utilizar alguno de los métodos definidos en la clase `GestureLibraries` para este fin. El método utilizado

Figura 4.12: Ejemplo de la GestureOverlayView

en la implementación es `GestureLibraries.fromFile`, el cual permite especificar el archivo que almacena la biblioteca o su ubicación en el dispositivo. En caso de que el archivo no exista se crea.

Si se desea realizar operaciones sobre la biblioteca de gestos, tales como acceder a algún elemento, agregar, eliminar, etc., se requiere invocar el método `load` utilizando el objeto `GestureLibrary`, para cargar la biblioteca y sus elementos (en caso de tener) a memoria.

Cuando el usuario desea agregar un nuevo comando de gesto táctil en la aplicación, se utiliza el método `addGesture` para agregar un nuevo gesto táctil a la biblioteca. Este método recibe como parámetros el nombre asociado al gesto y el objeto `Gesture` que lo define. Adicionalmente, se invoca el metodo `save` para almacenar en archivo el estado actual de la biblioteca.

La biblioteca de gestos puede se vista como un diccionario de patrones de gestos tactiles, su ubicación se almacena en la base de datos de la aplicación.

4.3.3 Reconocer un gesto táctil

En la etapa de reconocimiento de un gesto táctil se utiliza una `GestureOverlayView`, a la cual se asocia el `GesturePerformedListener` dedicado a estar a la escucha de cuándo el usario termina de dibujar el gesto que desea sea reconocido. En el momento en que esto ocurre, se invoca el método `onGesturePerformed` en el cual se ha implementado la funcionalidad de reconocimiento.

La clase `GestureLibrary` permite acceder al método `recognize`, que recibe como

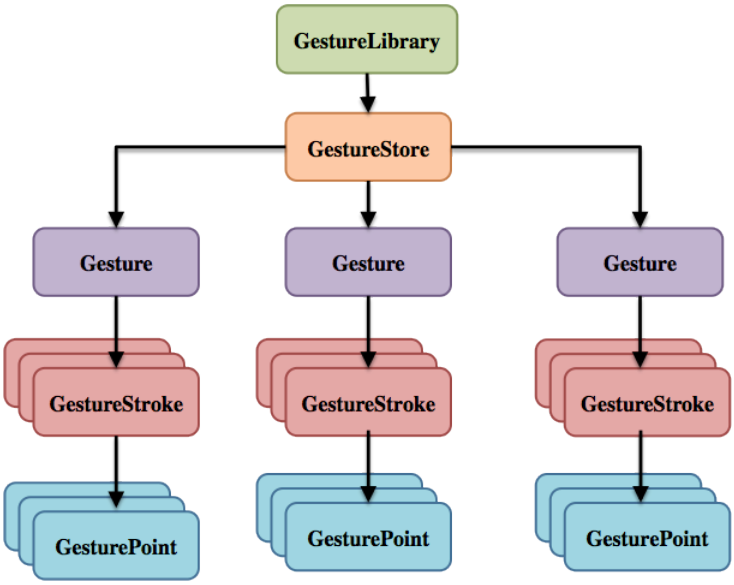


Figura 4.13: Estructura de una biblioteca de gestos táctiles

parámetro el gesto a reconocer y regresa como resultado un *ArrayList* de objetos de tipo `Prediction`. Estos objetos son las predicciones obtenidas por el algoritmo de reconocimiento que implementa el API de gestos. Según la documentación encontrada, se utilizan los algoritmos de distancia euclideana al cuadrado y distancia coseno.

Cada uno de los elementos de tipo `Prediction` tiene como atributos un nombre, que corresponde al nombre con el cual fue almacenado en la biblioteca, y una puntuación (*score*) asignada por el algoritmo de reconocimiento. Estos elementos se encuentran ordenados en la lista de resultados de manera descendiente de acuerdo a su puntuación. Por esto se obtiene el primer elemento de la lista, el de la más alta puntuación, como el gesto asociado al que se recibió como entrada por parte del usuario.

4.4 Módulo de texto

Este módulo es el encargado de procesar los comandos de texto definidos por el usuario y durante la etapa de reconocimiento, proveer las posibles coincidencias de resultados conforme el usuario ingresa una secuencia de caracteres. La figura 4.14 presenta el proceso de entrenamiento y reconocimiento de un comando de texto.

Para definir un patrón de texto basta con que el usuario ingrese a través de la interfaz gráfica una secuencia de caracteres, dicho de otra manera, un comando de texto asociado a un servicio del dispositivo móvil.

El sistema obtiene la cadena de caracteres a través del método `getText` y se almacena

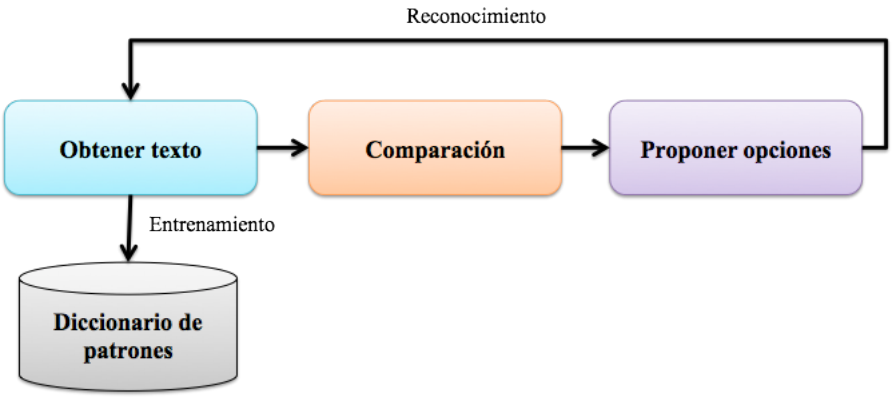


Figura 4.14: Proceso de entrenamiento y reconocimiento de un comando de texto

en la base de datos de la aplicación, así como el nombre del servicio al que está asociada. Esto representa un patrón de comando de texto.

Por otra parte, se implementó el método `onTextChanged` de la interfaz `TextWatcher`, para estar a la escucha de cuándo el usuario desea ingresar a través de la interfaz gráfica, un conjunto de caracteres para su reconocimiento.

Conforme el usuario ingresa o elimina caracteres, se realiza una consulta a la base de datos que contiene los comandos de texto almacenados y se lleva a cabo una comparación ente estos comandos y la cadena de caracteres que hasta el momento se tiene capturada, para proponer los servicios que pueden coincidir con el patrón que se está ingresando.

4.5 Módulo de análisis de contexto

El contexto de la aplicación está definido por la conectividad del dispositivo a la red celular y a las redes inalámbricas.

Para realizar la implementación del módulo de análisis de contexto se definieron las clases `MobileNetwork` y `WiFiNetwork`; estas clases contienen un conjunto de métodos que permiten obtener información del estado actual de la red celular e inalámbrica, respectivamente.

Adicionalmente, se creó la clase `ContextManager` para implementar la función de monitoreo de la red celular e inalámbrica.

En las siguientes subsecciones se presenta a detalle las funcionalidad que provee cada una de las clases mencionadas.

4.5.1 Monitoreo de la red celular

En la implementación de la clase `MobileNetwork`, cuyo objetivo es monitorear el estado de conexión a la red celular, se utilizó un conjunto de clases disponibles en Android para acceder a la información de conectividad del dispositivo y en específico a la de red celular.

Se creó una instancia de la clase `ConnectivityManager` de Android, la cual provee diversos métodos para obtener una vista de alto nivel sobre las conexiones de red disponibles. La instancia se obtiene al invocar el método `getSystemService`, pasando como parámetro el tipo de servicio del que se desea obtener un manejador, en este caso, el servicio de conectividad. Existe la constante `CONNECTIVITY_SERVICE` que se puede utilizar para este fin.

A través del método `getNetworkInfo`, se obtuvo el objeto `NetworkInfo` que abstrae la información sobre la red especificada por medio del parámetro de tipo de red. En el caso de la red celular, se pasa como parámetro la constante `TYPE.MOBILE`, definida por la clase `ConnectivityManager`.

Los métodos implementados en la calse `MobileNetwork` son los siguientes:

- *isMobileNetworkConnected*. Este método es el encargado de verficar si el dispositivo móvil está conectado a la red celular. Esta función se realiza utilizando el objeto `NetworkInfo` y accediendo al método `getState`; en caso de que el valor devuelto por este método corresponda al de la constante `CONNECTED`, se regresa el valor booleano verdadero, en otro caso falso.

- *getMobileNetworkInfo*. Método que retorna el objeto `NetworkInfo`.

- *getMobileNetworkState*. Retorna un objeto tipo `NetworkInfo.State` que contiene la información sobre el estado de conexión a la red celular.

- *getMobileNetworkDetailedState*. Este método da como resultado un objeto de la clase `NetworkInfo.DetailedState` con la información a detalle del estado actual de la red celular.

- *getSignalStregthLevel*. Este método retorna un valor entero que representa la intensidad de señal de la red celular. Para obtener este valor, se apoya de una clase privada que hereda de la clase `PhoneStateListener`.

  Esta última clase dispone del método `onSignalStrengthsChanged`, por medio del cual es posible acceder al objeto de tipo `SignalStrength`, que abstrae la información sobre la intensidad de señal celular.

La primera acción que se realiza es verificar si el objeto `SignalStrength` corresponde al tipo *GSM*; en caso de ser verdadero se procede a obtener el valor de intensidad celular utilizando la sintaxis:

```
ASU = signalStrength.getGsmSignalStrength(),
```

esta instrucción retorna un valor entero en unidades *ASU* (*Arbitrary Strength Unit*), a partir de este, se calcula el valor de nivel de intensidad de señal con base a los criterios que se presentan en el algoritmo 4.2.

Algoritmo 4.2 Cálculo del nivel de intensidad de señal

```
Si ASU <= 2 ó ASU == 99 entonces:
  nivel_señal = 0;
Si ASU >= 12 entonces:
  nivel_señal = 4;
Si ASU >= 8 entonces:
  nivel_señal = 3;
Si ASU >= 5 entonces:
  nivel_señal = 2;
Si no:
  nivel_señal = 1
```

4.5.2 Monitoreo de las redes inalámbricas

La clase `WiFiNetwork` se implementó para monitorear el estado de la conectividad del dispositivo móvil a redes inalámbricas *WiFi* y *3G*. Para lograr esta tarea, al igual que para el monitoreo de la red celular, se utilizaron algunas de las clases que Android define para esta función.

De igual manera a la que se realizó en la clase `MobileNetwork`, se obtuvó una instancia de la clase `ConnectivityManager`; utilizando esta instancia, se accedió al método `getNetworkInfo`, pasando como parámetro la constante `TYPE_WIFI`, para obtener un objeto de tipo `NetworkInfo` que abstrae la información sobre la red *WiFi*.

Adicionalmente, Android dispone de la clase `WifiManager` que provee el API principal para manejar todos los aspectos de conectividad *WiFi*. Se obtuvó una instancia de esta clase invocando el método `getSystemService`, explicado en la sección anterior, y pasando como parámetro la constante `WIFI_SERVICE`.

En la clase `WiFiNetwork` se implementaron los siguientes métodos:

- *isConnectedWifi.* Método encargado de verificar si el dispositivo móvil está conectado a alguna red inalámbrica *WiFi*. En su implementación se utiliza el objeto creado

de tipo `NetworkInfo` para acceder al método `getState`. Si este último método retorna el valor de la constante `CONNECTED`, se retorna un valor verdadero, en otro caso falso.

- *enableWifi.* Este método es responsable de activar/desactivar la conectividad *WiFi* con base al valor booleano recibido como parámetro. Para esto, invoca el método `setWifiEnable` a través de la instancia `WifiManager`, al que a su vez se pasa el valor recibido.

- *getConfiguredNetworks.* Retorna una lista de objetos `WifiConfiguration` que representan las propiedades de todas las redes *WiFi* configuradas en el dispositivo.

- *getScanResults.* Devuelve una lista de objetos tipo `ScanResult` que abstraen la información de las redes encontradas en la última exploración de puntos de acceso.

- *getOpenNetworks.* Obtiene una lista de objetos `ScanResult` que representan las redes sin clave de seguridad, es decir, redes abiertas. Para realizar esta función utiliza la propiedad `capabilities` del objeto `ScanResult`; si el objeto no cuenta con la descripción de esta propiedad se considera como una red abierta.

- *isAnyPreferedNetworkAvailable.* Método encargado de verificar si existe alguna red configurada anteriormente en el dispositivo, entre las que se encuentran al alcance en el momento de su invocación.

4.5.3 Analizador de contexto

La clase `ContextManager` se creó para proveer los métodos que permiten determinar si el tipo de conexión requerida por la aplicación en un momento determinado está disponible. Para la implementación de la funcionalidad brindada por esta clase, se utilizan instancias de las clases `MobileNetwork` y `WiFiNetwork`.

El contenido de la clase `ContextManager` está conformado por dos principales métodos:

- *isMobileNeteworkAvailable*. Este método tiene como objetivo evaluar si la conexión a la red celular está disponible para su uso, de acuerdo al nivel de intensidad de señal de la red móvil que registra el dispositivo. Devuelve un valor verdadero en caso de que el dispositivo se encuentre conectado a la red celular y ésta, cuente con un nivel mayor o igual a uno. El algoritmo 4.3 es el que implementa este método.

- *isWifiConnectionAvailable*. La finalidad de este método es determinar si el dispositivo está conectado a una red *WiFi*.

Algoritmo 4.3 Evaluar la disponiblidad de conexion a la red celular

```
red_disponible = falso;
si el dispositivo está conectado a la red celular entonces:
  nivel_señal = Obtener el nivel de intensidad de la señal
  si nivel_señal >= 1 entonces:
    red_disponible = verdadero;
dése como resultado esta_disponible;
```

En caso de no estar conectado, revisar si la opción de conexión a red *WiFi* está activada para el dispositivo, para posteriormente intentar establecer una conexión a alguna de las redes configuradas con anterioridad.

Si ninguna de las redes configuradas en el dispositivo, se encuentra al alcance en el momento, procede a intentar crear una conexión utilizando las redes abiertas disponibles, de acuerdo al nivel de intensidad de señal de cada una de ellas.

El procedimiento que implementa este método lo define el siguiente algoritmo 4.4.

Algoritmo 4.4 Determinar si el dispositivo está conectado a una red WiFi

```
wifi_disponible = falso;

si la opción de conexión a una red WiFi no está activada, entonces:
  activar opción de conexión a una red WiFi

si ninguna red configurada está disponible, entonces{
  redes_abiertas = obtener las redes WiFi abiertas
  ordenar redes_abiertas de acuerdo a su nivel de señal
  Por cada red en redes_abiertas{
     intentar conectar
     Si la red está conectada, entonces{
       esta_disponible = verdadero;
       dejar de intentar conectar
     }
  }
}
si no, entonces:
  wifi_disponible = verdadero;

Dése como resultado wifi_disponible;
```

4.6 Módulo de gestión de entradas

El sistema ha sido desarrollado para ser capaz de recibir por parte del usuario comandos de entrada en diferentes modalidades: audio, gesto físico, gesto táctil y texto.

Para activar un servicio en el dispositivo móvil, el usuario puede utilizar un sólo comando o ingresar dos comando de forma consecutiva en un intervalo corto de tiempo. En cualquiera de los casos, de acuerdo a la modalidad de la entrada, ésta pasa a ser procesada por el módulo respectivo para obtener el patrón correspondiente al comando recibido.

La finalidad del módulo de gestión de entradas es ejecutar el servicio asociado al patrón o patrones reconocidos.

Los servicios que pueden ser activados por medio de la aplicación son:

- Ejecutar en el dispositivo móvil la aplicación predeterminada para las funciones de realizar una llamada, enviar un mensaje SMS, navegar en Internet, capturar una fotografía, reproducir música, mostrar la lista de contactos, enviar un correo electrónico y mostrar una calculadora.
- Mostrar la información de un contacto.
- Enviar un mensaje o marcar el número telefónico de un contacto en específico.

Los primeros dos servicios puntualizados en la lista, se ejecutan utilizando una sóla entrada y el último, a través de una combinación de dos entradas.

4.6.1 Ejecutar un servicio con base a la entrada recibida

Una vez que se tiene el patrón asociado a la entrada recibida, se procede a ejecutar el servicio correspondiente. Para esta finalidad se implementó la clase `InputsManager` que recibe el objeto tipo `Pattern` que abstrae la información del patrón reconocido.

Un patrón está ligado a una aplicación o a un contacto, según se haya definido en su etapa de entrenamiento. Con base en esta información, cuando se desea ejecutar el servicio asociado a la entrada se invoca, de acuerdo al caso, alguno de estos métodos definidos en la clase `InputsManager`:

- *processApplication*. Este método se encarga de obtener el nombre de la aplicación asociada al patrón, en caso de ser la aplicación de llamadas o mensajes activa una bandera que indica a la aplicación la posibilidad de que exista una segunda entrada e inicia la ejecución de un proceso `TimerTask` para esperar un tiempo determinado la ocurrencia de una nueva entrada a través de la interfaz gráfica.

  En caso de no recibir una segunda entrada, se verifica si la aplicación que se desea ejecutar requiere de una conexión a la red celular o a una red inalámbrica a través del método *checkApplicationRequirement*.

Si la aplicación no requiere ningún tipo de conexión o la conexión que requiere está disponible, se invoca el método *launchApplication* para que se encargue de ejecutar la aplicación en el dispositivo móvil. En caso de no contar con la conexión requerida, se cancela la ejecución del servicio y se informa al usuario.

El método utilizado para verificar si la conexión que se requiere está disponible es *checkConnection*.

La figura 4.15 representa gráficamente el proceso descrito para ejecutar un servicio.

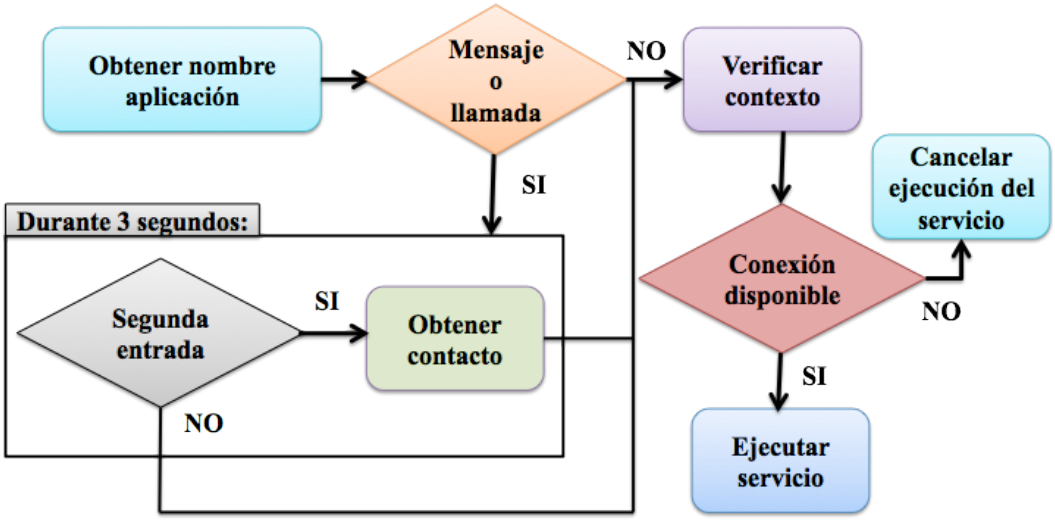


Figura 4.15: Proceso para ejecutar un servicio

- *processContact*. En este método se obtiene el identificador del contacto al que está relacionado el patrón. En caso de que este método se invoque cuándo esté en ejecución el proceso que espera una posible segunda entrada, se invoca el método *executeInputs*.

  En otro caso, se ejecuta el método *showContactInformation* para mostrar la información del contacto con el identificador obtenido en un inicio.

- *checkApplicationRequirement*. Método que recibe el nombre de una aplicación y retorna una constante que indica el tipo de conexión que requiere la aplicación (WIFI o MOBILE). En caso de no requerir ningún tipo de conexión la constante que se retorna es NONE. La relación de aplicaciones con el tipo de conexión que requieren se presenta en la tabla 4.1.

Tabla 4.1: Aplicaciones y tipo de conexión requerida

Nombre de la aplicación	Tipo de conexión
Marcador telefónico	MOBILE
Navegador de Internet	WIFI
Reproductor de música	NONE
Manejador de mensajes	MOBILE
Lista de contactos	NONE
Calculadora	NONE
Cámara fotográfica	NONE
Manejador de correo electrónico	WIFI

- *checkConnection*. Encargado de verificar si el tipo de conexión que se recibe como parámetro está disponible. Para esto, crea un objeto de la clase `ContextManager` e invoca el método correspondiente.

- *launchApplication*. Este método recibe el nombre de una aplicación y a partir de él obtiene el nombre del paquete que se require para su ejecución. Además utiliza la clase `PackageManager` de Android para adquirir el objeto `Intent` asociado a dicho paquete, a través del método `getLaunchIntentForPackage`.

  En caso de obtener un objeto Intent válido, este se pasa como parámetro al método `startActivity` para iniciar la aplicación.

- *executeInputs*. Método encargado de procesar dos diferentes entradas que corresponden a una aplicación (marcador telefónico o manejador de mensajes) y a un contacto. Este método tiene como finalidad brindar el servicio de enviar un mensaje o llamar al contacto, según el parámetro de nombre y contacto que se recibe.

  Para obtener el objeto Intent que se requiere para la acción de llamar a un contacto en específico se utiliza la siguiente sintaxis:

```
Intent intent = new Intent(Intent.ACTION_DIAL,
        Uri.parse(numberToDial));
```

  Y para la acción de enviar un mensaje a un contacto específico:

```
Intent intent = new Intent(Intent.ACTION_VIEW,
        Uri.parse(numberToMessaging));
```

4.7 Base de datos del sistema multimodal

El sistema multimodal requiere almacenar la información relacionada a las opciones de configuración y a los patrones en cada una de las modalidades disponibles. Para este fin, se ha diseñado e implementado una base de datos que resida en el dispositivo móvil utilizando *SQLite*. La figura 4.16 presenta el diagrama de la base de datos del sistema multimodal.

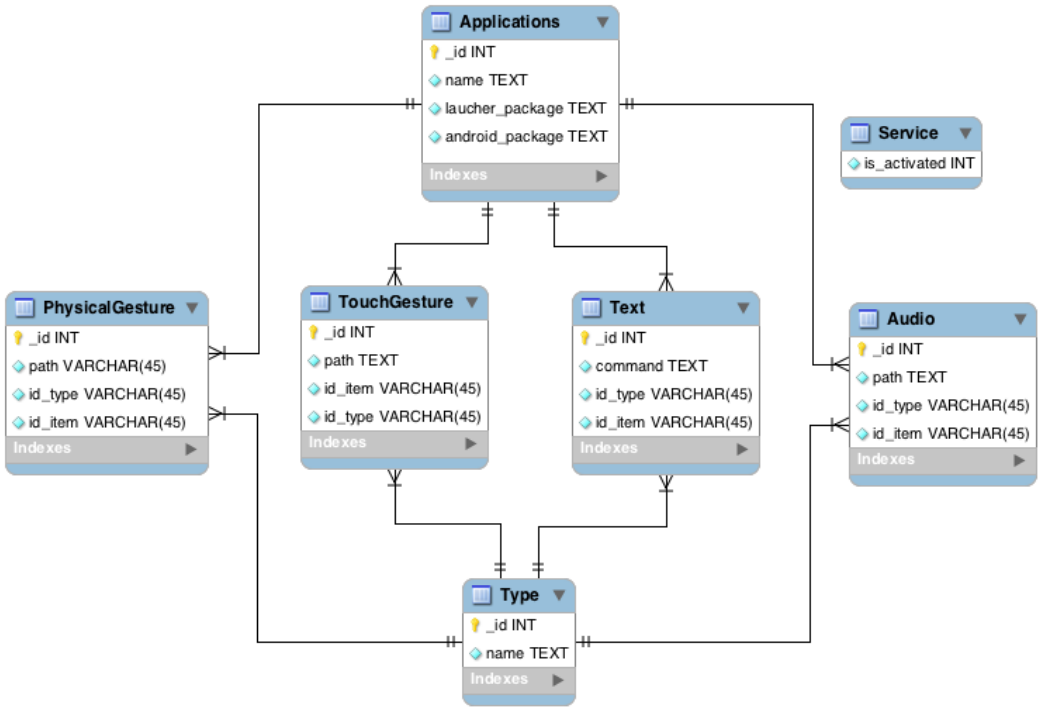


Figura 4.16: Diagrama de la estructura de la base de datos del sistema multimodal

La base de datos del sistema multimodal está conformada por las siguientes tablas:

- ***Applications***: almacena la información de las aplicaciones a las que se le puede asociar un comando en las diferentes modalidades disponibles. Contiene el nombre de la aplicación y la descripción de los paquetes que se requieren para su ejecución en Android.
- ***Type***: como se mencionó en la sección anterior, un patrón puede estar relacionado a una aplicación o a un contacto. Esta tabla proporciona una relación entre un identificador y el nombre de un tipo de servicio, 1: aplicación y 2: contacto.
- ***PhysicalGesture***: tabla destinada al registro de la información relacionada a un nuevo patrón de comando de gesto físico. Los campos de esta tabla corresponden a la

ubicación del archivo que define el patrón, el identificador del tipo de servicio y el identificador asociado a la aplicación o al contacto, según sea el caso.

- ***TouchGesture***: esta tabla registra la información de los patrones de gestos táctiles, como la ubicación del archivo que contiene los valores que describen el gesto táctil, el identificador del tipo de servicio al que está asociado y de acuerdo a éste, el identificador de la aplicación o el contacto.
- ***Text***: esta tabla está conformada por los campos que describen el comando de texto que define el usuario y la información sobre el servicio al que se relaciona dicho comando, como los identificadores del tipo de servicio y de la aplicación o contacto asociado.
- ***Audio***: tabla creada con el objetivo de registrar la información de un patrón de audio: la ubicación del archivo que se genera con las características representativas de la señal de audio, el identificador del tipo de servicio y según corresponda, el identificador del contacto o aplicación al que está asociado el patrón.
- ***Service***: tabla conformada por el campo *is_activated*, el cual indica si el servicio que ejecuta el sistema multimodal en su función de reconocimiento de entradas, está activado o desactivado. Registra un '0' para indicar un valor falso y '1' para un valor verdadero.

Capítulo 5

Pruebas y resultados

El objetivo de este capítulo es presentar las diferentes pruebas que validan el correcto funcionamiento de nuestra propuesta.

Para la realización de las pruebas se generó, a partir del código fuente, un archivo con extensión `.apk` asociado a las aplicaciones para dispositivos Android. Este archivo fue instalado en un dispositivo Samsung Galaxy SII (GT-I9100), con la versión 2.3.4 de Android, procesador Exynos Cortex A9 dual-core 1.2 GHz, con 1 GB en RAM.

Adicionalmente se utilizó la herramienta *Dalvik Debug Monitor Server (DDMS)*, que permite depurar las aplicaciones Android, para dar seguimiento al proceso de ejecución del sistema multimodal y capturar las pantallas del sistema durante la ejecución. Otra herramienta utilizada fue *SQLite Data Browser*, la cual permite examinar los datos que contienen las bases de datos tipo *SQLite* que utiliza Android.

En las siguientes subsecciones se detallan las pruebas realizadas para mostrar la funcionalidad del sistema en relación con los principales casos de uso identificados en la etapa de análisis, descritos en el capítulo 3.

5.1 Configuración del sistema multimodal

El sistema multimodal desarrollado requiere de una etapa de configuración (entrenamiento) del sistema, para definir cada uno de los patrones que se desean relacionar a un comando en específico.

La configuración del sistema multimodal consiste en la selección del servicio que se desea configurar, enseguida se describe un comando en una modalidad específica, se obtiene un patrón a partir de dicha entrada y se almacena en el sistema.

Para realizar la configuración, el usuario debe ejecutar la aplicación multimodal. Al realizar esta acción, el sistema presenta la interfaz gráfica de la pantalla principal que se muestra en la figura 5.1.

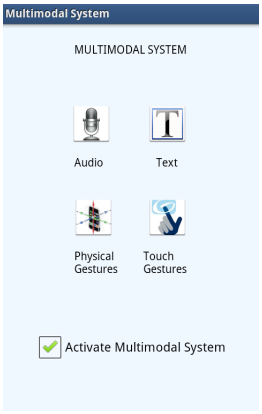


Figura 5.1: Pantalla de inicio del sistema multimodal

Mediante el uso de botones, se presenta al usuario las modalidades disponibles en el sistema (audio, gestos táctiles, gestos físicos y texto) . Adicionalmente, es posible activar/desactivar la ejecución del sistema multimodal en su función de reconocimiento de entradas por medio de una casilla de verificación *(checkbox*).

En el momento que el usuario selecciona una modalidad, el sistema le muestra la interfaz gráfica diseñada para ingresar un comando en la modalidad seleccionada y asociarlo a un servicio del dispositivo.

En las siguientes subsecciones se detallan las pruebas realizadas para validar el funcionamiento de cada una de las opciones de configuración disponibles en el sistema multimodal.

5.1.1 Mostrar y seleccionar los servicios disponibles

El sistema multimodal brinda al usuario la facilidad de seleccionar alguno de los servicios disponibles en el dispositivo móvil para que se le asocie un comando en diferentes modalidades.

La interfaz gráfica que permite al usuario agregar un nuevo comando en cada una de las modalidades, provee el botón , ubicado en la parte superior derecha de la pantalla (ver por ejemplo la figura 5.4). Al momento de presionar este botón, el sistema presenta las opciones de servicios disponibles y brinda la oportunidad de seleccionar alguna aplicación o contacto almacenado en el dispositivo, a los que aún no se le ha asignado un comando en la modalidad seleccionada.

Vale la pena recordar que cada modalidad tiene asociada una tabla en la base de datos del sistema multimodal (ver figura 4.16), es decir, en la base de datos del sistema existen las tablas *Audio*, *PhysicalGesture*, *TouchGesture* y *Text*; estas tablas almacenan la información relacionada con cada uno de los patrones que se han definido en dicha modalidad.

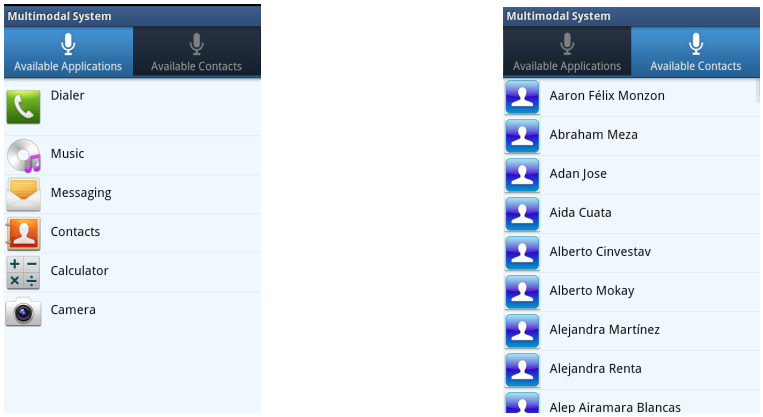
Además existen las tablas *Type* y *Applications*, la primera define un identificador para el tipo de servicio al que se le puede asociar un comando ('1' aplicación o '2' contacto) y la segunda, contiene la información general de cada una de las aplicaciones disponibles.

Para obtener los nombres de las aplicaciones disponibles, se realiza una consulta a la base de datos del sistema multimodal, que permite obtener los nombres de las aplicaciones que no cuentan con un patrón registrado para la modalidad seleccionada.

En el caso de los nombres de contactos, se realiza una consulta a la base de datos de los contactos almacenados en el teléfono y se descartan aquellos nombres de contactos que tienen un registro asociado en la tabla de la modalidad especificada.

A continuación se presenta una prueba realizada que permite mostrar esta funcionalidad en el sistema multimodal.

- **Objetivo:** mostrar los servicios disponibles para la modalidad de audio y seleccionar alguno de ellos.
- **Estado actual del sistema:** la tabla *Audio* de la base de datos contiene los registros asociados a las aplicaciones de *Internet* y *correo electrónico* (*Email*), no contiene ningún registro relacionado al tipo contacto.
- **Prerequisitos:** el usuario ha ejecutado la aplicación multimodal, ha seleccionado de la pantalla principal la opción de *Audio* y el sistema muestra la interfaz gráfica para agregar un nuevo comando de audio.
- **Proceso de prueba:**
  - El usuario presiona el botón  en la interfaz gráfica que proporciona el sistema multimodal para agregar un nuevo comando de audio.
  - El sistema establece una conexión a la base de datos y realiza la consulta que involucra a las tablas *Audio*, *Type* y *Applications*, de ésta última obtiene los nombres de las aplicaciones que no tienen un registro en la tabla *Audio*. Al obtener el resultado de esta consulta, el sistema presenta al usuario una lista con los nombres de las aplicaciones disponibles. Al realizar esta prueba el sistema presenta en pantalla la lista que se aprecia en la figura 5.2 a).
  - Para mostrar la lista de contactos que no tienen asociado un comando de audio, el sistema multimodal se apoya en la base de datos *Contacts* del dispositivo móvil, para obtener los nombres de los contactos que no tienen un registro en la tabla *Audio*. Al obtener el resultado de la consulta, el sistema presenta la interfaz gráfica que lista los nombres de los contactos disponibles para la modalidad de audio. En la figura 5.2 b) se muestra una parte de la lista de contactos que mostró el sistema al realizar esta prueba.



a) Lista de aplicaciones disponibles b) Lista de contactos disponibles

Figura 5.2: Servicios disponibles

- **Resultados:** el sistema presenta seis opciones de aplicaciones disponibles, ya que como se puede observar en la figura 5.3, el sistema tiene registrado que a dos aplicaciones (*Internet* y *correo electrónico*) anteriormente se les definió un comando de audio.

  El sistema muestra una lista con el nombre de ciento siete contactos, que corresponden al número total de contactos almacenados en el dispositivo móvil, esto debido a que no se encuentra almacenado ningún comado asociado a un contacto en la tabla *Audio*.

Applications

id	name	laucher_package	android_package
1	Dialer	com.android.contacts	com.android.phone
2	Internet	com.sec.android.app.latin.launcher.browser	com.android.browser
3	Music	com.sec.android.app.latin.launcher.musicplayer	com.android.music
4	Messaging	com.sec.android.app.latin.launcher.message	com.android.mms
5	Contacts	com.sec.android.app.latin.launcher.contact	com.android.contacts
6	Calculator	com.sec.android.app.calculator	com.sec.android.app.calculator
7	Camera	com.sec.android.app.latin.launcher.camera	com.sec.android.app.camera
8	Email	com.android.email	com.android.email

Audio

_id	path	id_type	id_item
1	/mnt/sdcard/MultimodalSystem/Audio/Internet/1345843480542/LPC.txt	1	2
2	/mnt/sdcard/MultimodalSystem/Audio/Email/1345843518372/LPC.txt	1	8

Type

_id	name
1	Application
2	Contact

Figura 5.3: Estado de la base de datos del sistema al mostrar los servicios disponibles

5.1.2 Agregar un comando de audio

Cuando el usuario selecciona la opción de *Audio*, disponible en la pantalla de inicio (ver figura 5.1), el sistema presenta al usuario la interfaz gráfica que se muestra en la figura 5.4.

Esta interfaz brinda al usuario diversos botones que le permiten realizar las siguientes acciones:

- Seleccionar el servicio al que desea asignar un comando de audio.
- Visualizar los servicios a los que ya se les ha asignado un comando de audio anteriormente.
- Manejar la captura de un comando de audio (iniciar y detener la grabación).
- Almacenar el comando de audio en el sistema.

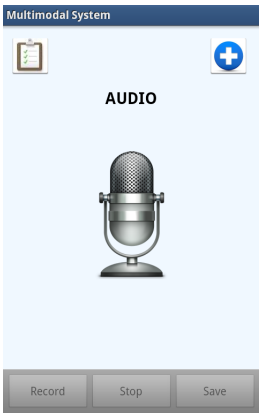


Figura 5.4: Interfaz gráfica para agregar un comando de audio

Una vez que el usuario selecciona el servicio al que desea asociar un comando de audio, el sistema activa los botones de *Record* y *Stop* en la interfaz gráfica para que el usuario pueda proceder a grabar a través del micrófono del dispositivo una palabra (comando de audio).

En el momento que el sistema recibe la indicación de fin de grabación, a través del botón *Stop*, activa el botón *Save*; que al presionarlo, el sistema realiza un conjunto de actividades que permiten preprocesar la señal capturada, obtener a partir de ella el patrón correspondiente y almacenar su información en la base de datos del sistema.

La prueba que se realizó para verificar la funcionalidad de agregar un comando de audio al sistema multimodal, se describe a continuación:

- **Objetivo:** agregar un comando de audio asociado al servico de manejador de mensajes al sistema multimodal.
- **Prerequisitos:** el usuario ha ejecutado la aplicación multimodal, ha seleccionado de la pantalla principal la opción de *Audio* y el sistema muestra la interfaz gráfica para agregar un nuevo comando de audio .

  El usuario ha seleccionado de la lista de aplicaciones disponibles la opción de *Messaging*.
- **Estado actual del sistema:** la tabla *Audio* no contiene ningún registro asociado al servicio de mensajería.

  La interfaz gráfica que brinda el sistema multimodal para agregar un nuevo comando de audio, tiene activado los botones *Record* y *Stop;* además, muestra el nombre del servicio al que se le desea asociar un comando de audio (*Messaging*).
- **Proceso de prueba:**
  - El usuario presiona el botón *Record* para indicar el inicio de la grabación de un comando de audio.
  - El sistema inicia la captura de la señal de audio utilizando el micrófono del dispositivo móvil.
  - El usuario pronuncia en voz alta la palabra "mensaje" que desea asociar como comando a la aplicación de *Messaging*.
  - El sistema graba los valores que registra el micrófono.
  - El usuario presiona el botón *Stop* para finalizar la grabación de audio.
  - El sistema libera los recursos que requirió del dispositivo para detener la grabación y genera un archivo que contiene los valores que representan la señal de audio de la palabra "mensaje". En la figura 5.5 se muestra la señal obtenida al momento de grabar la palabra "mensaje".
  - Si el usuario desea guardar la grabación del comando de audio debe presionar el botón *Save* en la interfaz gráfica.
  - El sistema procede a realizar un conjunto de tareas que permiten obtener el patrón asociado al comando y almacenarlo en el sistema. Las tareas que se llevan a cabo son las descritas en la sección 4.1.2, que corresponden a la etapa de preprocesamiento y extracción de características de la señal de voz.

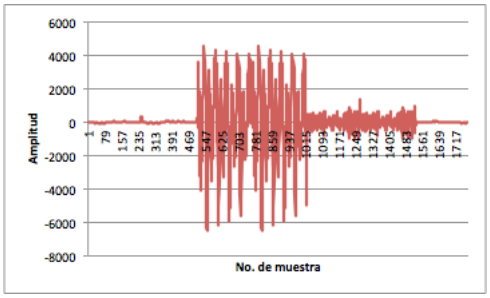


Figura 5.5: Señal de la palabra "mensaje"

La señal de audio de la palabra "mensaje" que se muestra en la figura 5.5, pasa al proceso de detección de voz para obtener la sección que representa la voz y se obtiene la señal de la señal de la figura 5.6, donde se puede observar que se han descartado aquellas secciones donde no existen valores representativos de voz.

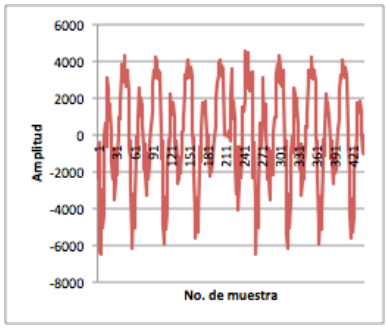


Figura 5.6: Señal de la palabra "mensaje" después del proceso de detección de voz

Posteriormente, a la señal obtenida se le aplica un filtro preénfasis y se obtiene la señal que se muestra en la figura 5.7, donde se han descartado características no deseadas en la señal y se han enfatizado las frecuencias altas.

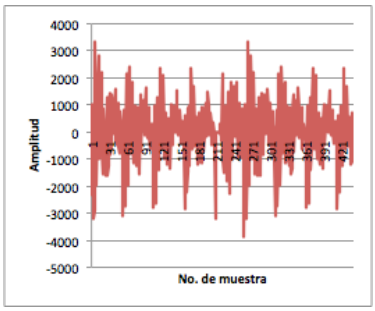


Figura 5.7: Señal de la palabra "mensaje" después de aplicar el filtro preénfasis

Enseguida, el sistema procede a realizar el proceso de ventaneo de la señal para aplicar la técnica de extracción de características con la finalidad de obtener una codificación representativa de la señal.

De este procedimiento se obtiene un arreglo con los valores de la codificación, como el que se muestra en la tabla 5.1.

Tabla 5.1: Valores de la codificación de la señal de audio

Índice	Valor	Índice	Valor
0	-0.21934006	10	-0.17343959
1	-0.097685635	11	-0.060151353
2	-0.060309242	12	-0.03252835
3	-0.15136312	13	-0.15924722
4	-0.08260168	14	-0.07057901
5	-0.012760382	15	-0.023507353
6	-0.066312306	16	-0.100821115
7	-0.10240088	17	-0.1272217
8	0.034375392	18	-0.008743662
9	-0.19993657	19	-0.20934474

Estos valores representan el patrón obtenido del comando de audio “mensaje”, los cuales son almacenados en un archivo de texto y la referencia de la ubicación de este archivo se registra en la base de datos del sistema.

- **Resultados:** el sistema crea un nuevo registro en la tabla *Audio*, donde almacena la información relacionada al patrón obtenido: la ubicación del archivo, el identificador del servicio asociado y del tipo de servicio. De esta manera se agrega un nuevo comando de audio al sistema multimodal, sin registrarse ningún problema en el procedimiento.

  En la figura 5.8 se presenta el estado de la tabla *Audio* después de agregar el comando “mensaje”. Como se observa, se agrega el registro asociado al servicio de mensajería (*Messaging*) con el identificador ’3’ en la tabla *Audio*. De esta manera se encuentran registrados tres patrones de audio relacionados a aplicaciones y ninguno a algún contacto.

Audio

_id	path	id_type	id_item
1	/mnt/sdcard/MultimodalSystem/Audio/Internet/1345843480542/LPC.txt	1	2
2	/mnt/sdcard/MultimodalSystem/Audio/Email/1345843518372/LPC.txt	1	8
3	/mnt/sdcard/MultimodalSystem/Audio/Messaging/1345855090071/LPC.txt	1	4

Type

_id	name
1	Application
2	Contact

Applications

_id	name	laucher_package	android_package
1	Dialer	com.android.contacts	com.android.phone
2	Internet	com.sec.android.app.latin.launcher.browser	com.android.browser
3	Music	com.sec.android.app.latin.launcher.musicplayer	com.android.music
4	Messaging	com.sec.android.app.latin.launcher.message	com.android.mms
5	Contacts	com.sec.android.app.latin.launcher.contact	com.android.contacts
6	Calculator	com.sec.android.app.calculator	com.sec.android.app.calculator
7	Camera	com.sec.android.app.latin.launcher.camera	com.sec.android.app.camera
8	Email	com.android.email	com.android.email

Figura 5.8: Estado de la tabla Audio después de agregar el comando "mensaje"

5.1.3 Agregar un comando de gesto táctil

Otra de las modalidades disponibles en el sistema es la de gestos táctiles. Para agregar un nuevo comando utilizando esta modalidad, el usuario selecciona la opción *Touch Gesture* disponible en la pantalla de inicio (ver figura 5.1).

La figura 5.9 muestra la interfaz gráfica diseñada para que el usuario agregue un comando de gesto táctil al sistema multimodal.

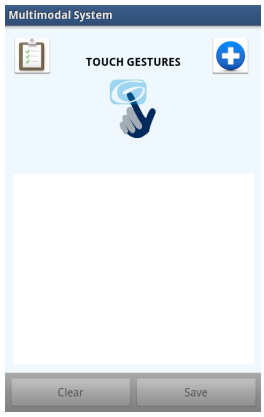


Figura 5.9: Interfaz para capturar un comando de gesto táctil

A través de esta interfaz el sistema brinda las siguientes funciones al usuario:

- Seleccionar el servicio al que desea asignar un comando de gesto táctil.
- Visualizar los servicios a los que previamente se les ha asignado un comando de gesto táctil.
- Dibujar sobre la pantalla táctil del dispositivo el gesto que desea definir como comando al servicio seleccionado.
- Borrar de la pantalla el gesto dibujado.
- Almacenar el comando de gesto táctil en el sistema.

Cuando el usuario realiza la selección del servicio que desea, el sistema queda a la espera de que se dibuje un gesto sobre el área definida para esta acción (recuadro blanco de la interfaz gráfica de la figura 5.9).

En el momento que el sistema recibe como entrada un gesto táctil, se activan los botones *Clear* y *Save*, para brindar al usuario las opciones de borrar el gesto táctil dibujado o guardarlo como comando en el sistema, respectivamente.

Para validar el correcto funcionamiento de las acciones requeridas para agregar un nuevo comando de gesto táctil en el sistema multimodal, se realizaron una serie de pruebas. Una de las cuales consistió en asociar un gesto táctil a un contacto disponible en el dispositivo móvil; esta prueba se detalla a continuación:

- **Objetivo:** agregar un comando de gesto táctil para que esté asociado a un contacto disponible en el dispositivo móvil.
- **Prerequisitos:** el usuario ha ejecutado la aplicación multimodal, ha seleccionado de la pantalla principal la opción de *Touch Gesture* y el sistema muestra la interfaz gráfica para agregar un nuevo comando de gesto táctil.

  El usuario ha seleccionado de la lista de contactos disponibles el nombre *Eduardo Urias*.
- **Estado actual del sistema:** la tabla *TouchGesture* no contiene ningún registro asociado al contacto *Eduardo Urias*.

  La interfaz gráfica, que brinda el sistema multimodal, para agregar un nuevo comando de gesto táctil muestra un texto con el nombre del contacto al que se le desea asociar un comando en dicha modalidad.

- **Proceso de prueba:**

  - El usuario dibuja sobre la interfáz gráfica el gesto que desea asociar al contacto *Eduardo Urias*.
  - El sistema activa los botones *Clear* y *Save* en la interfaz gráfica.

    En ese momento el sistema presenta la interfaz gráfica que se muestra en la figura 5.10, en la que se puede apreciar el gesto capturado.

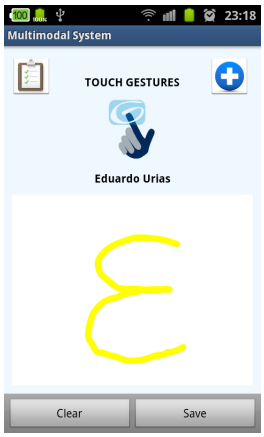


Figura 5.10: Interfaz gráfica al dibujar un gesto táctil asociado a un contacto

  - Si el usuario considera que el gesto dibujado no es correcto, presiona el botón *Clear* y el sistema lo borra de la pantalla. Los botones *Clear* y *Save* son desactivados hasta el momento que el sistema reciba como entrada un nuevo gesto.
  - Cuando el usuario dibuja el gesto que desea asociar al contacto *Eduardo Urias*, presiona el botón *Save* para almacenarlo como un nuevo comando de gesto táctil.

    Para realizar esta acción el sistema crea un archivo que contiene la información sobre los valores que definen cada uno de los puntos del dibujo del gesto. La ubicación de este archivo se almacena en la base de datos del sistema.

- **Resultados:** en la base de datos del sistema se crea un nuevo registro en la tabla *TouchGesture*, que incluye la información sobre la ubicación del archivo mencionado, el identificador del contacto al que está relacionado el comando de gesto táctil y el tipo de servicio (2: contacto).

  En la figura 5.11 se puede observar este nuevo registro en la tabla *TouchGesture* con el identificador '3'. Además es posible apreciar las tablas *Contacts* y *Phone*, que forman parte de la base de datos de los contactos del dispositivo móvil, las cuales se

encuentran relacionadas con la tabla *TouchGesture* a través del identificador del contacto.

Al momento de agregar el registro asociado al contacto *Eduardo Urias*, se encuentran otros dos registros, el primero de ellos con el identificador '1', asociado a otro contacto (*Fabiola Ponce*) y el segundo, con el identificador '2', relacionado a la aplicación de cámara fotográfica.

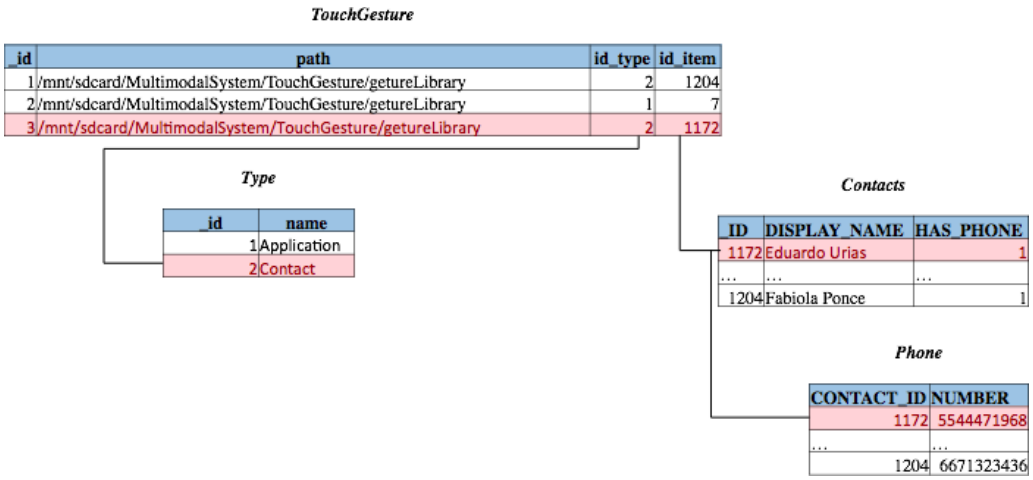


Figura 5.11: Estado de la tabla TouchGesture después de agregar un gesto táctil asociado a un contacto

5.1.4 Agregar un comando de gesto físico

El sistema multimodal permite al usuario agregar comandos de tipo gesto físico; este tipo de comandos se definen por el movimiento del dispositivo móvil, al seguir una cierta trayectoria.

Para acceder a la opción que permite agregar un nuevo comando de gesto físico, el usuario selecciona la opcion *Physical Gesture* de la pantalla principal (ver figura 5.1) y al hacerlo el sistema presenta la interfaz gráfica de la figura 5.12.

Esta interfaz brinda al usuario diversos botones que le permiten llevar a cabo las siguientes acciones:

- Seleccionar el servicio al que desea asociar un comando de gesto físico.
- Visualizar los servicios a los que ya se les ha asignado un comando de gesto físico.
- Indicar el inicio y fin de la grabación del comando.
- Almacenar el comando de gesto físico en el sistema.

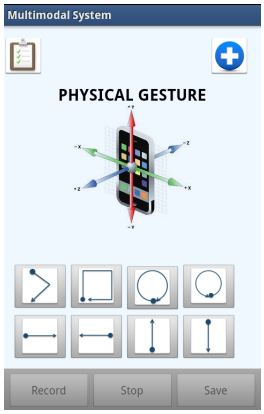


Figura 5.12: Interfaz para capturar un comando de gesto físico

El sistema no deja al usuario realizar el proceso de agregar un nuevo comando de gesto físico hasta que seleccione el servicio al cual desea asociarlo. Una vez seleccionado el servicio, el sistema permite al usuario seleccionar alguna descripción de gesto físico de las propuestas o ingresar directamente alguna de su preferencia.

Cuando el usuario realizar esto último, puede iniciar la grabación de un gesto físico al presionar el botón *Record*. En ese momento el sistema inicia la grabación de los valores que registra el acelerómetro del dispositivo móvil.

Para indicar el fin de la grabación se presiona el botón *Stop* y al hacerlo, se activa el botón *Save* cuya función es comunicar al sistema que, a partir la grabación que acaba de realizar, se debe obtener un patrón que defina un comando de gesto físico y la información relacionada a dicho patrón debe ser almacenada.

Para probar esta funcionalidad se procede a grabar un comando de gesto físico que se asocia al servicio de calculadora. A continuación se describe la manera en que se realizó la prueba.

- **Objetivo:** agregar un comando de gesto físico y relacionarlo al servicio de calculadora del dispositivo móvil.

- **Prerequisitos:** el usuario ha ejecutado la aplicación multimodal, ha seleccionado de la pantalla principal la opción de *Physical Gesture* y el sistema muestra la interfaz gráfica para agregar un nuevo comando de gesto físico.

  El usuario ha seleccionado de la lista de aplicaciones disponibles la opción de *Calculator*.

- **Estado actual del sistema:** la tabla *PhysicalGesture* no contiene ningún registro asociado al servicio de calculadora.

La interfaz gráfica que brinda el sistema multimodal para agregar un nuevo comando de gesto físico, tiene activado los botones *Record* y *Stop*. Además muestra una etiqueta con el texto *Calculator* indicando el nombre del servicio seleccionado.

- **Proceso de prueba:**

  - El usuario selecciona el botón  , de la interfaz gráfica, indicando de esta manera la trayectoria de desea describir con el movimiento del dispositivo móvil y presiona el botón *Record* disponible en la interfaz gráfica para iniciar la grabación.
  - El sistema registra el servicio de escuchar los cambios registrados por el acelerómetro del dispositivo móvil.
  - El usuario describe la trayectoria seleccionada con el movimiento del dispositivo, mientras el sistema graba los valores que registra el acelerómetro en sus ejes x, y, z.
  - El usuario termina de realizar el movimiento y presiona inmediatamente el botón *Stop*.
  - El sistema recibe el mensaje que indica el fin de la captura del comando de gesto físico y termina el servicio que escucha y graba los valores registrados por el acelerómetro. Una representación de los datos obtenidos puede obrservarse en la figura 5.13 , la cual muestra las señales de los ejes x, y z que describen el movimiento del dispositivo.

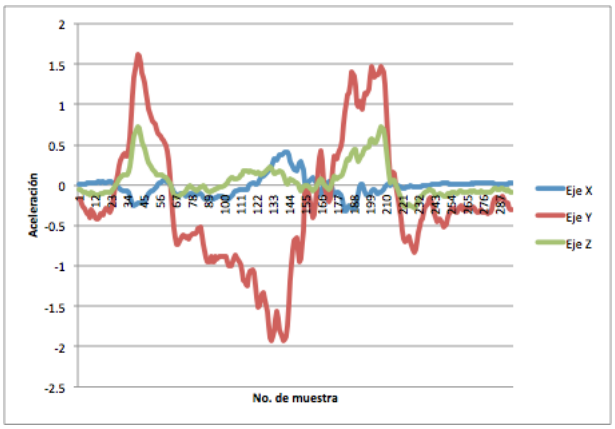


Figura 5.13: Señales grabadas en los ejes x, y, z para el comando de gesto físico

  - El usuario presiona el botón *Save* para indicar al sistema que desea almacenar el gesto físico que acaba de grabar y asociarlo como comando a la aplicación *Calculator*. Al realizar esta acción, el sistema lleva a cabo una serie de tareas que

permiten obtener el patrón del gesto físico y almacenar su referencia en la base de datos del sistema.

- A partir de los valores que representan las señales de los ejes *x*, *y*, *z*, el sistema calcula los valores que corresponden a la señal de aceleración total. La figura 5.14 presenta la señal de aceleración obtenida.

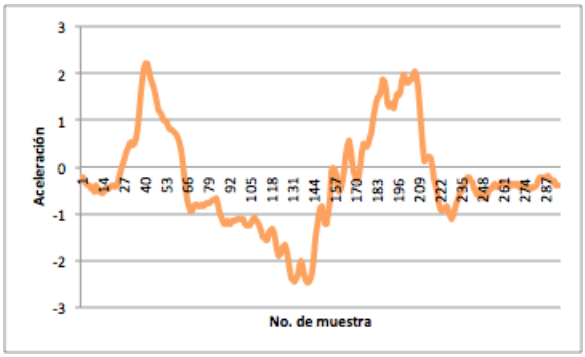


Figura 5.14: Señal de aceleración total para el comando de gesto físico

Una vez que se tiene los valores de la aceleración total, el sistema realiza el proceso de suavizado de la señal para reducir el ruido. Estos valores se almacenan en un archivo y representan el patrón del gesto físico asociado al servicio de calculadora. La figura 5.15 muestra la señal después del proceso de suavizado.

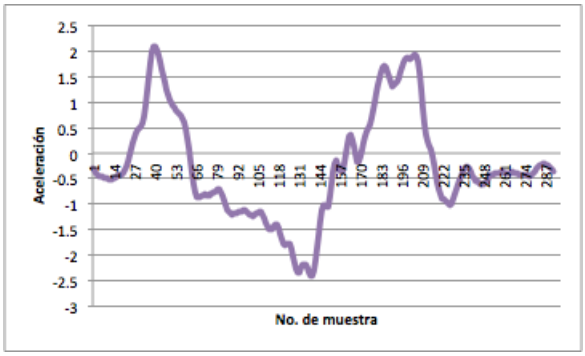
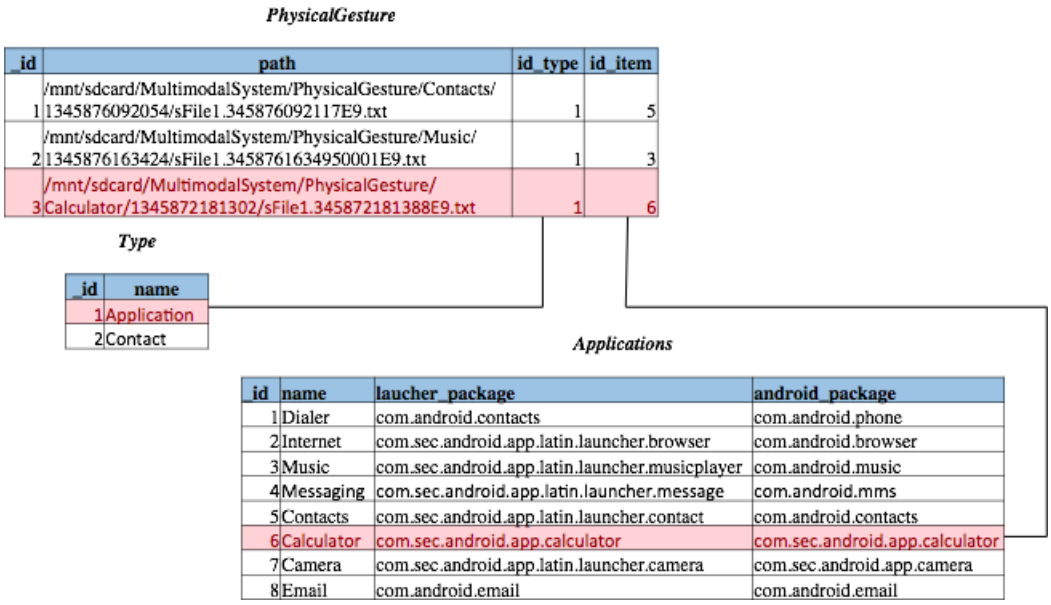


Figura 5.15: Señal de aceleración total después del proceso de suavizado

- El sistema almacena en la tabla *PhysicalGesture*, la referencia de ubicación del archivo que contiene los valores del patrón, así como el nombre de la aplicación a la que está relacionado.

- **Resultados:** el sistema agrega un nuevo registro a la tabla *PhysicalGesture* donde se almacenan el identificador de la aplicación *Calculator*, el identificador del tipo de servicio (1: aplicación) y la ubicación del archivo que contiene los valores que describen el patrón de gesto físico. La figura 5.16 muestra el registro de este nuevo elemento en la tabla *PhysicalGesture.*

  Se puede observar que además de este registro, se tienen otros dos relacionados a las aplicaciones *Contacts* y *Music*. De esta manera se tienen tres comandos en la modalidad de gesto físico registrados en el sistema.



PhysicalGesture

_id	path	id_type	id_item
1	/mnt/sdcard/MultimodalSystem/PhysicalGesture/Contacts/1345876092054/sFile1.345876092117E9.txt	1	5
2	/mnt/sdcard/MultimodalSystem/PhysicalGesture/Music/1345876163424/sFile1.3458761634950001E9.txt	1	3
3	/mnt/sdcard/MultimodalSystem/PhysicalGesture/Calculator/1345872181302/sFile1.345872181388E9.txt	1	6

Type

_id	name
1	Application
2	Contact

Applications

id	name	laucher_package	android_package
1	Dialer	com.android.contacts	com.android.phone
2	Internet	com.sec.android.app.latin.launcher.browser	com.android.browser
3	Music	com.sec.android.app.latin.launcher.musicplayer	com.android.music
4	Messaging	com.sec.android.app.latin.launcher.message	com.android.mms
5	Contacts	com.sec.android.app.latin.launcher.contact	com.android.contacts
6	Calculator	com.sec.android.app.calculator	com.sec.android.app.calculator
7	Camera	com.sec.android.app.latin.launcher.camera	com.sec.android.app.camera
8	Email	com.android.email	com.android.email

Figura 5.16: Estado de la tabla PhysicalGesture después de agregar un gesto físico asociado a al servicio calculadora

5.1.5 Agregar un comando de texto

Los comandos de texto también son una modalidad más disponible en el sistema; para que el usuario pueda agregar un nuevo comando de este tipo selecciona en la pantalla de inicio la opción *Text* (ver figura 5.1).

El sistema provee la interfaz gráfica que se muestra en la figura 5.17 para que el usuario realice las siguientes acciones cuando lo requiera:

- Seleccionar el servicio al que desea asociar un comando de texto.
- Visualizar los servicios a los que ya se les ha asignado un comando de texto.
- Borrar el contenido del cuadro de captura de texto disponible en la interfaz gráfica.

- Almacenar el comando de texto asociado al servicio seleccionado.

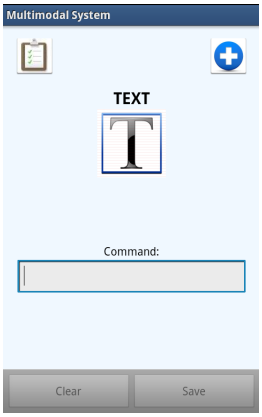


Figura 5.17: Interfaz para capturar un comando de texto

Se agregaron varios comandos de texto al sistema multimodal para probar esta funcionalidad, entre ellos está el comando al que se le asocia el servicio de navegador de Internet. El proceso de esta prueba en específico se presenta a continuación:

- **Objetivo:** agregar en el sistema multimodal un comando de texto al servicio de navegador de Internet .

- **Prerequisitos:** el usuario ha ejecutado la aplicación multimodal, ha seleccionado de la pantalla principal la opción de *Text* y el sistema muestra la interfaz gráfica para agregar un nuevo comando de texto.

  El usuario ha seleccionado de la lista de aplicaciones disponibles la opción *Internet*, que cumple el servicio de navegador de Internet.

- **Estado actual del sistema:** la tabla *Text* no contiene ningún registro asociado al servicio de navegador de Internet.

  La interfaz gráfica que brinda el sistema multimodal para agregar un nuevo comando de texto, tiene activado los botones *Clear* y *Save*. Además muestra el texto Internet indicando el nombre del servicio seleccionado.

- **Proceso de prueba:**

  - El usuario ingresa una secuencia de caracteres en el cuadro de captura de texto que brinda la interfaz gráfica.
  - El sistema, al detectar la entrada de caracteres (el cuadro de captura de texto no está vacío) activa los botones *Clear* y *Save*.

- Si el usuario desea borrar la secuencia de caracteres capturada hasta el momento, presiona el botón *Clear* y el sistema elimina el contenido del cuadro de captura. En ese momento los botones *Clear* y *Save* son desactivados por el sistema.

- El usuario captura el texto "chrome" para asociarlo a la aplicación de Internet y presiona el botón *Save*. La figura 5.18 presenta la apariencia de la interfaz gráfica en el momento en que el usuario ingresó el comando de texto.

- El sistema obtiene el texto del cuadro de captura y lo registra en la base de datos del sistema como un nuevo comando de texto.

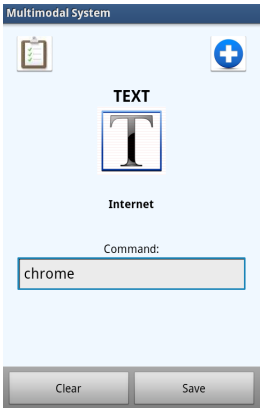


Figura 5.18: Interfaz gráfica al ingresar un comando de texto para el servicio de navegador de Internet

- **Resultados:** la tabla *Text* cuenta con un nuevo registro que contiene el texto obtenido de la interfaz gráfica y la información del servicio de navegador de Internet: el comando de texto "chrome", el identificador tipo de servicio (1: aplicación) y el identificador de la aplicación Internet.

  El procedimiento de agregar un nuevo comando de texto se realizó de manera satisfactoria. En la figura 5.19 se puede apreciar que este nuevo registro se agrega con el identificador '6' en la tabla *Text*. Otros cinco registros se encuentran en la misma tabla, uno de ellos asociado al contacto con el identificador '1204' y los demás a las aplicaciones *Music*, *Calculator*, *Contacts* y *Camera*.

Text

_id	command	id_type	id_item
1	play	1	3
2	faby	2	1204
3	calc	1	6
4	cont	1	5
5	cam	1	7
6	chrome	1	2

Type

_id	name
1	Application
2	Contact

Applications

id	name	laucher_package	android_package
1	Dialer	com.android.contacts	com.android.phone
2	Internet	com.sec.android.app.latin.launcher.browser	com.android.browser
3	Music	com.sec.android.app.latin.launcher.musicplayer	com.android.music
4	Messaging	com.sec.android.app.latin.launcher.message	com.android.mms
5	Contacts	com.sec.android.app.latin.launcher.contact	com.android.contacts
6	Calculator	com.sec.android.app.calculator	com.sec.android.app.calculator
7	Camera	com.sec.android.app.latin.launcher.camera	com.sec.android.app.camera
8	Email	com.android.email	com.android.email

Figura 5.19: Estado de la tabla Text después de agregar el comando de texto asociado al navegador de Internet

5.1.6 Consultar los servicios entrenados

En caso de que el usuario desee consultar las aplicaciones a las que ya ha asociado un comando en alguna de las modalidades, pude realizarlo a través del botón  , este botón es desplegado en la interfaz gráfica que provee el sistema para agregar un nuevo comando en cada modalidad (ver por ejemplo la figura 5.17).

En el momento que el usuario presiona este botón, el sistema presenta las opciones de servicios disponibles y brinda la posibilidad de consultar las aplicaciones o contactos a los que ya se les ha asignado un comando en la modalidad seleccionada.

Para llevar a cabo esta funcionalidad, el sistema de manera similar a la función de mostrar los servicios disponibles, utiliza las tablas que almacenan la información de los comandos de cada modalidad y las que contienen la información sobre los contactos del dispositivo móvil y las aplicaciones.

El sistema realiza una consulta a la base de datos del sistema multimodal, que permita obtener los nombres de las aplicaciones que cuentan con un registro en la tabla que almacena la información de los patrones definidos para la modalidad seleccionada y los muestra al usuario en forma de lista.

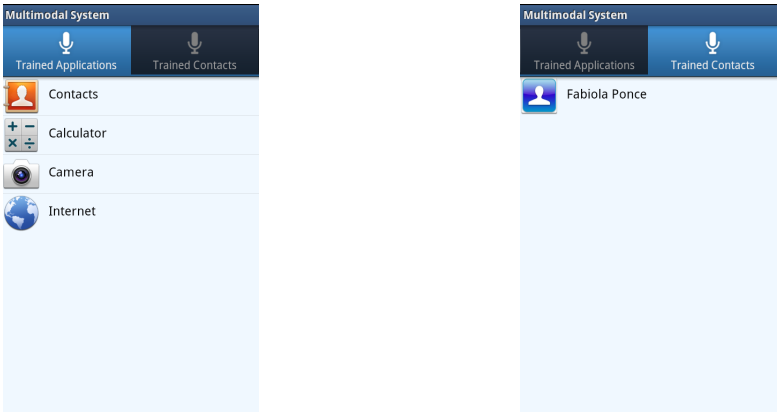
Para obtener los nombres de contactos, se realiza una consulta a la base de datos de los contactos almacenados en el teléfono y se adquieren los nombres de contactos que tienen

un registro asociado en la tabla de la modalidad especificada.

A continuación se presenta una de las pruebas realizadas, que permite mostrar la funcionalidad de mostrar los servicos entrenados previamente en el sistema multimodal.

- **Objetivo:** consultar aquellos servicios a los que anteriormente se les ha asignado un comando de texto.
- **Estado actual del sistema:** la tabla *Text* de la base de datos está en el estado que muestra la figura 5.19, contiene los registros asociados a las aplicaciones de *Internet, Contacts, Calculator* y *Camera*, además contiene el registro de un comando para el contactos *Fabiola Ponce*.
- **Prerequisitos:** el usuario ha ejecutado la aplicación multimodal, ha seleccionado de la pantalla principal la opción de *Text* y el sistema muestra la interfaz gráfica para agregar un nuevo comando de audio.
- **Proceso de prueba:**
  - El usuario presiona el botón  en la interfaz gráfica que proporciona el sistema multimodal para agregar un nuevo comando de texto (figura 5.17).
  - El sistema establece una conexión a la base de datos y realiza la consulta que involucra a las tablas *Text*, *Type* y *Applications*; de ésta última obtiene los nombres de las aplicaciones que tienen un registro en la tabla *Text*. Al obtener el resultado de esta consulta, el sistema presenta al usuario una lista con los nombres de las aplicaciones que ya tienen un comando de audio asignado. Al realizar esta prueba el sistema muestra en pantalla la lista que se aprecia en la figura 5.20 a).
  - Para mostrar la lista de contactos que tienen asociado un comando de audio, el sistema multimodal se apoya en la base de datos *Contacts* del dispositivo móvil, para obtener los nombres de los contactos que tienen un registro en la tabla *Text*. Al obtener el resultado de la consulta, el sistema presenta la interfaz gráfica que lista los nombres de los contactos que tienen registrado un comando en la modalidad de texto. En esta caso, se aprecia en pantalla la figura 5.20 b).
- **Resultados:** el sistema presenta el nombre de cuatro aplicaciones que actualmente tienen asignado un comando de texto. En la opcion de contactos, muestra el nombre del contacto *Fabiola Ponce*.

  Este mismo procedimiento se repitió sin presentar ningún inconveniente, cada vez que se deseaba consultar las aplicaciones o contactos que ya cuentan con un comando asociado en una cierta modalidad.



a) Lista de aplicaciones asociadas a un comando de texto

b) Lista de contactos asociadas a un comando de texto

Figura 5.20: Consulta de servicios entrenados

5.2 Activación de la ejecución del sistema multimodal

La aplicación desarrollada brinda la opción activar la ejecución del sistema multimodal, en su función de reconocimiento de entradas en las modalidades disponibles, para activar algún servicio en el dispositivo móvil.

La opción para activar la ejecución del sistema está disponible en la pantalla de inicio de configuración del sistema. El usuario puede seleccionar/deseleccionar la casilla de verificación (*checkbox*) para activar/desactivar esta función. La figura 5.21 muestra remarcada en color rojo esta opción.

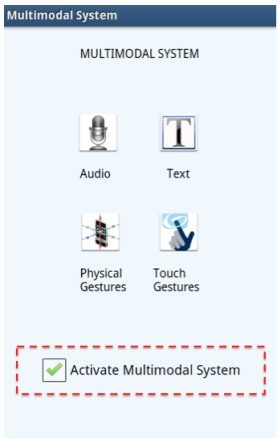


Figura 5.21: Opción de activar la ejecución del sistema multimodal

Cuando el usuario activa esta opción, el sistema registra en la tabla *Service* un valor verdadero al campo *is_activated* e inicia la ejecución del servicio que está a la escucha de que el usuario realice un movimiento de sacudida (*shake*) con el dispositivo móvil (ver figura 5.22[1]), para mostrar la interfaz gráfica diseñada para recibir y reconocer las entradas de los comandos en la diferentes modalidades (audio, gesto táctil, gesto físico y texto).

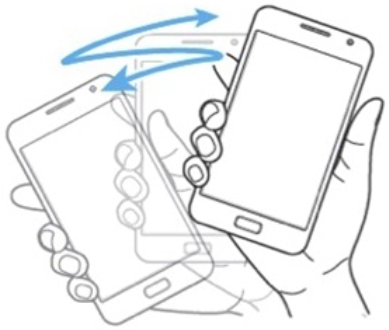


Figura 5.22: Movimiento de sacudir el dispositivo móvil (*shake*)

Un caso especial se presenta cuando el dispositivo móvil es apagado, debido a que todos los servicios que se están ejecutando en ese momento son terminados. Es por esta razón, que el sistema multimodal incluye un *broadcast receiver,* el cual está programado para que en el momento que se produce el evento de arranque (*boot*) del sistema del dispositivo móvil, realice una consulta a la base de datos y obtenga el valor que registra el campo *is_activated* de la tabla *Service*. En caso de que el valor obtenido de esta consulta sea verdadero, inicia de nuevo la ejecución del servicio que está a la escucha de un movimiento de sacudida.

De esta manera el usuario puede iniciar la ejecución del sistema multimodal de manera facíl, sin necesidad de ingresar al menú de aplicaciones instaladas en el dispositivo y realizar una búsqueda. Por otra parte, puede ejecutar el sistema en cualquier momento sin importar si se encuentra ejecutando otra aplicación, en la pantalla de inicio del dispositivo móvil, en el menú de aplicaciones, etc.

A continuación se describe una prueba realizada que detalla la forma en que el usuario activó la ejecución del sistema multimodal y éste muestra la interfaz gráfica que permite el ingreso de comandos para su reconocimiento.

[1]http://software2tech.com

- **Objetivo:** activar la ejecución del sistema multimodal en la función de reconocimiento de entradas.
- **Prerequisitos:** el usuario ha ejecutado la aplicación multimodal.
- **Estado actual del sistema:** el campo *is_activated* de la tabla *Service* tiene registrado un valor falso.

  La casilla de verificación de la interfaz gráfica está desactivada.
- **Proceso de prueba:**
  - El usuario activa la ejecución del servicio del sistema multimodal, al seleccionar la casilla de verificación (*checkbox*) en la pantalla de inicio de la aplicación multimodal.
  - El sistema registra en el campo *is_activated* un valor verdadero e inicia la ejecución del servicio en segundo plano, que está a la escucha de que ocurra un movimento de sacudida del dispositivo móvil. Para realizar esta última tarea, el sistema obtiene los valores registrados por el acelerómetro.
  - El usuario sale de la aplicación multimodal, continua interactuando normalmente con el dispositivo móvil y ejecuta manualmente la aplicación de lista de contactos.
  - Mientras tanto el sistema permanece a la escucha de los valores del acelerómetro y verificando si el dispositivo recibe un movimiento de sacudida.
  - El usuario realiza un movimiento de sacudida del dispositivo móvil.
  - El sistema registra que ha ocurrido un cambio en los valores del acelerómetro que corresponden al movimiento *shake* y ejecuta el sistema multimodal en su función de reconocimiento de entradas.
- **Resultados:** la base de datos del sistema multimodal registra un valor verdadero en la tabla *Service* y el sistema multimodal presenta la interfaz gráfica de la figura 5.23 que permite al usuario ingresar los comandos que ha agregado anteriormente en el sistema para ejecutar los servicios disponibles en el dispositivo móvil.

Una prueba más que se realizó consistió en verificar que una vez que el usuario ha activado la ejecución del sistema multimodal y el dispositivo móvil es apagado, al encenderlo de nuevo, el servicio multimodal continue ejecutándose y permita acceder a la interfaz gráfica destinada para el ingreso de comandos. A continuación se detalla esta prueba:

- **Objetivo:** verificar el correcto funcionamiento del servicio multimodal después de que el dispositivo móvil es apagado.

- **Prerequisitos:** el usuario ha activado la ejecución del servicio multimodal en la interfaz de configuración.
- **Estado actual del sistema:** el campo *is_activated* de la tabla *Service* tiene registrado un valor verdadero.

  El dispositivo móvil se encuentra apagado, por lo que ningún servicio se está ejecutando.
- **Proceso de prueba:**
  - El usuario enciende el dispositivo móvil.
  - El dispositivo móvil inicia el proceso de arranque del sistema.
  - El sistema multimodal, a través de su *broadcast receiver,* recibe la notificación de que se está llevando a cabo el evento de *boot* por lo que procede a realizar una consulta a la tabla *Service* para obtener el valor del campo *is_activated*.
  - El sistema obtiene un valor verdadero de la consulta e inicia inmediatamente la ejecución en segundo plano del servicio que está a la escucha de la ocurrencia de un movimiento de sacudida del dispositivo móvil.
  - En cuanto el dispositivo se enciende y está disponible para su uso, el usuario realiza el movimiento de sacudida utilizando dicho dispositivo.
  - El sistema multimodal, a través del servicio que se está ejecutando en trasfondo, detecta que ha ocurrido un movimiento tipo *shake* y muestra la interfaz gráfica de la figura 5.11 que permite al usuario ingresar comandos utilizando las modalidades disponibles.
- **Resultados:** el servicio que está a la escucha de que ocurra un movimiento de sacudida del dispositivo se activa y permite ejecutar el sistema multimodal en su función de reconocimiento.

5.3 Activación de servicios a través de diferentes modalidades

Como se mencionó anteriormente, el sistema multimodal brinda una interfaz gráfica que permite el ingreso de comandos utilizando las diferentes modalidades disponibles, para que sean reconocidos por el sistema y éste active el servicio del dispositivo móvil al que están relacionados dichos comandos. La interfaz gráfica se presenta en la figura 5.23, sus elementos y funcionalidad se describen a continuación:

- Provee el botón  , que al presionarlo inicia el proceso de captura de un comando de voz.

- Dispone del botón  , que brinda al usuario la opción de ingresar un comando de gesto físico.
- Integra un cuadro de captura de texto para que el usuario pueda ingresar un comando de texto.
- Brinda un área destinada para el dibujo de comandos de gestos táctiles.

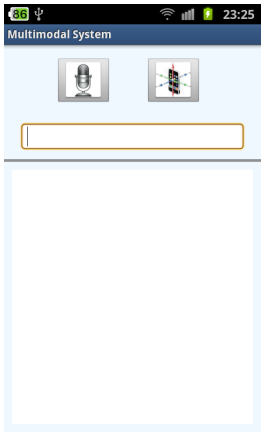


Figura 5.23: Interfaz para ingresar comandos para su reconocimiento

En el momento que el sistema detecta que el usuario ingresó algún comando utilizando esta interfaz, realiza el proceso de reconocimiento para activar el servicio asociado al comando que recibió.

En las siguientes subsecciones se presentan una conjunto de pruebas que se realizaron para validar el funcionamiento que permite activar los servicios disponibles en el dispositivo, utilizando los comandos que se han agregado al sistema en las modalidades de audio, gestos físicos, gestos táctiles y texto.

5.3.1 Modalidad de audio

El usuario puede ingresar un comando en la modalidad de audio para activar alguno de los servicios a los que les asoció un comando en esta misma modalidal.

Para realizar esta función, el usuario debe ejecutar el sistema multimodal en su modo de reconocimiento de entradas, para que se muestre la interfaz de la figura 5.23.

El sistema brinda al usuario dos formas de ingresar un comando de audio, la primera de ellas consiste en seleccionar el botón  , que provee la interfaz gráfica y la segunda es presionar el botón de incrementar volumen del dispositivo móvil. Al realizar cualquiera de estas acciones el sistema inicia el proceso de captura de un comando de audio.

Una vez que el sistema captura el comando de audio, obtiene el patrón a partir de la señal de audio capturada y realiza una etapa de reconocimiento de patrones para determinar qué patrón de los almacenados anteriormente en esta modalidad tiene mayor similaridad con el que acaba de obtener. El sistema utiliza la información del patrón que reconoció, para poder activar el servicio al que está asociado.

Se realizaron varias pruebas para validar la funcionalidad de activar un servicio mediante un comando de audio; una de ellas consistió en activar el servicio de mensajería utilizando el comando de audio "mensaje", que se agregó en el proceso de prueba descrito en la subsección 5.1.1.

- **Objetivo:** activar el servicio de mensajería utilizando la modalidad de audio.
- **Prerequisitos:** el usuario ha ejecutado el sistema multimodal en su función de reconocimiento de entradas.
- **Estado actual del sistema:** la tabla *Audio* contiene tres registros, relacionados a los patrones obtenidos de los comandos asociados a los servicios de mensajería, Internet y correo electrónico.
- **Proceso de prueba:**
  - El usuario presiona el botón de incremento de volumen del dispositivo móvil, para iniciar el proceso de ingreso del comando de audio "mensaje".
  - El sistema muestra al usuario el mensaje que se aprecia en la figura 5.24 e inicia la captura de la señal de audio por medio del micrófono del dispositivo móvil.

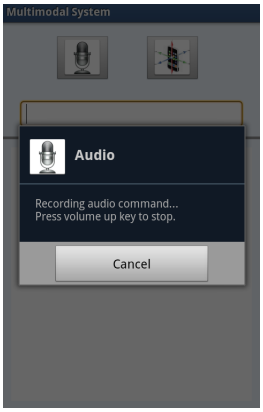


Figura 5.24: Mensaje durante la grabación del comando de audio

  - El usuario pronuncia en voz alta el comando "mensaje" que definió anteriormente para la aplicación *Messaging*.

- El sistema graba los valores que registra el micrófono.
- El usuario presiona nuevamente el botón de incremento de volumen del dispositivo móvil para finalizar la grabación de audio. Si el usuario decide abortar la grabación, puede seleccionar el botón *Cancel* del mensaje de la figura 5.24.
- El sistema libera los recursos que requirió del dispositivo para detener la grabación y obtiene un archivo que contiene los valores que representan la señal de audio de la palabra "mensaje".
- El sistema muestra un mensaje en la interfaz gráfica indicando al usuario, que el sistema recibió el comando de audio y está realizando el proceso de reconocimiento. Simultáneamente, el sistema procede a realizar un conjunto de tareas que permiten obtener el patrón asociado al comando.

  Estas tareas son las mismas que realiza el sistema cuando el usuario agrega un nuevo comando de audio, que corresponden a la etapa de preprocesamiento y extracción de características de la señal de voz. Por esta razón, omitimos el detalle de cada una de ellas ya que se obtienen resultados muy similares a los que se muestran en la subsección 5.1.1.
- Posteriormente a la obtención del patrón actual, el sistema obtiene los patrones almacenados en la tabla *Audio* y aplica el algoritmo de comparación de patrones *DTW*. Al final del proceso de comparación de patrones, se tiene un conjunto de valores que indican el grado de similaridad que existe entre el patrón actual y cada uno de los patrones previamente almacenados. El sistema obtiene el patrón cuyo valor de similaridad es menor.

  Los valores que se obtuvieron al comparar el patrón del comando "mensaje" que ingresó el usuario en un inicio, con los patrones almacenado en el sistema, se presentan en la tabla 5.2.

Tabla 5.2: Valores de la codificación de la señal de audio

Índice	Valor
0	-0.21934006
1	-0.097685635
2	-0.060309242

- El sistema accede a la información de este patrón y obtiene el identificador del tipo de servicio que tiene asociado; en este caso obtiene un identificador con valor '1', que indica su relación a una aplicación.
- El sistema pasa esta información al manejador de entradas descrito en la sección 4.6.1 y éste, accede al identificador (*id*) de la aplicación para obtener, me-

diante una consulta a la tabla *Applications*, el nombre de la aplicación que tiene asignado dicho identificador. Obtiene como resultado el nombre *Messaging*.

- Al detectar que el servicio que se desea activar corresponde al de mensajería, el sistema queda a la espera de una posible segunda entrada. En este caso, el usuario no proporciona otra entrada, por lo tanto en el momento que finaliza el tiempo de espera, el sistema continua con el proceso de activación del servicio.
- El sistema verifica si el sistema de mensajería requiere de algún tipo de conexión y obtiene que necesita de una conexión a la red celular. Dada esta circunstancia, el sistema revisa si la conexión a la red celular está disponible.
- En el proceso de verificación de disponibilidad de la conexión a la red celular se obtiene que este tipo de red tiene una intensidad de señal de nivel 4, es decir, un grado alto. Por lo tanto, la respuesta es un valor verdadero y el sistema continua con el proceso para activar el servicio de mensajería.
- El sistema obtiene de la tabla *Applications* el nombre del paquete que permite invocar la aplicación *Messaging* y ejecuta esta aplicación en el dispositivo móvil.

- **Resultados:** el usuario activa el servicio de mensajería en el dispositivo móvil, por lo que el sistema muestra la pantalla de inicio de la aplicación *Messaging* diseñada para que el usuario pueda enviar mensajes SMS (ver figura 5.25).

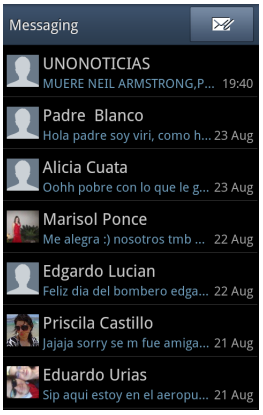


Figura 5.25: Servicio de manejo de mensajería activado a través del comando de audio

5.3.2 Modalidad de gesto táctil

Otra modalidad disponible es la de gestos táctiles, el usuario puede ingresar un comando en esta modalidad y el sistema realiza el proceso de reconocimiento para determinar a qué servicio está asociado e intentar ejecutar dicho servicio.

El sistema multimodal por medio de la interfaz de la figura 5.23, provee un área destinada al ingreso de comandos tipo gestos táctiles, ubicada en la parte inferior de la pantalla.

Cuando el usario dibuja un gesto en el área destinada para esta acción, el sistema obtiene esta entrada y procede a realizar una etapa de reconocimiento en la que intenta obtener de los patrones previamente almacenados en esta modalidad, aquél cuyo gesto táctil se asemeje en mayor medida al que el usuario acaba de dibujar.

Una de las pruebas realizadas para corroborar la funcionalidad de activar un servicio utilizando un comando de gesto táctil, está descrita a continuación:

- **Objetivo:** activar el servicio de mostrar la información del contacto *Eduardo Urias*, al cual se le asoció un comando de gesto táctil previamente.
- **Prerequisitos:** el usuario ha ejecutado el sistema multimodal en su función de reconocimiento de entradas.
- **Estado actual del sistema:** la tabla *TouchGesture* contiene tres registros, uno asociado a la aplicación de la cámara fotográfica y los otros dos, a los contactos *Eduardo Urias* y *Fabiola Ponce*, respectivamente.
- **Proceso de prueba:**
  - El usuario dibuja un gesto sobre el área delimitada por el rectangulo blanco en la interfaz gráfica, imitando el gesto táctil que definió previamente para el contacto *Eduardo Urias*, como se muestra en la figura 5.26.

Figura 5.26: Entrada de comando en la modalidad de gesto táctil

  - El sistema detecta cuando el usuario termina de dibujar el gesto y obtiene esta entrada.
  - Utilizando los valores que conforman el trazo del gesto táctil, el sistema define un patrón y lo compara con los patrones de gestos táctiles almacenados anteriormente en el sistema multimodal.

- El sistema realiza el reconocimiento del gesto táctil, como se describe en la sección 4.3.3, en el que se utiliza un algoritmo para realizar la comparación de los patrones y da como resultado una lista ordenada de manera descendente con respecto al grado de similaridad del gesto táctil de entrada y los patrones almacenados en el sistema.

  El primer elemento de la lista, es decir, el de mayor grado de similaridad, corresponde al patrón del que el sistema obtiene la información para determinar el tipo de servicio al que está asociada la entrada del usuario. Los valores que contiene la lista para este caso de prueba se presentan en la tabla 5.3.

Tabla 5.3: Valores obtenidos al comparar los patrones de gestos táctiles

Índice	Valor
0	6.600714533787998
1	1.5486694279033832
2	0.5231439822004244

- El sistema determina que el patrón más similar corresponde al tipo de servicio relacionado a un contacto y obtiene el identificador asociado a dicho contacto.
- El sistema pasa esta información al manejador de entradas y éste utiliza el identificador del contacto para mostrar la información que está registrada en el dispositivo móvil para el contacto *Eduardo Urias.*

- **Resultados:** en la pantalla del dispositivo móvil se muestra la información del contacto *Eduardo Urias*, tal como se aprecia en la figura 5.27.

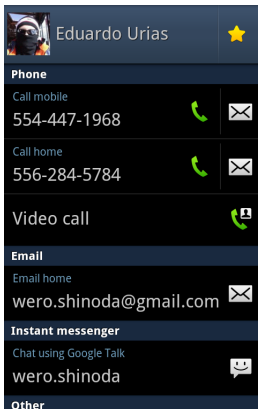


Figura 5.27: Información del contacto activada a través de un comando de gesto táctil

5.3.3 Modalidad de gesto físico

Al utilizar la modalidad de gesto físico para activar un servicio, el usuario describe un movimiento específico con el dispositivo móvil y después de que el sistema realiza un procesamiento de este gesto, ejecuta el servico al cual está asociado.

El usuario para indicar al sistema el inicio de la captura de un gesto físico, dispone de dos posibles acciones: seleccionar el botón  , disponible en la interfaz gráfica o presionar el botón de disminuir volumen del dispositivo móvil.

Cuando el sistema detecta que ha recibido un comando gesto físico, obtiene los valores que registro el acelerómetro del dispositivo durante la descripción de dicho gesto. Posteriormente, realiza una serie de tareas para obtener un patrón a partir de dichos valores y compara este patrón con los que se encuentran almacenados en el sistema.

El sistema, al determinar el patrón más similar al que define el gesto físico que el usuario ingresó, obtiene la información sobre el servicio que debe activar en el dispositivo móvil.

Una de las pruebas que se realizaron para validar esta funcionalidad, consistió en activar el servicio de calculadora utilizando el comando de gesto físico que se agregó en la prueba que se describe en la sección 5.1.3.

- **Objetivo:** activar el servicio de calculadora del dispositivo móvil utilizando la modalidad de gesto físico.
- **Prerequisitos:** el usuario ha ejecutado el sistema multimodal en su función de reconocimiento de entradas.
- **Estado actual del sistema:** la tabla *PhysicalGesture* contiene tres registros asociados a las aplicaciones calculadora, reproductor de música y lista de contactos.
- **Proceso de prueba:**
  - El usuario presiona el botón  en la interfaz gráfica, para ingresar el gesto físico a reconocer.
  - En la interfaz gráfica se presenta una ventana con el mensaje que se aprecia en la figura 5.28, para informar al usuario que el sistema ha iniciado la captura del gesto físico. Adicionalmente provee los botones *Done* y *Cancel* para que el usuario indique al sistema el momento en que ha terminado de ingresar el gesto o si desea cancelar la captura, respectivamente.
  - El usuario describe con el movimiento del dispositivo móvil, el gesto predefinido para el servicio de calculadora.
  - Mientras tanto, el sistema obtiene los valores que registra el acelerómetro.

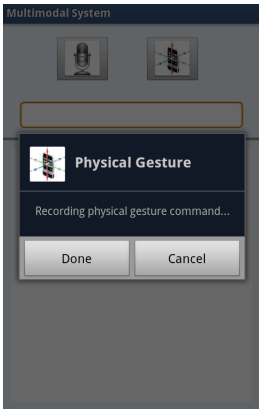


Figura 5.28: Mensaje durante la grabación del comando de gesto físico

- Cuando el usuario termina de describir el gesto físico, presiona el botón *Done* para indicar al sistema el fin del gesto.
- El sistema deja de registrar los valores del acelerómetro y obtiene los que registro durante la definición del gesto.
- El sistema muestra un mensaje en la interfaz gráfica indicando al usuario que el sistema recibió el comando de gesto físico y está realizando el proceso de reconocimiento. Al mismo tiempo, el sistema realiza las tareas que le permiten obtener un patrón representativo del gesto físico que el usuario ingresó. Estas tareas corresponden a las que realiza el sistema cuando el usuario agrega un nuevo comando de gesto físico (obtener la señal de aceleración y suavizado de la señal). Los resultados que se obtienen al realizar estas tareas, son muy semejantes a los que se presentarón en la subseccion 5.1.3, debido a que el gesto físico de entrada es una imitación del gesto que se agregó anteriormente para el servicio de calculadora.
- El sistema obtiene los patrones almacenados en la tabla *PhysicalGesture* y aplica el algoritmo que le permite reconocer cual de estos patrones es el más semejante al patrón del gesto de entrada.

  Durante el proceso de reconocimiento, el algoritmo calcula los valores de similaridad entre el patrón de entrada y los que están almacenados en el sistema, para determinar a qué patron corresponde el menor valor. En la tabla 5.4 se aprecian los resultados de este proceso.
- El sistema obtiene el patrón con identificador 6, accede a su información y obtiene el identificador del tipo de servicio que tiene asociado; en este caso obtiene un identificador con valor '1', lo que indica que corresponde a una aplicación.

Tabla 5.4: Valores obtenidos en el proceso de comparación de patrones de gestos físicos

Identificador	Servicio asociado	Valor
6	Calculadora	63.047867
5	Contactos	66.03526
3	Música	105.48175

- El sistema pasa esta información al manejador de entradas y este accede al identificador (*id*) de la aplicación para obtener mediante una consulta a la tabla *Applications* el nombre de la aplicación que tiene asignado dicho identificador. En este caso obtiene el nombre de *Calculator*.

  Con el nombre de la aplicación, el sistema verifica si esta aplicación requiere de algún tipo de conectividad. Dado que se trata del servicio de calculadora, obtiene un valor *NONE*, el cual indica al sistema que la aplicación especificada no requiere conexión a la red celular ni a alguna red inalámbrica.

- El sistema obtiene de la tabla *Applications* el nombre del paquete que permite activar la aplicación *Calculator* y utiliza esta información para ejecutar este servicio en el dispositivo móvil.

- **Resultados:** el sistema realizó el proceso de reconocimiento del gesto físico de entrada de manera adecuada y esto permitió que activara el servicio de calculadora. El usuario visualiza en el dispositivo móvil la aplicación predeterminada para mostrar una calculadora, como se aprecia en la figura 5.29.

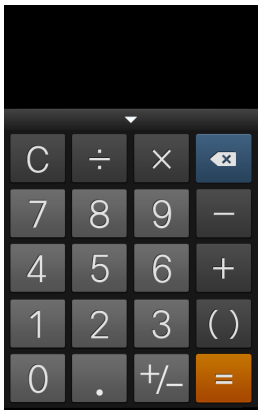


Figura 5.29: Servicio de calculadora activado por medio del un comando de gesto físico

5.3.4 Modalidad de texto

El sistema multimodal en su función de reconocimiento de entradas, permite al usuario ingresar un comando en la modalidad de texto para ejecutar algún servicio disponible en el dispositivo.

Para realizar esta función, el sistema multimodal obtiene el comando de entrada que el usuario ingresa a través del cuadro de captura de texto que provee la interfaz gráfica que se aprecia en la figura 5.23 y consulta la tabla *Text* del sistema para buscar alguna coincidencia con dicho comando. Conforme el usuario ingresa los caracteres del comando de texto, el sistema de manera proactiva muestra en una lista los nombres de servicios que tienen asociado un comando que puede coincidir al que se está brindando como entrada.

El usuario selecciona de la lista el nombre del servicio que desea activar en el dispositivo móvil.

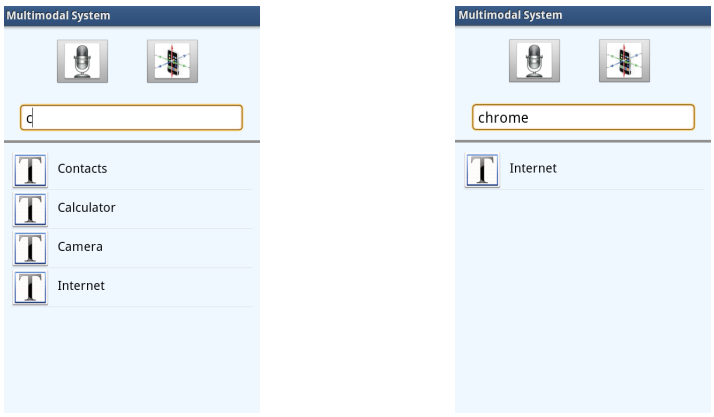
A continuación se detalla la prueba que consiste en activar el servicio de navegador de Internet al ingresar el comando de texto que se agregó para este fin en la sección 5.1.4.

- **Objetivo:** activar el servicio de navegador de Internet en el dispositivo móvil utilizando un comando de texto.
- **Prerequisitos:** el usuario ha ejecutado el sistema multimodal en su función de reconocimiento de entradas.
- **Estado actual del sistema:** la tabla *Text* contiene seis registros de comandos de texto, uno de ellos asociado a un contacto y el resto, a los servicios de reproductor de música, calculadora, lista de contactos, cámara fotográfica y navegador de Internet. En la figura 5.19 se pueden apreciar estos comandos.

  El dispositivo móvil tiene desactivada la opción de conectividad a redes inalámbricas.
- **Proceso de prueba:**
  - El usuario selecciona el cuadro de captura de texto de la interfaz gráfica 5.23 y empieza a ingesar caracteres.
  - Cada vez que el usuario ingresa o borra algún carácter en el texto que está capturando, el sistema realiza una consulta a la tabla *Text* de la base de datos para obtener los nombres de los servicios que tienen un comando de texto similar al que hasta el momento está capturado.
  - El sistema muestra una lista con los resultados de la consulta a través de la interfaz gráfica.

- Cuando en la lista de servicios que presenta el sistema, aparece aquél que el usuario desea activar, de ser el caso, puede dejar de ingresar los caracteres faltantes para completar el comando de texto que tiene asociado y seleccionar el nombre del servicio para su ejecución.

  En esta caso de prueba, el comando asociado al servicio de navegador de internet corresponde a la palabra "chrome", por lo tanto el usuario ingresó el carácter 'c'. En ese momento el sistema presentó la lista de servicios que se aprecia en la figura 5.30 a). Como se puede observar, aparece el nombre de cuatro aplicaciones cuyos comando de texto asociados tienen una coincidencia con la letra 'c', entre ellos aparece la aplicación de Internet a la cual está relacionado el servicio de navegador de Internet; el usuario puede seleccionar de esta lista dicha aplicación o puede continuar ingresando los caracteres que falta para completar el comando de texto. La figura 5.30 b) presenta la lista que muestra la interfaz gráfica cuando el usuario terminó de ingresar el comando "chrome".



a) Captura del carácter 'c' b) Captura del comando "chrome"

Figura 5.30: Respuesta del sistema al ingresar un comando de texto

- En el momento que el usuario selecciona de la lista de resultados la opción Internet el sistema, a través del manejador de entradas, verifica si el servicio requiere de algún tipo de conectividad. Esta validación da como resultado que la aplicación de Internet requiere de una conexión a la red inalámbrica.

  Dado que el dispositivo móvil no está conectado a ninguna red inalámbrica, el sistema proactivamente realiza las siguientes acciones para intentar establecer alguna conexión para poder activar el servicio solicitado:

  - Verifica si la opción de conectividad a redes inalámbricas está activada en el dispositivo móvil. Al realizar esta validación el sistema obtiene un va-

lor falso, por lo tanto, activa esta opción para que el dispositivo intente conectarse a alguna de las redes predeterminadas (aquellas a las que anteriormente se ha conectado).

- En este caso de prueba, no se encuentra ninguna red predeterminada al alcance, por lo que el sistema obtiene una lista de redes disponibles y a partir de ésta, obtiene las redes que son abiertas (sin seguridad) y su nivel de intensidad de señal. La tabla 5.5 muestra los resultados obtenidos al buscar las redes al alcance del dispositivo móvil en el momento que se realizó esta prueba.

Tabla 5.5: Redes inalámbricas disponibles

Nombre	Seguridad	Intensidad de señal
TotalPlay6505	WPA	3
arris54g	abierta	2
garrido	WPA	2
INFINITUM3566	WEP	2
INFINITUM5c09	WEP	2
IXBA	WPA	2
INFINITUM8528	WEP	1
linksys	abierta	1
MOTOROLA-9070D	WPA	1

- El sistema obtiene una lista ordenada con base a la intensidad de señal de las redes abiertas disponibles e intenta establecer una conexión a alguna de ellas en ese orden. En este caso, el sistema logró conectarse a la red *arris54g*.
- El manejador de entradas es notificado que ya se cuenta con una conexión a una red inalámbrica.

- Una vez que se cuenta con la conectividad necesaria, el sistema obtiene de la tabla *Applications* el nombre del paquete que permite activar la aplicación *Internet* y utiliza esta información para ejecutar este servicio en el dispositivo móvil.

  Si el sistema no logra establecer la conexión requerida por la aplicación, informa al usuario que no es posible activar el servicio solicitado por la falta de conectividad a una red inalámbrica.

- **Resultados:** el sistema logra establecer una conexión a la red abierta *arris54g*, disponible al alcance del dispositivo móvil en el momento que el usuario desea activar el servicio de navegador de Internet. Gracias a esto, el sistema presenta al usuario en pantalla la aplicación *Internet*, como se presenta en la figura 5.31.

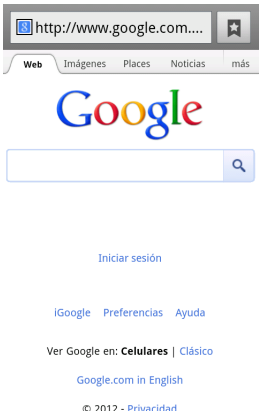


Figura 5.31: Servicio de navegador de Internet al utilizar un comando de texto

5.3.5 Combinación de modalidades

El sistema multimodal provee la funcionalidad de activar los servicios de enviar un mensaje o marcar el número telefónico de un contacto en específico. Para esto, el usuario debe ingresar dos comandos en forma consecutiva, a través de la interfaz que provee el sistema multimodal en su función de reconocimiento de entradas (ver figura 5.23).

El usuario puede ingresar estos dos comandos utilizando la misma modalidad para ambos o una combinación de modalidades, es decir, que el primer comando puede ser proporcionado en una modalidad y el segundo en una diferente. Cualquier combinación de modalidades es aceptada por el sistema.

Para corroborar esta funcionalidad se realizarón varias pruebas utilizando diferentes combinaciones de modalidades, entre ellas la combinación de un comando de audio y uno de gesto táctil, para activar el servicio de llamar a un contacto.

- **Objetivo:** activar el servico de marcar al contacto *Fabiola Ponce* utilizando la combinación de un comando de audio y un comando de gesto táctil.

- **Prerequisitos:** el usuario ha ejecutado el sistema multimodal en su función de reconocimiento de entradas.

- **Estado actual del sistema:** la tabla *Audio* contiene entre sus registros, uno asociado al servicio de marcardor teléfonico.

  La tabla *TouchGesture* tiene un registro asociado al contacto *Fabiola Ponce*.

- **Proceso de prueba:**

  - El usuario presiona el botón  en la interfaz gráfica que presenta el sistema multimoda (ver figura 5.23), para iniciar el proceso de ingreso del comando de audio “llamar”.
  - El sistema muestra al usuario el mensaje que se aprecia en la figura 5.32 e inicia la captura de la señal de audio por medio del micrófono del dispositivo móvil.

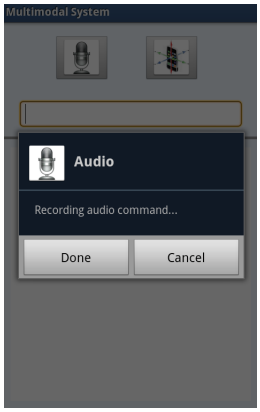


Figura 5.32: Mensaje durante la grabación del comando de audio

  - El usuario pronuncia en voz alta el comando “llamar” que definió anteriormente para el servicio de marcador telefónico.
  - El sistema registra los valores que capta el micrófono durante la grabación.
  - El usuario presiona el botón *Done* del mensaje que se aprecia en la figura 5.32, para indicar al sistema el fin de la grabación de audio. Si el usuario decide anular la grabación puede seleccionar el botón *Cancel* que se muestra en el mismo mensaje.
  - El sistema detiene la grabación del comando de audio y obtiene un archivo que contiene los valores que representan la señal de audio de la palabra “llamar”.
  - El sistema muestra un mensaje en la interfaz gráfica indicando al usuario que ha recibido el comando de audio y está realizando el proceso de reconocimiento. Al mismo tiempo, el sistema realiza las tareas de preprocesamiento y extracción de características para obtener el patrón del comando actual.

    Debido a que estas tareas se han explicado a detalle en secciones anteriores, en esta prueba no son detalladas.
  - El sistema realiza una comparación entre el patrón actual y cada uno de los almacenados en el sistema en la modalidad de audio.

- El sistema accede a la información del patrón que obtuvó un valor menor en la comparación y al acceder a la información del tipo de servicio al que está asociado, se obtiene un identificador con valor '1' que indica su relación a una aplicación.
- El sistema pasa esta información al manejador de entradas y éste, consulta con el identificador de la aplicación el nombre de la aplicación correspondiente al patrón y como resultado obtiene el nombre de *Dialer*. La aplicación *Dialer* está asociada al servicio de marcador telefónico.
- Al detectar que el servicio que desea activar corresponde al del marcador telefónico, el sistema queda a la espera de una posible segunda entrada.
- Mientras transcurre el tiempo de espera, el usuario ingresa un nuevo comando utilizando la modalidad de gesto táctil imitando el que definió para el contacto *Fabiola Ponce*, como se muestra en la figura 5.33.

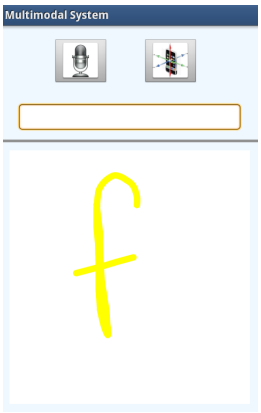


Figura 5.33: Comando de gesto táctil del contacto Fabiola Ponce

- El sistema lleva a cabo el reconocimiento de este último comando de gesto táctil y determina que está asociado al contacto con identificador 1204. Esta información se transmite al manejador de entradas.
- El sistema detecta el ingreso de una segunda entrada asociada a un contacto cuando aún no ha expirado el tiempo de espera, por lo tanto, procesa ambas entradas para activar el servicio de marcar al contacto *Fabiola Ponce*.

  En caso de que el sistema reciba como segunda entrada un comando asociado a una aplicación, esta entrada es ignorada y el servicio que está asociado a la primera entrada es activado al finalizar el tiempo de espera.

- **Resultados:** el sistema realiza un correcto reconocimiento de cada uno de los comando que se propocionaron como entrada y activa el servicio de marcar el número telefónico asociado al contacto *Fabiola Ponce*, como se observa en la figura 5.34.

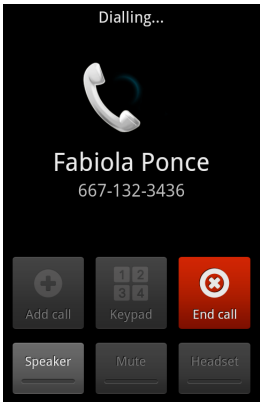


Figura 5.34: Activación del servicio de marcar a un contacto específico

Capítulo 6

Conclusiones y trabajo futuro

En la primera sección de este capítulo se presentan las conclusiones obtenidas al final de este trabajo de investigación, tomando en cuenta los objetivos planteados en un inicio y los requisitos funcionales y de calidad establecidos en la propuesta de solución.

En la segunda sección se plantean algunas propuestas de trabajo a futuro que enriquecerían en mayor medida este trabajo.

6.1 Conclusiones

Durante el desarrollo del presente trabajo de tesis se creó un sistema multimodal para dispositivos móviles Android, con la finalidad de activar algunos de los servicios disponibles en el dispositivo móvil utilizando diferentes comandos en las modalidades de audio, gesto táctil, gesto físico y texto.

Los servicios que permite activar el sistema a través de las modalidades mencionadas, corresponden a la ejecución de las principales aplicaciones que utiliza un usuario en los dispositivos móviles denominados *smartphones*: marcador telefónico, navegador de Internet, reproductor de música, manejador de mensajes, lista de contactos, manejador de correo electrónico, cámara fotográfica y calculadora. Además de los servicios de mostrar la información de un contacto y enviar un mensaje o marcar el número telefónico de un contacto en específico, esto mediante el uso de una combinación de modalidades. Esta función puede extenderse para activar cualquier aplicación que se encuentre instalada en el dispositivo móvil, incluyendo aplicaciones desarrolladas para un fin específico.

Se diseñaron diversas interfaces que permiten al usuario interactuar con el sistema de una manera sencilla e intuitiva. Estas interfaces se pueden dividir en dos conjuntos principales: el primero corresponde a aquellas que permiten entrenar el sistema, es decir, brindan al usuario la funcionalidad de agregar nuevos comandos al sistema y asociarlos a un servicio disponible en el dispositivo móvil; el segundo conjunto corresponde a las

interfaces que fueron diseñadas para que el usuario maneje el ingreso de los comandos en las diferentes modalidades para su reconocimiento.

Se implementaron un conjunto de módulos de software que permiten al sistema recibir, procesar y dar respuesta a cada una de las entradas que el usuario proporciona al sistema. Además de un módulo que monitorea el contexto de conectividad del dispositivo móvil.

En particular, los módulos de texto y análisis de contexto brindan un cierto grado de proactividad al sistema multimodal. El primero de ellos, al proponer al usuario los servicios que pueden coincidir con el comando de texto, conforme se encuentra ingresando los caracteres que lo conforman. El segundo, al realizar las tareas de monitoreo de las redes celular e inalámbricas, obtener el estado de conectividad y tratar de establecer una conexión (si no existe), en caso de que el usuario desee activar un servicio que requiera una conexión a alguna de estas redes.

El mecanismo de activación del sistema multimodal en su función de reconocimiento de entradas, definido por un movimiento de sacudida del dispositivo móvil (*shake*), permite al usuario acceder al sistema de una forma rápida, práctica y sencilla, al no requerir que el usuario realice un proceso de búsqueda de la aplicación en el dispositivo móvil.

El sistema multimodal utiliza el micrófono para capturar un comando de audio, la pantalla táctil para obtener un comando de gesto táctil, el acelerómetro para adquirir un comando de gesto físico e implementar el mecanismo de activación del sistema en su función de reconocimiento y el teclado para el ingreso de un comando de texto. De esta forma el sistema explota las capacidades de los diferentes sensores embebidos en el dispositivo móvil.

Con base en los resultados que arrojaron cada una de las pruebas realizadas al sistema, se puede decir que el sistema multimodal funciona correctamente en un dispositivo móvil Android y permite activar la ejecución de los servicios disponibles mencionados al inicio de esta sección.

Vale la pena destacar que todo el procesamiento que requiere el sistema multimodal se realiza en el dispositivo móvil. Esta característica marca una gran diferencia con respecto a los proyectos que se encuentran en la literatura, ya que por lo general delegan la tarea de cómputo a un servidor, razón que convierte a la aplicación dependiente a una conexión a Internet para su uso.

Otro aspecto importante que aporta este proyecto, es brindar el acceso a los principales servicios que dispone un dispositivo de avanzada tecnología, como lo es un *smartphone*, a personas con capaciades diferentes que sufren de algún impedimento que no les permita interactuar de la forma habitual con estos dispositivos. Esto gracias a que el sistema que se desarrolló permite activar dichos servicios a través de diferentes modalidades. Así una persona que por ejemplo, tenga un impedimiento motríz y por lo tanto, no sea capaz

controlar de forma adecuada sus movimientos para manipular un *smartphone*, que por lo general presenta una interfaz táctil, puede utilizar la modalidad de audio para activar el servicio que desee en dicho dispositivo.

6.2 Trabajo a futuro

A continuación se presentan algunas propuesta que pueden ser consideradas como continuidad a este proyecto para aportar una mayor funcionalidad y usabilidad:

- Implementar el sistema multimodal para otros sistemas operativos como *iOS* y *Windows mobile*, para que sea utilizado en una mayor gama de dispositivos móviles.
- Incluir nuevas modalidades al sistema, por ejemplo, gestos corporales como expresiones faciales, mímica y movimientos del ojo. Para esto se puede utilizar la cámara fotográfica que actualmente todos los dispositivos *smartphones* integran.
- Proporcionar un mecanismo de retroalimentación al usuario de las acciones que realiza en el sistema. Una forma podría ser por medio de mensajes de voz o diferentes vibraciones del dispositivo móvil.
- Brindar al usuario la función de seleccionar cualquier servicio disponible en el dispositivo para asociarle un comando en alguna modalidad.
- Realizar un reconocimiento independiente del usuario para los comandos de audio y gestos físicos. Es decir, que el usuario no requiera realizar una etapa de entrenamiento.
- Guiar al usuario en el proceso de entrenamiento del sistema. Esto debido a que actualmente, un usuario con algún impedimento visual requiere del apoyo de alguien más para poder definir los comandos en las diferentes modalidades en el sistema.
- Probar otros algoritmos de extracción de características y reconocimiento de patrones. Existe una gran variedad de algoritmos de este tipo, que sería conveniente probar para realizar una comparación con el algoritmo que implementa actualmente el sistema multimodal. Esto podría corroborar que el algoritmo *DTW* es el apropiado para esta aplicación o si sería preferible implementar algún otro.
- Desarrollar el sistema multimodal para otras plataformas como *tablets*, computadoras portátiles y de escritorio.
- Implementar los mecanismos que se requieran para que el usuario pueda ingresar comandos en las modalidades de audio y gestos físicos, sin la necesidad de interactuar con una interfaz gráfica o presionar algún botón.

- Realizar un estudio en el que participen usuarios de diferentes edades y con alguna capacidad diferente, para evaluar cualitativamente el sistema.

Bibliografía

[1] P. Zheng and L. Ni, *Smart phone and next generation mobile computing*. Morgan Kaufmann Series in Networking, USA: Elsevier, 1 ed., 2006.

[2] S. Dasbit and B. K. Sikdar, *Mobile computing*. New Delhi, India: PHI Learning, 1 ed., 2009.

[3] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, pp. 140–150, September 2010.

[4] E. Milluzo, *Smartphone sensing*. PhD thesis, Dartmouth College, June 2011.

[5] R. Want, "When cell phones become computers," *Pervasive Computing*, vol. 8, pp. 2–5, April-June 2009.

[6] R. Ballagas, J. Borchers, M. Rohs, and J. Sheridan, "The smart phone: a ubiquitous input device," *Pervasive Computing, IEEE*, vol. 5, pp. 70–77, January-March 2006.

[7] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, "Identifying diverse usage behaviors of smartphone apps," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement*, IMC '11, (New York, NY, USA), pp. 329–344, ACM, November 2-4 2011.

[8] J. Liao and Q. Cao, "Uno: a sharing infrastructure for smartphone sensors and files," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, (New York, NY, USA), pp. 393–394, ACM, November 1-4 2011.

[9] R. Llamas, K. Restivo, and M. Shirer, "Smartphone market hits all-time quarterly high due to seasonal strength and wider variety of offerings, according to idc," *International Data Corporation*, February 06 2012.

[10] R. Llamas, K. Restivo, and M. Shirer, "Worldwide smartphone market continues to soar, carrying samsung into the top position in total mobile phone and smartphone shipments, according to idc," *International Data Corporation*, May 01 2012.

[11] R. Llamas, K. Restivo, and M. Shirer, "Strong demand for smartphones in second quarter continues to drive the worldwide mobile phone market, according to idc," *International Data Corporation*, July 26 2012.

[12] J. John and C. Riddle, "Smart phone power," in *Proceedings of the 47th Design Automation Conference*, (Anaheim, CA, USA), pp. 935–936, ACM, June 2010.

[13] D. Gavalas and D. Economou, "Development platforms for mobile applications: Status and trends," *IEEE Software*, vol. 28, pp. 77–86, January-February 2011.

[14] R. Cameron, *Pro Windows Phone App Development*. USA: Apress, 1 ed., 2011.

[15] S. P. Hall and E. Anderson, "Operating systems for mobile computing," *Journal of Computer Science in Colleges*, vol. 25, pp. 64–71, December 2009.

[16] A. S. Tanenbaum, *Modern Operating Systems*. USA: Pearson Education, 3 ed., 2009.

[17] R. Llamas, K. Restivo, and M. Shirer, "Android and ios surge to new smartphone os record in second quarter, according to idc," *International Data Corporation*, August 2012.

[18] X. Li, P. Ortiz, J. Browne, D. Franklin, J. Oliver, R. Geyer, Y. Zhou, and F. Chong, "Smartphone evolution and reuse: Establishing a more sustainable model," in *Proceedings of the 39th International Conference on Parallel Processing Workshops*, (San Diego, CA, USA), pp. 476 –484, IEEE Computer Society, september 13-16 2010.

[19] G. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, pp. 114–117, April 1965.

[20] R. Asad-Uj-Jaman, "Sensors in smartphones," *Mobile Device Insight*, December 01 2011.

[21] J. Rouillard, "Multimodal and multichannel issues in pervasive and ubiquitous computing," in *Multimodality in Mobile Computing and Mobile Devices: Methods for Adaptable Usability* (S. Kurkovsky, ed.), ch. 1, pp. 1–23, Idea Group, Inc., 2010.

[22] R. Englert and G. Glass, "An architecture for multimodal mobile applications," in *Proceedings of the 20th Symposium on Human Factor in Telecomunication (HFT 2006)*, (Sophia Antipolis, France), ETSI, March 20-23 2006.

[23] J. Chang and M.-L. Bourguet, "A usability framework for the design and evaluation of multimodal interaction: Application to a multimodal mobile phone," in *Multimodality in Mobile Computing and Mobile Devices: Methods for Adaptable Usability* (S. Kurkovsky, ed.), ch. 8, pp. 196–215, Idea Group, Inc., 2010.

[24] S. Oviatt and P. Cohen, "Perceptual user interfaces: multimodal interfaces that process what comes naturally," *Communications of the ACM*, vol. 43, pp. 45–53, March 2000.

[25] S. Oviatt, T. Darrell, and M. Flickner, "Multimodal interfaces that flex, adapt, and persist," *Communications of the ACM*, vol. 47, pp. 30–33, January 2004.

[26] P. Ehlen and M. Johnston, "Multimodal local search in speak4it," in *Proceedings of the 16th international conference on Intelligent user interfaces*, IUI '11, (Palo Alto, CA, USA), pp. 435–436, ACM, February 13-16 2011.

[27] A. Acero, N. Bernstein, R. Chambers, Y. Ju, X. Li, J. Odell, P. Nguyen, O. Scholz, and G. Zweig, "Live search for mobile: Web services by voice on the cellphone," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, (Las Vegas, Nevada, USA), pp. 5256 –5259, IEEE Computer Society, March 30-April 4 2008.

[28] M. Turunen, J. Hakulinen, A. Kainulainen, A. Melto, and T. Hurtig, "Design of a rich multimodal interface for mobile spoken route guidance," in *INTERSPEECH '07*, (Antwerp, Belgium), pp. 2193–2196, August 27-31 2007.

[29] J. Papaj, M. Pleva, A. Cizmar, L. Dobos, J. Juhar, and S. Ondas, "Mobitel- mobile multimodal telecommunications systems and services," in *Proceedings of the 17th International Conference Radioelektronika*, (Brno, Czech Republic), IEEE Computer Society, April 24-25 2007.

[30] G. Frattini, F. Gaudino, and V. S. di Carlo, "Mobile multimodal applications on mass-market devices: experiences," in *Proceedings of the 18th International Conference on Database and Expert Systems Applications*, (Regensburg, Germany), pp. 89–93, IEEE Computer Society, September 3-7 2007.

[31] T. Hurtig, "A mobile multimodal dialogue system for public transportation navigation evaluated," in *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, MobileHCI '06, (Espoo, Finland), pp. 251–254, ACM, September 12-15 2006.

[32] A. Gentile, A. Santangelo, S. Sorce, A. Augello, G. Pilato, A. Geneo, and S. Gaglio, "Exploiting multimodality for intelligent mobile access to pervasive services in cultural heritage sites," in *Multimodality in Mobile Computing and Mobile Devices: Methods for Adaptable Usability* (S. Kurkovsky, ed.), ch. 9, pp. 217–240, Idea Group, Inc., 2010.

[33] J.-C. Lin and Y.-m. Yeh, "A research on telephone-supported multimodal accessible website," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 2, (Taichung, Taiwan), pp. 118–123, IEEE Computer Society, June 5-7 2006.

[34] M. Malenke, M. Baumler, and E. Paulus, "Speech recognition performance assessment„" in *Verbmobil: Foundations of Speech-to-Speech Translation, Artificial Intelligence* (W. Wahlster, ed.), pp. 583–591, Germany: Springer, 2000.

[35] S. Sakti, K. Markov, S. Nakamura, and W. Minker, *Incorporating knowledge sources into statistical speech recognition*, vol. 42. USA: Springer, 2009.

[36] D. AbuZeina and M. Elshafei, *Cross-word modeling for arabic speech recognition*. Springerbriefs in Electrical and Computer Engineering: Springerbriefs in Speech Technology, USA: Springer, 2011.

[37] G. Saha, S. Chakroborty, and S. Senapati, "A new silence removal and endpoint detection algorithm for speech and speaker recognition applications," in *Proceddings of the 17th National Conference on Communications*, (Kharagpur, India), pp. 291–295, Indian Institute of Technology, January 28-30 2005.

[38] G. McLachlan, "Mahalanobis distance," *Resonance Journal of Science Education*, vol. 4, pp. 20–26, June 1999.

[39] R. W. Hamming, *Digital filters*. USA: Courier Dover Publications, 3 ed., 1998.

[40] D. O'Shaughnessy, "Linear predictive coding," *Potentials*, vol. 7, pp. 29–32, February 1988.

[41] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, pp. 43–49, February 1978.

[42] M. Muller, *Information Retrieval for Music and Motion*. USA: Springer, 2 ed., 2010.