Universidad de las Ciencias Informáticas
Facultad 4

Título: “Sistema de filtrado de SMS para los comentaristas deportivos cubanos”

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Jorge Antonio Cabrera Cruz

Tutores:

MsC. Yerandy Manso Guerra

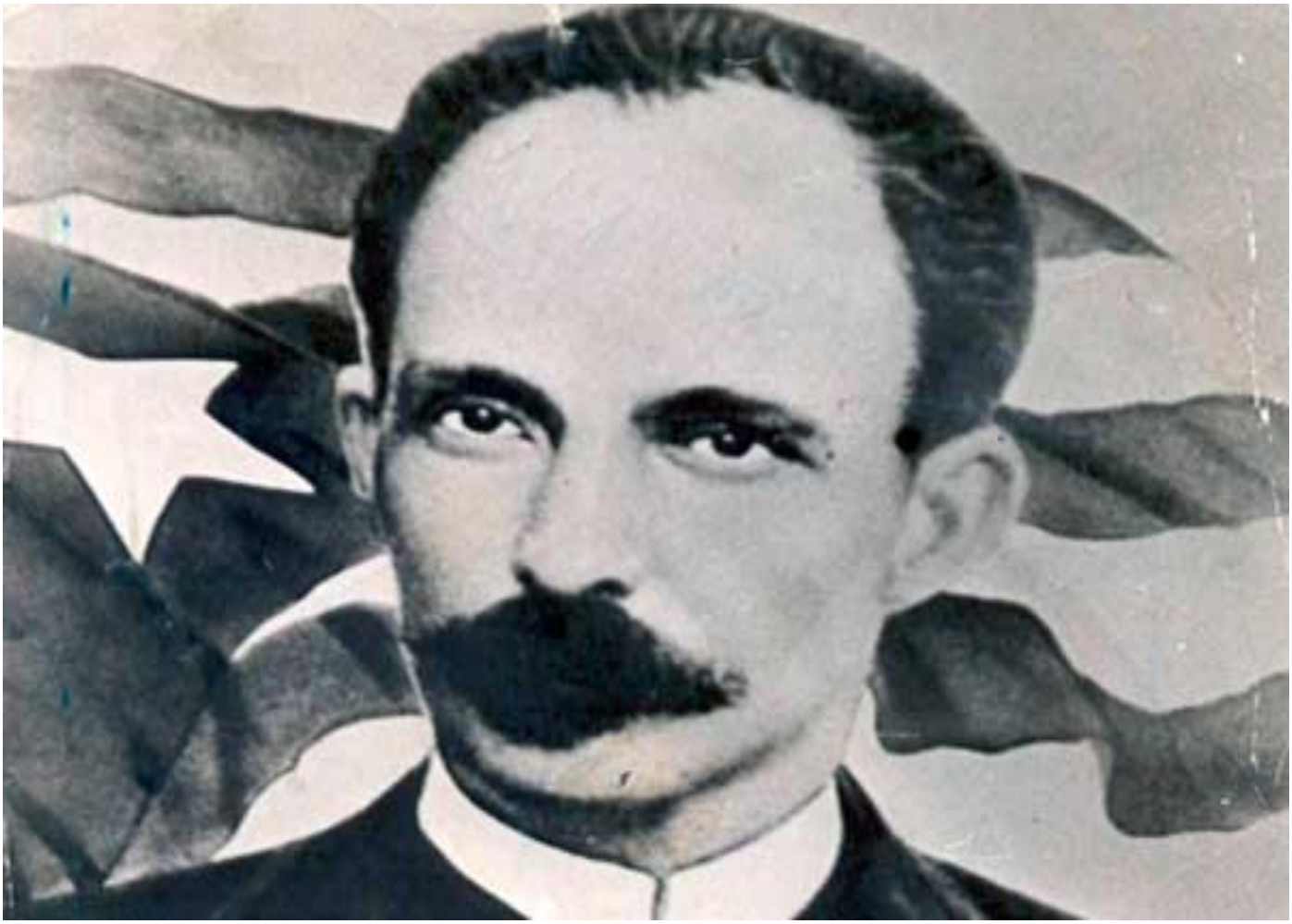
Ing. Noralys Almeida Milanés

La Habana, 2018.

Año 60 de la Revolución.

“Los derechos se toman, no se piden; se arrancan, no se mendigan.”

José Martí



Declaración de Autoría

Declaro que soy el único autor del trabajo titulado:

“Sistema de filtrado de SMS para los comentaristas deportivos cubanos” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____

Jorge Antonio Cabrera Cruz

Autor

_____ _____

Ing. Noralys Almeida Milanés **MsC. Yerandy Manso Guerra**

Tutor **Tutor**

Agradecimientos

Son muchas las personas a las que agradecer, pero primero quisiera darle gracias a Dios por permitirme contar con esas maravillosas personas que puso en mi camino.

Gracias a:

A mis padres por el gran apoyo y por darme tantas fuerzas para levantarme siempre y seguir adelante, todos mis logros se los debo y dedico a ustedes.

A mi hermano Brayan por estar ahí siempre que lo necesito, estoy muy orgulloso de ser tu hermano.

A Pulpa por ser esa segunda madre tan especial con la que también puedo contar siempre.

A tía Nery, Laurita y familia, por la gran ayuda estos 5 años de mi vida.

A mi pareja Yosleidy por entenderme y apoyarme siempre, y por ser una gran compañera en estos últimos años.

A mi tutor Yerandy por guiarme y aconsejarme durante todo este proceso de tesis.
A Reiman por ese apoyo incondicional durante la carrera, cada vez que lo necesité nunca dijo un no como respuesta.

A Leduán por ayudarme y aclararme tantas dudas cuando lo necesité.

A Randy por ser ese gran amigo en esta importante etapa de mi vida, que es la universidad.

A Raymel por ser mi Dj, compañero de Táctik Ofensiva, y brindarme esa amistad tan transparente, como la nuestra quedan pocas.

A todos mis amigos y compañeros de aula por compartir tantos momentos juntos.

Y muchas gracias a todos los presentes.

Dedicatoria

Dedicado a mis padres y mi hermano Brayan que han sabido sacrificarse toda una vida para que hoy yo sea ingeniero.

Resumen

Con el desarrollo de las Tecnologías de la Información y la Comunicación(TIC), se ha propiciado un incremento en el uso de los teléfonos celulares por las personas. En Cuba se transmiten varios programas deportivos en la televisión, en ellos los comentaristas deportivos realizan debates, dan noticias de interés nacionales o internacionales y reciben una gran cantidad de SMS por los aficionados. Para facilitar la organización de los SMS recibidos, se hace necesario el desarrollo de esta investigación. Con el sistema de filtrado de SMS para los comentaristas deportivos cubanos se contribuye a una búsqueda rápida de un SMS determinado que se quiera consultar. Para el desarrollo del sistema se utilizó la metodología AUP, en su variante UCI y la siguiente base tecnológica: como motor de base de datos HSQL versión 2.4.0, como entorno de desarrollo IntelliJ IDEA versión 2017.2.2 y AndroidStudio en su versión 3.0, además de Visual Paradigm 8.0 como herramienta para el modelado. Se realizaron pruebas de caja blanca y de caja negra para garantizar el correcto funcionamiento del sistema.

Palabras claves: SMS, filtrado, comentarista deportivo, aficionados, teléfonos celulares.

Índice

Índice de figuras

Índice de tablas

Introducción

En la actualidad las Tecnologías de la Información y la Comunicación (TIC) han provocado un gran impacto en el ámbito social, económico, político y cultural, pues no existen barreras para acceder a la información. Se está viviendo una época de innovaciones profundas que implican cambios en todas las dimensiones de la realidad. En tal sentido, puede señalarse que las TIC son la base de un nuevo tipo de relaciones: las relaciones de red (Julio, 2016).

La generalización del uso de las TIC en el mundo se manifiesta a través de su presencia en la mayoría de las actividades y relaciones sociales, plataformas de comunicación móvil cada vez más potentes e inteligentes; crecimiento exponencial del uso y servicios en “la nube”; contenidos digitales interactivos producidos por las personas (no sólo por empresas especializadas); Internet de las cosas y sus aplicaciones a prácticamente todas las actividades humanas, medios electrónicos, arte digital y administración de actividades comerciales (Garino, 2016).

Entre las tecnologías más utilizadas están los dispositivos móviles inteligentes, que se han convertido en una fuente rápida de comunicación entre las personas; estos con el tiempo muestran mejoras evidentes, incorporan más y mejores servicios a la vez que también mejoran la calidad de las comunicaciones. La cobertura de la infraestructura y los servicios crece, y al propio tiempo los precios siguen una clara tendencia de disminución. Los dispositivos móviles inteligentes también generan innovaciones porque promueven y facilitan la invención y producción de nuevos servicios, productos o procesos. Los ejemplos son comunes, desde la utilización de llamadas perdidas para actividades de la vida cotidiana hasta las operaciones bancarias móviles, tanto en las zonas rurales como en las zonas urbanas (Ardévol , 2011).

Los beneficios asociados a la diseminación de una tecnología para fines generales van más allá de su aplicación a procesos comerciales, y permiten generar mejoras en la calidad y variedad de los productos y los servicios que se ofrecen en el mercado. Como ocurre con los teléfonos fijos, la diseminación de la telefonía móvil entraña cambios en la organización diaria de la vida privada y los negocios. Ya se trate de empresas grandes o pequeñas, o de negocios de la economía estructurada o no estructurada, desde un punto de vista puramente económico se puede identificar varias esferas en las que la presencia de dispositivos móviles está promoviendo cambios. Tanto si funciona en combinación con la telefonía fija como si no lo hace, la

comunicación inalámbrica permite mayor flexibilidad de gestión y acelera los procesos que dependen de las comunicaciones (Ardévol , 2011).

Las evidencias disponibles demuestran que la utilización de dispositivos móviles inteligentes puede reducir los costos del acceso a la información y la incertidumbre en la adopción de decisiones. Esto es válido también en los casos en que no hay barreras técnicas o de precios para el acceso a la información. Cuando este último se facilita, los negociantes pueden tomar decisiones más informadas y, en consecuencia, puede mejorar la eficiencia del mercado. Pueden reducirse los gastos de transacción y debe aumentar la transparencia del mercado.

Existen diferentes tipos de comunicación que se pueden usar con estos dispositivos, ejemplo de ello son las aplicaciones *Whatsapp Messenger* que es sin duda la aplicación de mensajería instantánea más descargada del mundo, sentando un precedente en el camino de las comunicaciones a través de móvil; *Gmail* es también otra aplicación líder, que ya ha ganado el terreno a otros servicios de correo que estaban antes como *Hotmail* y *Yahoo*, se basa en la idea de hacer que el correo electrónico resulte más intuitivo, eficiente, útil e incluso divertido; *Line*, esta aplicación es la última revolución en cuanto a servicios de mensajería instantánea se refiere, entre las claves de su éxito, el hecho de que integra a la perfección el envío de mensajes y llamadas por datos de manera gratuita, el ser multidispositivo y poder usarse desde la PC o su genial gestión de las notificaciones; también están el SMS(*Short Message Service*),que es el servicio de la telefonía celular móvil que posibilita enviar y recibir mensajes de texto de extensión reducida; Facebook *Messenger*, es la aplicación de mensajería instantánea de Facebook, a través de la cual se puede mandar mensajes privados o tipos de chats a un contacto concreto, a todos los contactos o a determinados grupos (Javier S. , Las mejores aplicaciones con filtros artísticos para tu Android , 2017); las Llamadas, es el servicio de telefonía celular que permite la comunicación verbal entre dos personas, sin necesidad de estar cerca una de la otra (Pérez Porto Julián, 2016); *Skype*, permite realizar llamadas y videollamadas gratuitas; el estándar de mensajería MMS*(Multimedia Messaging Service)*; además de Viber, que permite crear grupos y mandar mensajes de forma instantánea y KakaoTalk, que es otra aplicación de mensajería móvil instantánea y llamadas a través de internet (Javier S. , Las mejores aplicaciones con filtros artísticos para tu Android , 2017).

En Cuba cada día es más creciente el uso de los teléfonos móviles inteligentes en la sociedad. Entre los diferentes tipos de comunicación antes mencionados, los más usados en el país son las llamadas, Facebook

Messenger y los SMS. En el año 2003 se funden en ETECSA las empresas celulares CCom y Cubacel, creando un operador unificado con una nueva concesión y un nuevo plan de desarrollo de telefonía fija, celular y datos. La obtención de créditos y tecnologías a partir del año 2007 permitió implementar un programa de desarrollo de la telefonía móvil, que posibilitó el comienzo de la comercialización de este servicio a personas naturales el 14 de abril del 2008. Desde el inicio de la expansión y hasta el cierre del 2014, la cantidad de celulares se ha incrementado en más de 2 millones. En este crecimiento ha incidido los cambios en la cuota de activación; aplicación de promociones con rebaja de tarifas internacionales para llamadas y envíos de SMS; promociones de recarga; ampliación del ciclo de vida de la línea, independientemente del monto que se le asigne; reducción de tarifas en zonas de bajo tráfico; la apertura del correo electrónico desde móviles que ya tiene 500 mil usuarios, y de otros servicios a través de la red celular (Salomón, 2017).

Además, ya existe una mayor posibilidad de que los ciudadanos cubanos puedan acceder a las redes sociales, gracias a las posibilidades que brinda ETECSA, pero, aún así, el medio de comunicación más usado es el SMS; es por esto que para los comentaristas deportivos cubanos, cuando están en vivo realizando una transmisión, lo usual es que utilicen los SMS como vía de comunicación para crear polémicas entre los aficionados del deporte, dar saludos, responder preguntas en los espacios televisivos y de esta manera crear un ambiente de discusión sobre el tema que estén tratando en ese programa. Por el gran cúmulo de mensajes que reciben constantemente, se les hace difícil leer todos los SMS recibidos durante una emisión (sobre todo en un gran evento), así como sacar información necesaria de los SMS que reciben y buscar algún mensaje que necesiten consultar en un momento determinado.

De la situación antes expuesta se origina el siguiente **problema de investigación** ¿Cómo facilitar el proceso de filtrado de los SMS que reciben los comentaristas deportivos cubanos durante las transmisiones en vivo?

Se define como **objeto de estudio**: proceso de filtrado de SMS, siendo el **campo de acción**: proceso de filtrado de SMS por varios criterios.

El **objetivo general** de la investigación es: desarrollar un sistema para facilitar el proceso de filtrado de SMS que reciben los comentaristas deportivos cubanos durante las transmisiones en vivo.

Del objetivo general se derivan los siguientes **objetivos específicos**:

- Construir los referentes teóricos relacionando los aspectos fundamentales que sustentan la investigación, mediante los cuales se consulta, extrae y recopila la información relevante sobre el problema a investigar.
- Analizar las diferentes tecnologías, sistemas de filtrados de SMS y aplicaciones existentes, valorando su posible adaptación.
- Implementar un sistema de filtrado de SMS que permita el envío de información desde los dispositivos móviles inteligentes a una aplicación de escritorio multiplataforma para filtrar los SMS por varios criterios.
- Realizar pruebas al sistema implementado, utilizando los métodos científicos seleccionados.

Como **idea a defender** se tiene que con el desarrollo de un sistema de filtrado de SMS que conste de una aplicación Android para dispositivos móviles inteligentes y una aplicación de escritorio multiplataforma se facilitará a los comentaristas deportivos cubanos la búsqueda de los SMS que reciben durante las transmisiones en vivo.

Para sustentar la investigación fueron utilizados los siguientes métodos de investigación:

Métodos teóricos

Los métodos teóricos utilizados en la investigación son:

- **Analítico sintético**: Permite un análisis mental del objeto o fenómeno en sus múltiples relaciones o componentes facilitando su estudio, así como mezclar partes anteriormente analizadas descubriendo características generales y relaciones entre ellas (Jennifer, 2013). Este método es utilizado en la investigación con el objetivo de analizar definiciones y conceptos importantes que tienen que ver con el filtrado de SMS por varios criterios, además permite resumir la información e identificar las características que aplican a la investigación.
- **Histórico lógico**: Analizan la trayectoria del fenómeno e investiga las leyes generales del funcionamiento y desarrollo de los fenómenos (Valeria, 2015). Se utiliza en la investigación para analizar la evolución de los SMS e investigar las leyes generales del funcionamiento y desarrollo de los tipos de filtrados de SMS para saber cómo debe funcionar un sistema de este tipo.

Métodos empíricos

- **Observación**: Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado (Jaime, 2014). Este método es usado para percibir directamente la realidad y actualidad que presentan las aplicaciones de filtrado de SMS en el mundo, se estudia su comportamiento y se recoge la mayor información necesaria que ofrecen.

Estructuración del contenido con una breve explicación de sus partes. El presente trabajo consta de una introducción, tres capítulos, conclusiones generales, recomendaciones, referencias bibliográficas utilizadas y, por último, los anexos que complementan el cuerpo del trabajo. A continuación, se explicará de forma breve en que consiste cada capítulo.

Capítulo 1: Se definen los conceptos de SMS, se describen sus principales características, cómo han evolucionado desde su surgimiento, se analizan diferentes aplicaciones de filtrado que existen y se realiza la selección de las tecnologías y herramientas actuales, que se van a utilizar para el desarrollo del sistema.

Capítulo 2: Se exponen las características de la propuesta de solución. Se realiza un levantamiento de requisitos, se obtienen las historias de usuarios. Además, se exponen los elementos del diseño de la solución propuesta.

Capítulo 3: En este capítulo se documenta lo referente a la implementación de los requisitos funcionales anteriormente definidos y posteriormente se le hace una serie de pruebas al sistema con el objetivo de verificar su correcto funcionamiento.

Capítulo 1 Fundamentación teórica

Introducción

El objetivo de este capítulo es identificar los conceptos y elementos teóricos que permitan comprender las características principales de los SMS y de los tipos de filtrado de SMS. También, se da una panorámica de la evolución de los SMS desde sus inicios hasta la actualidad y se estudian algunos de los tipos de filtrado que existen. Además, se realiza una breve explicación sobre las principales tecnologías y herramientas que se utilizarán en la solución propuesta.

1.1. SMS (*Short Message Service)*

El servicio de mensajes cortos o servicio de mensajes simples, más conocido como SMS (por las siglas del inglés *Short Message Service*), es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos, conocidos como mensajes de texto, entre teléfonos móviles (Pérez Porto Julián, 2016).

El SMS nació en Newbury, Berkshire (Reino Unido) el 3 de diciembre de 1992 cuando el programador Neil Papworth envió un mensaje desde su ordenador a un amigo, empleado de Vodafone, para felicitarlo por la Navidad. La operadora descubrió entonces un sistema útil y divertido para comunicarse dentro de su propia empresa (Pérez Porto Julián, 2016).

De todos modos, no fue hasta siete años más tarde cuando se formalizó el uso de este nuevo sistema para comunicarse de forma rápida y personal sin tener que realizar una llamada de voz. Ahora son más de 8 trillones de SMS los que recorren nuestra red móvil mundial al año, 15 millones por minuto (Aniversario del SMS: cómo se logró enviar el primer mensaje de texto de la historia hace 25 años, 2017).

Las operadoras, compitiendo entre ellas, comenzaron a usar el SMS como nuevo servicio para atraer clientes, ya que permitía múltiples formas de comunicación frente a la idea de usar los móviles sólo para llamar, que venían inculcando ellas mismas desde hace años.

El 92% de usuarios actuales de un *Smartphone*, con todas las opciones que hay actualmente, prefieren escribir un mensaje a llamar. Al principio fueron muchas las webs que ofrecían el envío de SMS gratuitos.

Los jóvenes situados entre 18 y 25 años fueron los usuarios más potentes del sistema, ya que enviaban una media de 133 mensajes por semana. También se sabe que el segundo uso más común de un terminal móvil es, precisamente, mandar SMS, el primero es mirar la hora (Pérez Porto Julián, 2016).

Después de analizar lo planteado anteriormente se puede concluir que el SMS es un mecanismo que surge con el objetivo de transmitir información textual a través de dispositivos móviles. Esto abre un nuevo canal de comunicación para usuarios de todo el mundo que puede ser aplicado para fines tanto personales como comerciales.

1.2. Spoofing de SMS

Desde el uso masivo de Internet en los *Smartphone*, la cantidad de SMS que se envían y reciben ha disminuido mientras los usuarios se vuelcan a plataformas *online*. No obstante, aún siguen siendo una herramienta usada de la cual poco se sabe en términos de la seguridad y autenticidad del origen de los mensajes.

Los ciberdelincuentes se han ido profesionalizando, acompañando los avances tecnológicos e innovando en distintas técnicas y canales. Las capacidades del *Spoofing* SMS (suplantación de SMS) son un claro ejemplo de cómo los ciberdelincuentes utilizan tecnologías emergentes para reutilizar los métodos de Ingeniería Social, una técnica vigente de suplantación de identidad en correos electrónicos.

El *Spoofing* SMS es el envío de mensajes de texto sin un origen veraz, lo que significa que el ciberdelincuente falsifica o anonimiza la dirección del remitente. Muchas veces este proceso es utilizado para el envío de campañas publicitarias, aunque en muchos países este tipo de publicidad es ilegal (Standby, 2017).

Como consecuencia, los proveedores de servicios telefónicos han comenzado a filtrar estos mensajes; sin embargo, estas técnicas aún siguen vigentes en el mundo de los ciberdelitos.

El *Spoofing* SMS funciona en todo el mundo en casi todas las redes móviles. Europa y Australia son dos de los lugares donde resulta más fácil falsificar los mensajes SMS. En cambio, Estados Unidos o Canadá son probablemente los países en donde, por protocolos de seguridad, es más difícil falsificar mensajes aplicando las técnicas usuales (Standby, 2017).

Hoy en día, la suplantación de SMS no es tan fácil como lo era hace unos años atrás. Pero es posible encontrar muchos servicios que por algunos centavos de dólar permiten este tipo de envíos. La efectividad de la recepción del mensaje variará mucho según las tecnologías aplicadas en el país y la compañía telefónica involucrada (Support, 2017).

Se debe tener cuidado especial al utilizar estos servicios, debido a que muchos realmente no funcionan, sino que en realidad son una trampa: solo capturarán un número de teléfono válido para luego enviar *spam*. De este modo, este tipo de servicios es utilizado en muchas ocasiones para generar bases de datos conteniendo números telefónicos válidos (Support, 2017).

También existen varias aplicaciones tanto en *Google Play* como en *Apple Store*, que prometen a sus usuarios este tipo de técnicas, ya sea cambiando el origen del SMS por uno anónimo o bien por el que el usuario desee. Particularmente, este caso es más crítico que el uso de servicios, debido a que si se decide instalar este tipo de aplicaciones (que muchas veces se encuentran fuera de repositorios oficiales) se corre mucho riesgo de infectarse con *malware* (Standby, 2017).

Por todo lo antes expuesto, la información en la presente investigación debe tener una forma de encriptación, y así se logra una mayor seguridad en el envío y recibimiento de la misma.

1.3. Plataformas para la gestión de servicios de mensajerías de texto

Muchas empresas a nivel internacional cuentan con plataformas para la gestión de servicios destinados a dispositivos móviles. Cada una de estas plataformas tiene sus propias características, tanto desde el punto de vista de sus funcionalidades como de sus arquitecturas.

A continuación, se describirán las principales funcionalidades ofrecidas por las plataformas de diversas empresas a nivel mundial. Entre estas se encuentran empresaSMS.com y Une Internet de España, la norteamericana Google, Vodafone y Esendex de Gran Bretaña y Mercacel y Nuestro Site de México.

EmpresaSMS.com

EmpresaSMS.com es una compañía española de nuevas tecnologías especializada en la creación, desarrollo de nuevas alternativas y tecnologías para el envío de mensajes de texto a teléfonos móviles, cuyo objetivo

es ofrecer una plataforma y soporte de comunicación por SMS simple, rápido, seguro, económico y efectivo al alcance de todo tipo de empresas y organizaciones públicas y privadas (SMS empresas. Servicios SMS para empresas, s.f.).

EmpresaSMS.com ofrece la posibilidad de enviar uno o más mensajes de texto a uno o más teléfonos móviles a través de una página que se encuentra en su sitio web y por un costo determinado por cada mensaje que se envíe (SMS empresas. Servicios SMS para empresas, s.f.).

El acceso a la página de envío requiere que el usuario posea un identificador de usuario y una contraseña. En lo adelante, deberá indicar el país y los destinatarios del mensaje, el texto del mensaje y otros datos de personalización como si se van a utilizar caracteres especiales, o si se va a programar el envío del mensaje a una hora determinada; y podrá enviar el mensaje a su destino (SMS empresas. Servicios SMS para empresas, n.d.).

La solución que da empresaSMS.com es sencilla y consiste en dar la posibilidad de que los usuarios subscritos al sitio puedan enviar mensajes de texto a móviles en varios países. Para poder enviar mensajes, cada usuario debe solicitar a través del sitio web de la empresa que se le cargue su crédito de SMS, en paquetes que pueden ir de 500 a 50 000 SMS, cada paquete por un precio determinado (SMS empresas. Servicios SMS para empresas, s.f.).

Vodafone

Vodafone es una de las mayores compañías de redes de telecomunicaciones móviles del mundo, con intereses en redes móviles y que cuenta con más de 147 millones de clientes. Vodafone ofrece un servicio llamado Dicta SMS, que en esencia se encarga de convertir los mensajes de voz en mensajes de texto. Este servicio funciona de dos formas diferentes. La primera es mediante activación. Todos los usuarios que tengan activado este servicio, si su teléfono está fuera de servicio o no contesta a la llamada, la persona que los llama les puede dejar un mensaje de voz que el usuario recibirá en forma de mensaje de texto. Con la segunda variante de este servicio, llamada SMS Fácil, no es necesario que la persona que llame o reciba la llamada tenga activado el servicio. Esta permite dejar un mensaje de voz que el interlocutor recibirá en forma de mensaje de texto, aunque el servicio no esté activado. En general, este servicio de mensajería permite

convertir mensajes de voz en mensajes de texto, sustituyendo de esta forma el contestador tradicional (Vodafone España, 2017).

Mercacel

Mercacel es una empresa mexicana cuyo objetivo es ofrecer el servicio de mensajería para empresas, basándose fundamentalmente en servicios de mensajería de texto. Mercacel cuenta con un servicio de mensajería que permite enviar mensajes de texto a cualquier móvil a través de una página web, además de una aplicación de escritorio para enviar mensajes. Esta aplicación permite enviar mensajes desde una computadora en cuestión de minutos y constituye una herramienta de gran utilidad para las empresas que necesitan comunicarse con empleados y clientes o con muchas personas. Esta aplicación permite además el envío de mensajes masivos, posee plantillas para mensajes estándar, informes de entrega en tiempo real, y la posibilidad de hacer encuestas a los clientes. Además, Mercacel ofrecen un SDK SMS, un paquete que permite la integración de estos servicios de mensajería en las aplicaciones empresariales de sus clientes (Mercacel. Servicios. , 2009).

La solución de Mercacel difiere de las anteriores por su enfoque empresarial. Su propósito es permitir que distintas empresas utilicen sus servicios de mensajería desde las propias aplicaciones de las empresas. Para ello Mercacel ofrece una aplicación de escritorio con un conjunto de funcionalidades estándar, y ofrece un SDK para que otras aplicaciones puedan utilizar sus servicios (Mercacel. Servicios. , 2009).

Une Internet

Une Internet es otra empresa española que propone un sistema de envío de mensajes de texto masivos. Los usuarios o empresas que deseen este servicio, deben pagar previamente uno de los paquetes ofertados, que pueden ser de 1000 a 50 000 mensajes de texto. Los envíos de mensajes se pueden realizar a través de una aplicación en línea, o bien mediante el programa que se conecta con la aplicación del cliente. La fecha y hora de los envíos de mensajes se pueden programar con antelación para asegurar una recepción óptima del mensaje, y la información sobre el estado de los envíos está permanentemente disponible en tiempo real gracias al sistema que se pone a disposición del cliente. El servicio de Une Internet es multi-operador, es decir, permite conectarse con distintos proveedores como son Movistar, Amena y Vodafone (Une Internet. Servicios SMS. , 2009).

La solución de Une Internet es similar a la anterior: da la posibilidad de envíos de mensajes de texto a través de una página web, y también a través de una aplicación de escritorio que se instala localmente. Como característica distintiva, esta aplicación permite conocer el estado de los mensajes enviados por los clientes. Además, permite conectarse con distintos proveedores de servicios (Une Internet. Servicios SMS. , 2009).

Esendex

Esendex es una empresa británica que ofrece servicios de mensajería de texto a empresas en Gran Bretaña y otros países europeos. En la actualidad tienen como clientes a más de 7 000 compañías, tanto pequeñas y medianas empresas como grandes compañías, a nivel mundial. Su gama de servicios para empresas comprende diferentes herramientas que permiten enviar y recibir mensajes de texto por vía web y desde la cuenta de correo electrónico, además de una API que permite la integración de sus servicios de mensajería con sitios web o aplicaciones de terceros. Permiten además el envío de mensajes de texto que serán recibidos como mensajes de voz. La solución de Esendex añade a las características vistas anteriormente, el envío de mensajes de texto desde la cuenta de correo electrónico (Esendex, 2017).

Nuestro Site

Nuestro Site es una empresa mexicana dedicada al desarrollo de software para terceras empresas. Actualmente atiende a clientes de México en el sector privado, financiero, educativo y de gobierno, y brinda soluciones de infraestructura tecnológica. Entre sus productos se encuentra una plataforma para el envío de SMS y MMS (Nuestro Site. Plataforma de envío de mensajes SMS/MMS, 2009).

Esta plataforma brinda una herramienta de gestión y entrega de SMS, la cual permite integrar de una manera sencilla a distintas empresas al mercado de los dispositivos móviles. La plataforma de envío de mensajes consiste de tres módulos principales: conector, enrutador y administrador de contenido (Nuestro Site. Plataforma de envío de mensajes SMS/MMS, 2009).

- El conector se encarga de establecer las comunicaciones con los centros de mensajes, mediante diferentes protocolos como SMPP 3.4, *Web Services*, HTTP y otros. Permite además gestionar la conexión con diferentes operadores de manera simultánea.

- El enrutador es el encargado de administrar las marcaciones y las aplicaciones, con el fin de poder gestionar mejor los servicios con una o más marcaciones. Una marcación puede administrar diversos tipos de servicios como contenido, servicios en demanda, encuestas y otros. El administrador de contenidos gestiona la entrega del mismo, el proceso de las estadísticas y reportes.
- La gestión de las estadísticas permite conocer qué medio o publicidad es la que mejor está funcionando, por lo que la plataforma maneja canales de venta con el fin de poder administrar las marcaciones y los canales de venta por donde están cursando tráfico.

La plataforma permite gestionar diferentes palabras clave, de tal forma que cada servicio tenga una o más de una palabra clave. Permite también la gestión de clientes y campañas. Para cada cliente se genera un prefijo, con el fin de distinguir su tráfico, así como generar reportes a través de los cuales se pueda consultar solo sus servicios (Nuestro Site. Plataforma de envío de mensajes SMS/MMS, 2009).

En resumen, esta plataforma permite la gestión de diferentes servicios basados en mensajes de texto, se puede conectar con varios operadores, se encarga de enrutar los pedidos de los usuarios a las aplicaciones. Además, permite el registro de clientes y ofrece un módulo de estadísticas y reportes. Introduce, de forma adicional, el manejo de contenido para móviles como pueden ser juegos, tonos, animaciones, fondos de pantalla y otros.

Teleios Systems

Teleios Systems es una empresa fundada en 1997 que desarrolla y comercializa soluciones informáticas enfocadas en la telefonía inalámbrica y las redes, para un amplio rango de clientes que incluye instituciones gubernamentales, corporaciones y operadores telefónicos. Uno de sus productos es una plataforma para la gestión de mensajes, llamada Teleios MessageCentral (Ronald, 2017).

Teleios MessageCentral es una plataforma de entrega de servicios de mensajería móvil que permite a los operadores de telefonía móvil múltiples servicios y aplicaciones de mensajería, al tiempo que mantiene control sobre los accesos de red y el tráfico (Ronald, 2017).

Entre las características de esta plataforma se encuentra el soporte para múltiples protocolos, herramientas de administración que brindan información sobre las cuentas de mensajería, los privilegios de acceso,

tamaños de los mensajes y tiempo de entrega; permite la administración de las aplicaciones a través de herramientas de configuración, gestión del contenido, reportes de tráfico, integración y rastreo de facturas, e inteligencia empresarial, entre otras (Ronald, 2017).

En específico, las herramientas de inteligencia empresarial permiten:

- Capturar la información del subscriptor para conocer el contenido y los servicios que más se utilizan.
- Generar estadísticas para identificar horas de mayor consumo y generar reportes por hora y diarios.
- Toma de decisiones basadas en las tendencias del mercado, realizando perfiles demográficos de los usuarios.
- Ayudar a los objetivos estratégicos, ya que los reportes de tráfico pueden resultar útiles para implementar actividades buscando aumentar el tráfico de mensajes.

Resumen de las principales funcionalidades

Tras estudiar las distintas soluciones que brindan las empresas dedicadas a los servicios de mensajería de texto a nivel mundial, se puede enumerar un grupo de funcionalidades que podrían formar parte de una solución al problema de desarrollar una aplicación para el filtrado de mensajes de texto. Estas son:

- Comunicación con uno o varios móviles a través de distintos protocolos.
- Gestión de mensajes desde uno o más teléfonos a través de una aplicación de escritorio.
- Soporte para distintos tipos de servicios basados en mensajes de texto, teniendo en cuenta la palabra clave del mensaje. Estos servicios pueden dar distintos tipos de información a los usuarios.
- Utilización de técnicas de Inteligencia Artificial para determinar tendencias y otros elementos de interés.

1.4. Soluciones similares

Como parte de la investigación se estudiaron diferentes soluciones similares que utilizan diferentes filtros de SMS. Los filtros para los mensajes permiten configurar y organizar automáticamente los mismos, pueden mover mensajes a carpetas, eliminar mensajes, reenviarlos a otras direcciones y mucho más (Support, 2017).

Estos se aplican a los nuevos mensajes que se tienen en la bandeja de entrada. Los filtros se ejecutan en el mismo orden en el que aparecen en la lista de filtros.

NQ Call Blocker es una aplicación muy cómoda y manejable. Las pocas configuraciones que requiere son realmente fáciles de comprender y realizar, siendo totalmente aptas para cualquier nivel de usuario. No presenta dificultad alguna para utilizarla (todoteach, 2016).

Además de poder bloquear, de manera selectiva o total, el bloqueo de llamadas o mensajes de texto en el *smartphone*, NQ Call Blocker puede crear una zona privada donde tener ocultos contactos y registros de llamadas o SMS de los mismos. Una opción que ofrece gratis para un tiempo de prueba y que en su versión por tiempo indefinido es de pago.

Filtros antispam bayesianos

Los filtros bayesianos se basan, principalmente, en la experiencia y van aprendiendo de aquellos mensajes que los propios usuarios marcan como spam. Por tanto, necesitan de la intervención de los usuarios para ser efectivos (Jiménez, 2016).

Además, los filtros antispam bayesianos pueden agrupar diferentes análisis de vocabulario que permiten generar una lista de palabras buenas y malas para tener en cuenta. Estas listas son las que se generan al detectar patrones partiendo de los emails marcados manualmente como spam (Jiménez, 2016).

Listas negras (Blacklists)

Las listas negras son tanto un bloqueador de llamadas como un filtro de SMS. Puede bloquear fácilmente llamadas y mensajes de números no deseados, privados (ocultos, anónimos) o desconocidos (Vlad, s.f.). Además, son listados donde se registran las direcciones IPs que generan spam de forma voluntaria o involuntaria. Así, las listas negras excluyen todo lo que llega de determinadas IPs o remitentes al agrupar a los servidores que se conoce que envían spam o que tienen alguna vulnerabilidad que permite realizar spam (Angel, 2017).

Simpler Contacts

Simpler Contacts hace honor a su nombre en que es una libreta de direcciones fácil de usar. Basta con importar los contactos para comenzar a hacer búsquedas, crear grupos o eliminar duplicados. La mejor herramienta es la gestión de grupos, que permite crear comunidades a las que enviar mensajes SMS o correos electrónicos. Además, los filtros de la aplicación son extremadamente útiles cuando se está tratando de localizar a cierto contacto del que no se recuerde el nombre, pero sí el cargo, o simplemente se quiere echar un vistazo rápido a los cumpleaños recientes. Está muy bien también que se pueda hacer copias de seguridad o exportar los contactos a una gran variedad de servicios de manera que si se decide dejar de usar la aplicación es muy sencillo llevar los contactos a otra parte (Thorin, 2015).

Mr. Number

Con Mr. Number se puede bloquear de modo sencillo tanto llamadas no deseadas como las que aparecen sin identificación. Además, también se puede bloquear SMS. Nada más entrar en la aplicación se accede a una lista en donde se muestran las últimas llamadas y mensajes de texto recibidos. Pulsando sobre uno de estos números se accede directamente a la pantalla de bloqueo donde solo hay que hacer clic en la señal de prohibido (Luis, 2018).

Filtros que analizan el contenido

Los filtros anti-spam analizan el contenido de los correos electrónicos para determinar si el contenido es sospechoso (si, por ejemplo, un correo se parece a un *phishing* o un spam previamente detectados).

También buscan la presencia de *spamwords* (o de *spamwords* escondidos con trucos ortográficos como por ejemplo ‘V14gr4’). De igual forma, analizan que los enlaces apunten a sitios respetables, así como que la relación entre texto e imágenes sea la adecuada (Paul, 2013).

Filtros que piden un segundo intento

Este método se basa en la premisa de que un *spammer* no hace un segundo intento de enviar un correo electrónico a una dirección si a la primera no lo consiguió entregar. En efecto, esto implicaría para el *spammer* una gestión muy costosa de sus rebotes.

El servidor de correo envía al servidor remitente un mensaje que informa de un error temporal en la recepción del correo. Un emisor normal, un tiempo más tarde, volverá naturalmente a intentarlo y demostrará así que, probablemente, no es un *spammer* (Javier S. , Las mejores aplicaciones con filtros artísticos para tu Android , 2017).

Filtros que piden una intervención manual

Una forma utilizada por los filtros para comprobar que el envío masivo no está realizado por un robot es enviar un mensaje solicitando una intervención manual que servirá como prueba de que el remitente es humano. Por ejemplo, pueden pedir que cesen los envíos durante 20 minutos, o que añadan una información complementaria. Estos mensajes no llegan por email sino en los '*logs*', mensajes directos enviados entre servidor remitente y servidor de recepción. Si el técnico consulta el mensaje y cumple con lo pedido, el mensaje se enviará y no se tendrá que realizar esta acción de nuevo en el futuro. Sino, se bloqueará como spam (Paul, 2013).

Análisis de las soluciones similares

Después de analizadas las soluciones, se decide que no pueden aplicarse para resolver el problema de investigación porque ninguna de ellas permite filtrar SMS por un criterio determinado; más bien se basan en bloqueo de llamadas y SMS, además otras permiten filtrar solo contactos o se basan en filtros solo para correos electrónicos. Pero a pesar de esto, se utilizan algunas características de los filtros que analizan el contenido, ya que la aplicación a desarrollar debe analizar el texto de los SMS para responder a temáticas determinadas. Se tendrán en cuenta, además, las características de los filtros de tipo *Simpler Contacts* dado que la aplicación también debe tener un tratamiento basado en los contactos de los cuales provienen los SMS.

1.5. Reglas de filtro

Estas son algunas reglas de filtros que existen (Santiago, 2018):

- El filtrado está desactivado. La aplicación no filtra las llamadas ni los SMS.
- Contactos bloqueados. La aplicación bloquea las llamadas y los SMS de los números de teléfono de la lista de contactos bloqueados (definidos de forma predeterminada). El número aparecerá ocupado

para los contactos bloqueados. La aplicación elimina los SMS bloqueados y la información queda registrada en un informe.

- Contactos permitidos. La aplicación permite únicamente las llamadas y los SMS de la lista de contactos permitidos. Las demás llamadas y SMS se bloquean. La aplicación elimina los SMS bloqueados y su información queda registrada en un informe.
- Filtrado estándar. La aplicación bloquea y permite llamadas y SMS de acuerdo con las listas de contactos bloqueados y permitidos. Si se recibe una llamada o un mensaje de texto de un número que no esté en ninguna de las listas, la aplicación lo entrega y solicita al usuario que agregue el número de teléfono a la lista de contactos permitidos o a la de bloqueados.

De este conjunto de reglas se decide utilizar algunas características del filtrado estándar para la investigación, puesto que en la aplicación se pueden recibir SMS desde cualquier número de teléfono, ya sea de algún contacto registrado o de un nuevo usuario, lo que no es necesidad del cliente que la aplicación de escritorio tenga una lista de contactos bloqueados.

1.6. Herramientas y tecnologías a utilizar

A continuación, se explican las herramientas y tecnologías que se utilizan para el desarrollo de la investigación.

1.6.1. Metodología de desarrollo de software

Una metodología es un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación que permiten guiar a los desarrolladores de aplicaciones.

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería (Metodología de Desarrollo de Software, 2017).

En el desarrollo de la aplicación se utiliza la metodología de desarrollo de software Variación de AUP para la UCI, una variante realizada por la Universidad de las Ciencias Informáticas a la metodología ágil AUP (Proceso Ágil Unificado) y está definida por la universidad como el documento rector de la actividad productiva.

Fases de Variación de AUP para la UCI:

La metodología Variación de AUP para la UCI está formada por tres fases, (Inicio, Ejecución y Cierre) para el ciclo de vida de los proyectos de la universidad, las cuales contienen las características de las cuatro fases (Inicio, Elaboración, Construcción y Transición) propuestas en AUP. Las características de las fases de la metodología de la universidad son (Tamara, 2015):

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el software es transferido al ambiente de los usuarios finales o entregado al cliente junto con la documentación. Además, en esta transición se capacita a los usuarios finales sobre la utilización de la aplicación.
- **Cierre:** En el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Descripción de las disciplinas

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMIDEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto). Para una mayor comprensión se muestra la siguiente tabla (Tamara, 2015):

Tabla 1. Descripción de AUP-UCI

Disciplinas AUP	Disciplinas VariaciónAUP-UCI	Objetivos Disciplinas(Variación AUP-UCI)
Modelo	Modelado de negocio	El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes: 1. Casos de Uso del Negocio (CUN) 2. Descripción de Proceso de Negocio (DPN) 3. Modelo Conceptual (MC) A partir de las variantes anteriores se condicionan cuatro escenarios para modelar el sistema en la disciplina Requisitos.
	Requisitos	El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos

		[Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP)], agrupados en cuatro escenarios condicionados por el Modelado de negocio
	Análisis y diseño	En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis
Implementación	Implementación	En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
Prueba	Pruebas internas	En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como

		las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas.
	Pruebas de liberación	Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
	Pruebas de aceptación	Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.
Gestión de configuración	Se cubren con las áreas de procesos PP, PMC y CM que propone CMMI-DEV v1.3. Las mismas son áreas de procesos de gestión y soporte respectivamente.	Consultar en mejoras.prod.uci.cu los libros de procesos de cada una de estas áreas.
Gestión de proyecto		
Entorno		

Escenarios para la disciplina Requisitos

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (CUN, DPN o MC) y existen tres formas de encapsular los requisitos (CUS, HU, DRP), surgen cuatro escenarios para modelar

el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma (Tamara, 2015):

Escenario No 1: Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

Escenario No 2: Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

Escenario No 3: Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con HU.

Para la investigación se escogió el escenario 4, el mismo se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información. Todas las disciplinas antes definidas (desde Modelado de negocio hasta Pruebas de Aceptación) se desarrollan en la Fase de Ejecución, de ahí que en la misma se realicen iteraciones y se obtengan resultados incrementales. En una iteración se repite el flujo de trabajo de las disciplinas, Requisitos, Análisis y diseño, Implementación y Pruebas internas. De esta forma se brinda un resultado más completo para un producto final de manera creciente. Para llegar a lograr esto, cada requisito debe tener un completo desarrollo en una única iteración (Tamara, 2015).

El desarrollo del sistema se basa fundamentalmente en la implementación, dado que los requisitos son concretos y no siguen un flujo de acciones sucesivas. Por ello no es necesario la utilización de un modelado de procesos, mapas conceptuales o casos de uso, haciendo que sea más comprensible la descripción mediante historias de usuarios. Por eso se decide aplicar el escenario 4 que está orientado a este tipo de artefacto.

1.6.2. Lenguaje de programación

Para el desarrollo de la aplicación se decide utilizar el lenguaje Java por ser multiplataforma y ser en la actualidad uno de los lenguajes más utilizados a nivel mundial (Fernando, 2017), además Java tiene librerías que son útiles para minimizar tiempo y esfuerzo en el desarrollo del sistema, se tiene mayor conocimiento de java y se posee una mayor experiencia en el desarrollo de software en este lenguaje.

A continuación, se especifican las características más importantes de este lenguaje de programación.

Java 8.0

Java surgió en 1991 cuando un grupo de ingenieros de Sun Microsystems trató de diseñar un nuevo lenguaje de programación destinado a correr sobre cualquier plataforma: desde computadoras hasta efectos electrodomésticos. Con este fin fueron de los primeros en introducir el concepto de máquina virtual con el objetivo de que sus aplicaciones fueran totalmente independientes del CPU que las procesara y esto constituye una ventaja, ya que, corriendo sobre la máquina virtual, te permite olvidarte de algo tan engorroso como es la gestión de memoria (punteros, reserva, etc.) (Fernando, 2017).

Sun describe a Java como simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico. Se ha desarrollado unido a Internet y es por ello que es ampliamente utilizado en la construcción tanto de sistemas distribuidos, como de complejas aplicaciones de gestión. Otra de sus ventajas es que cuenta con una librería de clases bastante completa que se puede aprovechar para el desarrollo de aplicaciones para distintas tecnologías (de escritorio, móvil, web) (Fernando, 2017).

Entre las limitaciones de Java se puede mencionar su velocidad, los programas hechos en Java no tienden a ser muy rápidos. Como los programas de Java son interpretados nunca alcanzan la velocidad de un verdadero ejecutable. Otra desventaja que posee Java es que algunas implementaciones y librerías pueden tener código rebuscado (Fernando, 2017).

1.6.3. Plataformas de Desarrollo que se utilizan en la investigación

Java FX 8.0

Java FX es una plataforma de desarrollo pensada para crear y desplegar aplicaciones para internet que funcionan en una gran cantidad de dispositivos (escritorio, navegadores y móviles). Esta plataforma provee al programador grandes ventajas debido a que posee un motor de renderización web y un motor de multimedia de alto rendimiento. Además, provee más de 50 componentes de diseño, formularios fácilmente personalizables mediante el uso de CSS. Lo que la diferencia de java es que esta plataforma les brinda estilo, diseño y personalidad a las aplicaciones (Rajeev, 2017).

Se utiliza JavaFX, ya que proporciona un amplio conjunto de controles UI, que simplifican el desarrollo de interfaces visualmente inmersivas para aplicaciones basadas en bases de datos. Cuando se combina con la base de datos PostgreSQL y la herramienta ORM de Hibernate, JavaFX puede manejar la capa de presentación para aplicaciones empresariales a gran escala basadas en datos (JavaFX PostgreSQL), así como aplicaciones de escritorio robustas **(JavaFX Hibernate).**

JavaFX es una familia de productos y tecnologías de Oracle Corporation (inicialmente Sun Microsystems), para la creación de *Rich Internet Applications* (RIAs), esto es, aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo aplicaciones multimedia interactivas. Las tecnologías incluidas bajo la denominación JavaFX son JavaFX Script y JavaFX Mobile, aunque hay más productos JavaFX planeados (Rajeev, 2017).

Android SDK 24.4.1

Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tabletas y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró. Android fue presentado en 2007 junto la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008. Android es el sistema operativo móvil más utilizado del mundo, con una cuota de mercado superior al 80% al año 2017, muy por encima de IOS.6789 (Juan, 2016).

El éxito del sistema operativo se ha convertido en objeto de litigios sobre patentes en el marco de las llamadas guerras de patentes entre las empresas de teléfonos inteligentes. Según los documentos secretos filtrados

en 2013 y 2014, el sistema operativo es uno de los objetivos de las agencias de inteligencia internacionales (Juan, 2016).

La versión básica de Android es conocida como Android Open Source Project (AOSP). El 25 de junio de 2014 en la Conferencia de Desarrolladores Google I/O, Google mostró una evolución de la marca Android, con el fin de unificar tanto el hardware como el software y ampliar mercados (Juan, 2016).

El 17 de mayo de 2017, se presentó Android Go. Una versión más ligera del sistema operativo para ayudar a que la mitad del mundo sin *smartphone* consiga uno en menos de cinco años. Incluye versiones especiales de sus aplicaciones donde el consumo de datos se reduce al máximo (Juan, 2016).

1.6.4. Entornos de Desarrollo Integrado (IDE)

Para el desarrollo del sistema se hace necesario la utilización de dos IDE que permitan una interacción con el lenguaje de programación. A continuación, se describen los IDE IntelliJ IDEA y Android Studio que se utilizan para el desarrollo en Java y Android simultáneamente.

IntelliJ IDEA 2017.2.2

Es un IDE para el desarrollo de programas informáticos. Es desarrollado por JetBrains (anteriormente conocido como IntelliJ), y está disponible en dos ediciones: community edition, y edición comercial. IntelliJ IDEA no está basada en Eclipse como MyEclipse o Oracle Enterprise Pack para Eclipse (Getting Started with IntelliJ IDEA, 2017).

Permite escritura de código sin complicaciones. Practica un abordaje no intrusivo e intuitivo para ayudar a escribir, depurar, refactorizar, probar y aprender el código. Gracias a su profunda comprensión de los lenguajes y tecnologías, IntelliJ IDEA proporciona un segundo par de manos para cuando se necesita (Getting Started with IntelliJ IDEA, 2017).

Además, crea un entorno adecuado en el que todos los miembros del equipo pueden trabajar juntos de manera eficiente. Integración transparente con una amplia variedad de sistemas de control de versiones, permite a los miembros del equipo permanecer en sincronía con los cambios de otros, asegurando que todas las contribuciones serán productivas. IntelliJ IDEA puede coexistir con otros IDEs populares, como Eclipse y

herramientas de gestión de proyectos como Maven, para que un equipo pueda utilizar cada herramienta donde es mejor aplicable (Getting Started with IntelliJ IDEA, 2017).

Permite a una empresa disponer de productos de software complejos, en cumplimiento de las normas de calidad y cronogramas. El IDE constantemente valida la calidad del código y ofrece soluciones inmediatas para los problemas encontrados en todos los niveles, desde la instrucción individual para arquitectura global, utilizando las inspecciones de código avanzado y análisis de matriz de dependencia. Sea un problema de codificación, problemas potenciales de rendimiento, o el incumplimiento de contrato, IntelliJ IDEA muestra una advertencia y corrige el problema, ayudando a producir de forma limpia, un código de primera línea en poco tiempo (Getting Started with IntelliJ IDEA, 2017).

Android Studio 3.0

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014. Está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de Android (AS, 2016).

Principales características de Android Studio (AS, 2016)**:**

- Soporte para programar aplicaciones para Android Wear.
- Herramientas Lint. Detecta el código no compatible entre arquitecturas diferentes o código confuso.
- Utiliza ProGuard, para poder optimizar y reducir el código del proyecto al exportar a APK, para dispositivos de gama con limitaciones.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de acabar la aplicación.
- Vista previa, en diferentes tipos de proyectos y resoluciones.
- Posibilita la opción del control de versiones accediendo a un repositorio y poder descargar Mercurial, Git, Github o Subversion.
- Permite la importación de los proyectos realizados desde Eclipse.

Ventajas y desventajas de Android Studio (AS, 2016) :

- Compilación rápida.
- Ejecución de la app en tiempo real gracias al emulador.
- Ejecución de la app directamente desde el móvil.
- No soporta el desarrollo para NDK, pero IntelliJ con el plugin Android sí.
- Funciona bien (sobre todo si se usan versiones estables).
- Contiene todo lo necesario para desarrollar cualquier IDE.
- Es capaz de asociar automáticamente carpetas y archivos con su papel en la aplicación, la creación de nuevas carpetas, borrado de archivos en values.
- Los requisitos son un poco elevados (se tendrá que tener una buena máquina para que funcione bien el emulador). Pero esto hace que sea el mejor entorno para programar en Android, por lo que es necesario.

1.6.5. Motor de base de datos

El motor de base de datos es el servicio principal para almacenar, procesar y proteger los datos. El mismo proporciona acceso controlado y procesamiento de transacciones rápido para cumplir con los requisitos de las aplicaciones consumidoras de datos más exigentes de una empresa (Gustavo R. , 2013).

A continuación, se realiza un estudio de diferentes motores de base de datos:

MongoDB

Es una base de datos *Open Source* de gran rendimiento, escalable, no es una base de datos relacional convencional, y orientada a documentos (esquemas de datos tipo JSON). Hay drivers preparados para usar esta base de datos desde lenguajes como PHP, Python, Perl, Ruby, JavaScript, C++ y muchos más (Picajoso, 2010).

Hypertable

Hypertable es un sistema de almacenamiento distribuido de datos de alto rendimiento diseñado para soportar aplicaciones que requieran máximo rendimiento, escalabilidad y eficiencia. Se ha diseñado y modelado a

partir del proyecto *BigTable* de *Google* y se enfoca sobre todo a conjuntos de datos de gran escala (Picajoso, 2010).

Apache CouchDB

Como en el caso de MongoDB, este proyecto está destinado a ofrecer una base de datos orientada a documentos que se pueden consultar o indexar en modo *MapReduce* usando JavaScript. CouchDB ofrece una API JSON RESTful a la que se puede acceder desde cualquier entorno que soporte peticiones HTTP (Picajoso, 2010).

HSQL 2.4.0 es un motor de bases de datos SQL ligero (611 Kb aprox.), OpenSource e implementado completamente en Java. Ideal para realizar pruebas sin tener que conectarse a un gestor de bases de datos. Es una herramienta increíblemente útil, ahorra bastante tiempo de desarrollo y al trabajar sobre bases de datos es flexible a la hora de realizar cambios en la información a tratar o cuando los requisitos de la información a tratar cambien. HSQL no debe tomarse como una solución, sería en un entorno multiusuario en donde los requisitos en tiempos de espera y seguridad son importantes. No hay que olvidar que trabaja sobre ficheros de texto (Carlos, 2006).

Riak

Riak es una base de datos ideal para aplicaciones web y combina (Picajoso, 2010):

- Una tienda con un valor clave descentralizado
- Un motor map/reduce flexible
- Una interfaz de consultas HTTP/JSPN amigable.

HBase

HBase es un almacén distribuido de tipo *column-oriented* que puede ser también denominado como la base de datos *Hadoop*. El proyecto está dirigido a ofrecer tablas enormes de “miles de millones de filas, y millones de columnas”. Dispone de un *gateway RESTful* que soporta XML, Protobug y opciones de codificación binaria de datos (Picajoso, 2010).

Keyspace

Se trata de un almacén del tipo *key-value* con replicación consistente y que funciona sobre sistemas operativos Windows. Keyspace ofrece una alta disponibilidad al enmascarar los fallos de servidor y red y al aparecer como un único servicio de alta disponibilidad.

Después de hacer un estudio de varios motores de base datos que existen y que más se utilizan en la actualidad, se decide utilizar para la investigación HSQL, pues puede realizar pruebas sin tener que conectarse a un gestor de base de datos, se ahorra tiempo de desarrollo, y se tiene una mayor experiencia con dicho motor.

Para la investigación se utiliza **Hibernate 5.2.2** como herramienta de mapeo objeto-relacional. A continuación, se explican algunas de sus características principales:

Hibernate 5.2.2 es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones (King Gavin, 2010).

Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la POO. Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por SQL, también genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución (King Gavin, 2010).

Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible; ofrece también un lenguaje de consulta de datos llamado HQL (Hibernate Query

Language), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria"). Hibernate para Java puede ser utilizado en aplicaciones Java independientes o en aplicaciones Java EE, mediante el componente Hibernate Annotations que implementa el estándar JPA, que es parte de esta plataforma (King Gavin, 2010).

1.6.6. Visual Paradigm 8.0 como herramienta de modelado

Visual Paradigm es una de las herramientas UML CASE del mercado, considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones (Visual Paradigm, 2014).

Fue creada para el ciclo vital completo del desarrollo de software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación. Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de las clases (Visual Paradigm, 2014).

Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros (Visual Paradigm, 2014).

Características

A continuación, se listan algunas de las características de Visual Paradigm (Visual Paradigm, 2014):

- Producto de calidad.
- Soporta aplicaciones Web.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- Se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, Jbuilder, NetBeans IDE, Oracle Jdeveloper, BEA Weblogic.

Ventajas (Visual Paradigm, 2014):

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes. Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Genera modelos VP-UML instantáneamente a partir de código binario .Net.
- Generación de documentación en formatos HTML y PDF.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.
- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Desventajas (Visual Paradigm, 2014):

- Las imágenes y reportes generados, no son de muy buena calidad.

Por las ventajas descritas anteriormente se selecciona Visual Paradigm para el modelado correspondiente al análisis de la investigación

1.6.7. Lenguaje de modelado UML 2.0 utilizado en la investigación

El lenguaje unificado de modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el Object Management Group (OMG). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados (César, 2016).

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo (César, 2016).

UML no puede compararse con la programación estructurada, pues UML significa Lenguaje Unificado de Modelado, no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que programación estructurada es una forma de programar como lo es la orientación a objetos, la programación orientada a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML solo para lenguajes orientados a objetos (César, 2016).

1.7. Conclusiones del capítulo

En este capítulo con el análisis de las plataformas que utilizan SMS, se logró definir características importantes para el desarrollo de la aplicación. Además, el estudio de los tipos de filtrado de SMS permitió establecer las reglas de filtrado que se van a utilizar. Para guiar el desarrollo de software se utiliza la metodología AUP-UCI, partiendo de que es la metodología más utilizada en la UCI. Se decide utilizar java

como lenguaje de desarrollo de software, por ser un lenguaje multiplataforma y el desarrollador cuenta con una mayor experiencia en este lenguaje.

Capítulo 2 Propuesta de solución. Análisis y diseño

Introducción

Las características del sistema a desarrollar y su descripción detallada son los elementos que se evidenciarán en este capítulo. Se utiliza el modelo de dominio para describir los principales conceptos asociados al dominio del problema y sus relaciones. Se diseña y describe la propuesta de solución al problema planteado. Se realiza la captura de los requisitos funcionales y no funcionales del producto.

2.1. Propuesta de solución

A los comentaristas deportivos cubanos se les hace difícil buscar mensajes de texto enviados por los aficionados para responder sus preguntas, mandar saludos, entre otras acciones que se realizan en los programas televisivos de este tipo, pues no tienen a su disposición una aplicación capaz de almacenar y filtrar los SMS enviados por teléfonos móviles de la población cubana. La ausencia de esta aplicación dificulta la comunicación fluida entre los aficionados y los comentaristas en una transmisión en vivo, provocando que no exista un debate sobre los comentarios enviados por los aficionados dando su opinión respecto al tema que se está tratando en ese momento.

Es por ello que como propuesta de solución se plantea el desarrollo de una aplicación Android para dispositivos móviles y una aplicación de escritorio multiplataforma que permita filtrar estos mensajes de textos y de esta manera facilitar la búsqueda de los mismos para los comentaristas deportivos. La aplicación de escritorio desarrollada debe reconocer los dispositivos móviles conectados a su misma red Wifi, que tengan instalada la aplicación Android implementada en la presente investigación. Después de conectados los móviles a la aplicación de escritorio, esta hace una captura de los SMS almacenados en el dispositivo móvil inteligente y los que se reciban en tiempo real. En función de dicha captura debe permitir al comentarista hacer búsquedas dentro de los SMS a través de palabras o filtros previamente creados. También debe permitir gestionar los filtros, gestionar SMS y gestionar contactos.

2.2. Modelo de dominio

El modelo del dominio captura, comprende y describe los objetos más importantes dentro del contexto de un sistema. Se describe mediante diagramas de UML, especialmente mediante diagramas de clases. Los objetos del dominio o clases pueden obtenerse a partir de una especificación de requisitos o mediante la entrevista con los expertos del tema. Las clases que lo componen suelen aparecer en tres formas típicas (Jacobson, 2000):

- Objetos del negocio que representan los elementos que se manipulan en el negocio
- Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento
- Sucesos que ocurrirán o que han ocurrido

El modelo de dominio que a continuación se describe constituye una representación gráfica de los principales conceptos que forman parte de la aplicación de filtrado de SMS para los comentaristas deportivos cubanos.

2.2.1. Descripción de los conceptos del modelo de dominio

A continuación, se esclarecen los conceptos asociados al dominio, definiendo los términos más importantes que serán de utilidad para la posterior confección del modelo de dominio.

Contacto: Persona registrada en el dispositivo móvil del comentarista deportivo, que envía SMS a dicho dispositivo.

Aficionado: Persona aficionada del deporte, que envía SMS al dispositivo móvil del comentarista deportivo.

SMS: Mensajes de textos enviados de dispositivos móviles, ya sea de los contactos o de los aficionados.

APK del dispositivo móvil: Es una aplicación en el dispositivo móvil inteligente del comentarista deportivo, donde recibe todos los SMS, que permite la conexión a la aplicación de escritorio.

Aplicación de escritorio: Es una aplicación en la PC del comentarista deportivo, la cual va a facilitar el filtrado de SMS en tiempo real.

Filtro: Va a ser ejecutado por el comentarista deportivo, según el SMS que quiera buscar, o la persona que envió el SMS, ya sea contacto o aficionado.

Usuario: Es la persona (comentarista deportivo) que va a utilizar la aplicación de escritorio y el dispositivo móvil con el que se reciben todos los SMS.

2.2.2. Diagrama del modelo de dominio

El siguiente diagrama muestra la relación entre los conceptos identificados del problema descrito anteriormente.

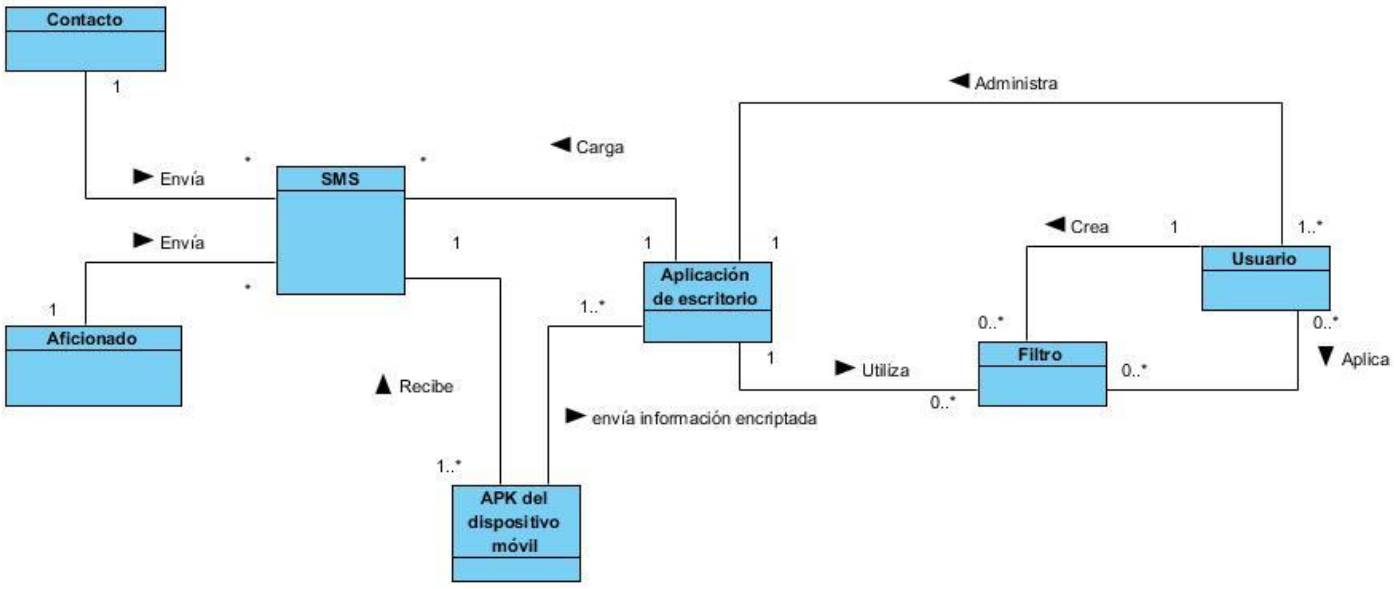


Ilustración 1. Diagrama de Modelo de Dominio

2.3. Requisitos funcionales y no funcionales

El proceso de desarrollo de software comprende en sus etapas tempranas la definición de tareas orientadas a captar las necesidades o características para satisfacer el sistema que se vaya a crear o modificar (Pressman, 2010). Durante la fase de análisis fueron identificados los requisitos funcionales y no funcionales necesarios para que la aplicación pueda satisfacer el problema planteado.

2.3.1. Requisitos funcionales

Los requisitos funcionales identificados para dar solución al problema planteado fueron:

RF1: Permitir conexión entre dispositivo móvil y servidor a través de red Wifi.

RF2: Permitir que el servidor soporte varias conexiones simultáneas.

RF3: Enviar información del dispositivo conectado.

RF4: Enviar contactos del dispositivo conectado.

RF5: Enviar mensajes del dispositivo conectado.

RF6: Notificar al servidor de la llegada de mensajes nuevos, además de enviarlos.

RF7: Encriptar información enviada a través de la conexión Wifi.

RF8: Permitir búsqueda de servidor disponible en la subred del dispositivo móvil.

RF9: Permitir al usuario especificar directamente el ip del servidor.

RF10: Listar dispositivos almacenados y conectados.

RF11: Solicitar autenticación al conectarse un dispositivo.

RF12: Solicitar autenticación al querer mostrar los mensajes de un dispositivo almacenado.

RF13: Mostrar mensajes de los dispositivos autenticados.

RF14: Ocultar mensajes propios al desconectar un dispositivo.

RF15: Mostrar contactos de los dispositivos conectados.

RF16: Mostrar mensajes de un contacto seleccionado.

RF17: Buscar mensajes que coincidan con un criterio de búsqueda.

RF18: Permitir tener activas varias búsquedas a la vez.

RF19: Eliminar mensaje de la base de datos.

RF20: Eliminar los mensajes que coincidan con el criterio de búsqueda.

RF21: Actualizar interfaz al recibir nuevos mensajes.

RF22: Mostrar los 5 contactos más activos.

RF23: Seleccionar varios temas.

A continuación, se muestra una tabla donde se separan los requisitos funcionales según el tipo de aplicación:

Tabla 2. Requisitos funcionales según el tipo de aplicación

Requisitos funcionales de la aplicación Android	Requisitos funcionales de la aplicación de escritorio
RF3,RF4,RF5,RF6,RF7,RF8,RF9	RF1,RF2,RF10,RF11,RF12,RF13,RF14,RF15,RF16,RF17,RF18,RF19,RF20,RF21,RF22,RF23

Las descripciones de estos requisitos funcionales se pueden visualizar en el anexo #1.

2.3.2. Requisitos no funcionales

Los requerimientos no funcionales, como su nombre lo dicen, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema (Sommerville, Ingenería de software, 2007) .

Los requerimientos no funcionales son requisitos que imponen restricciones en el diseño o implementación. Son propiedades o cualidades que el producto debe tener (Pressman, 2010). Seguidamente se listan los requisitos no funcionales para la solución propuesta:

Portabilidad

RNF.01- El sistema debe funcionar desde una aplicación de escritorio que permita sincronizar los SMS de dispositivos móviles inteligentes con una aplicación Android, a partir de conexión wifi, y de esta manera poder gestionar los SMS.

RNF.02- Utilizar el idioma español para los mensajes y textos de la interfaz.

Software

RNF.03- Se podrá utilizar la aplicación en los sistemas operativos Linux y Windows.

RNF.04- El dispositivo móvil debe tener sistema operativo Android, en su versión 4.4 o superior y una tarjeta SIM para el envío y recepción de mensajes.

Soporte

RNF.05- Las herramientas seleccionadas para el desarrollo del producto están respaldadas por licencias libres, bajo las condiciones de software libre.

Capacidad

RNF.06- Soporte de múltiples conexiones desde diferentes móviles.

Confiabilidad

RNF.07- La precisión y exactitud de las salidas del sistema se corresponden con la calidad y exactitud de la información contenida en la base de datos y en el dispositivo móvil, y los filtros introducidos por los usuarios a la aplicación.

Seguridad

RNF. 08- La información que se envían desde el dispositivo móvil debe viajar encriptada hacia el servidor.

RNF. 09- El sistema debe permitir almacenar información referente a la autenticación de cada dispositivo, para de esta manera solo visualizar los SMS de los dispositivos que se autentiquen.

2.3.3. Historias de usuario

Las HU son la forma en que se especifican en las metodologías ágiles los requisitos del sistema, las mismas no deben ser descritas en más de tres líneas e idealmente es el cliente quien las redacta y prioriza. Por tanto, serán descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica (Bruno, 2017).

Las HU deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios. Las HU deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia (Bruno, 2017) .

En el cuarto escenario de AUP, en su variante UCI, los requisitos se administran utilizando historias de usuario, por lo que para la solución propuesta se describieron un total de 23 Historias de Usuario (Ver Anexo #1), de las cuales se presenta la correspondiente al RF17:

Tabla 3. Historia de Usuario para el RF17

Número: RF17	**Nombre del requisito:** Buscar mensajes que coincidan con un criterio de búsqueda.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir buscar los SMS del dispositivo móvil a partir de un criterio determinado. **2- Acciones para lograr el objetivo (precondiciones y datos):**	

- Para buscar los SMS, debe haber algún dispositivo móvil conectado, y además el usuario debe haber creado algún filtro previamente.

3- Comportamientos válidos y no válidos (flujo central y alternos):

El campo es obligatorio.

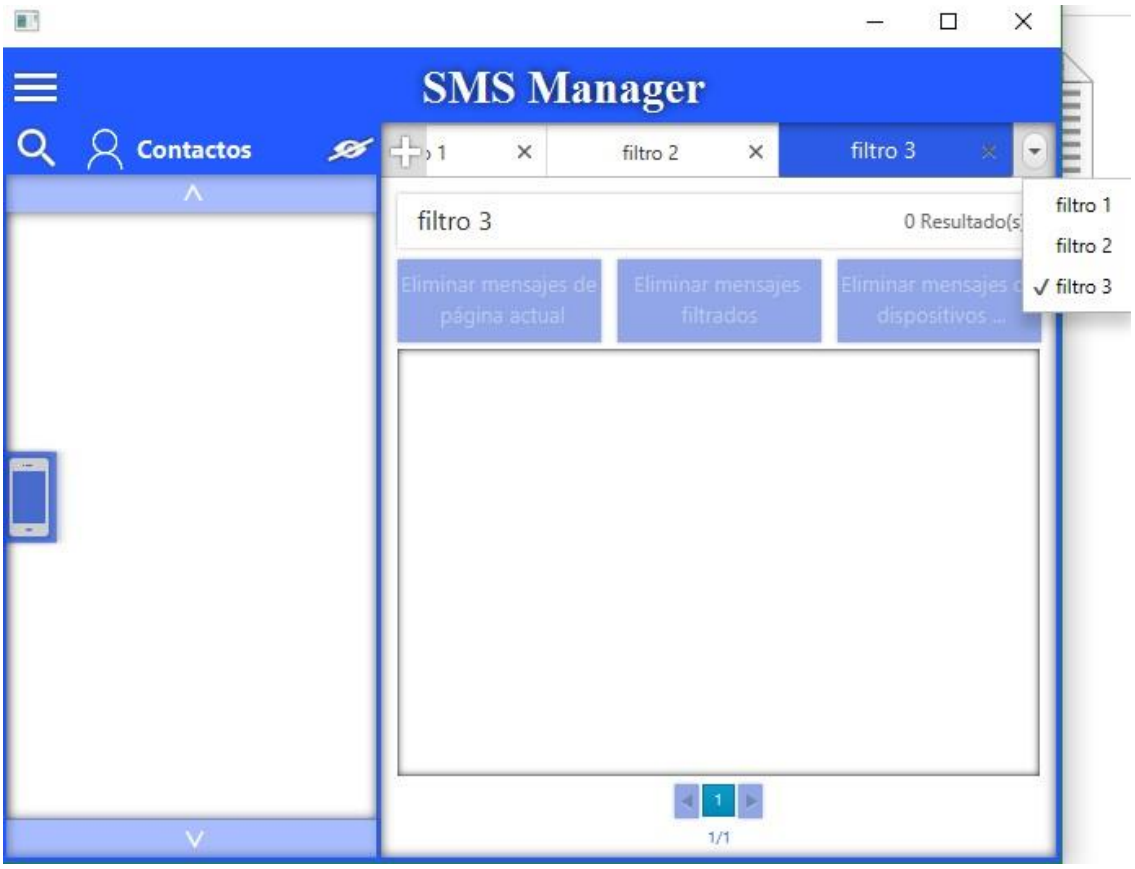
- (*) Filtro: permite todo tipo de caracteres.

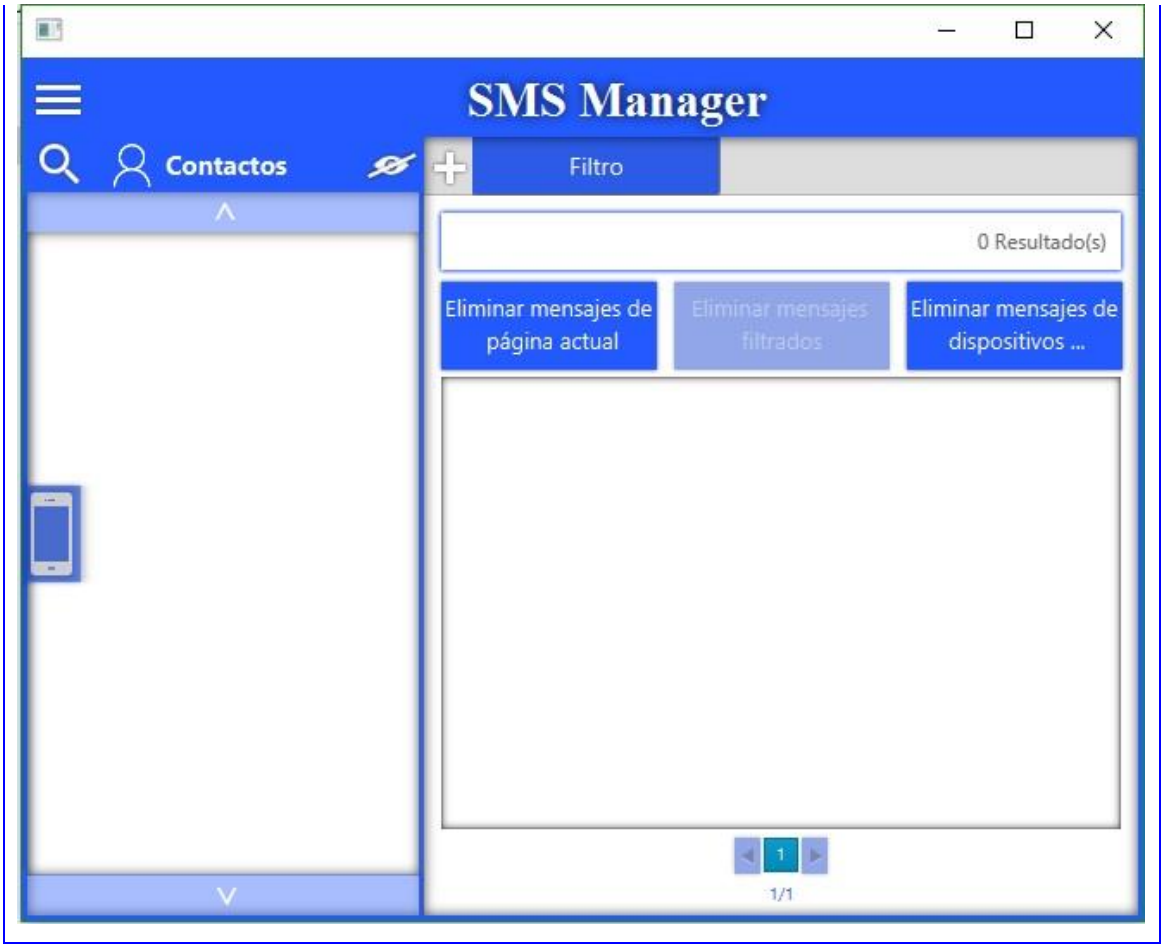
4- Flujo de la acción a realizar:

Una vez que el usuario comience a teclear el filtro, el sistema muestra los SMS que contengan el filtro tecleado hasta el momento.

Observaciones:

Prototipo de interfaz:





2.4. Modelo de Diseño

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en los requisitos funcionales y no funcionales, junto a otras restricciones relacionadas con el entorno de implementación (Sommerville, Ingenería de software(8ed), 2007).

La disciplina de modelado de la metodología AUP en su variante UCI comprende las fases de modelado del negocio, requisitos, análisis y diseño. Este modelado es colaborativo, generando los artefactos necesarios para la comprensión del sistema con un mayor nivel de abstracción y una vez obtenidos dichos elementos se da paso a la implementación y las pruebas.

A continuación, se presenta el diagrama de clases del diseño del Sistema de filtrado de SMS para los comentaristas deportivos cubanos:

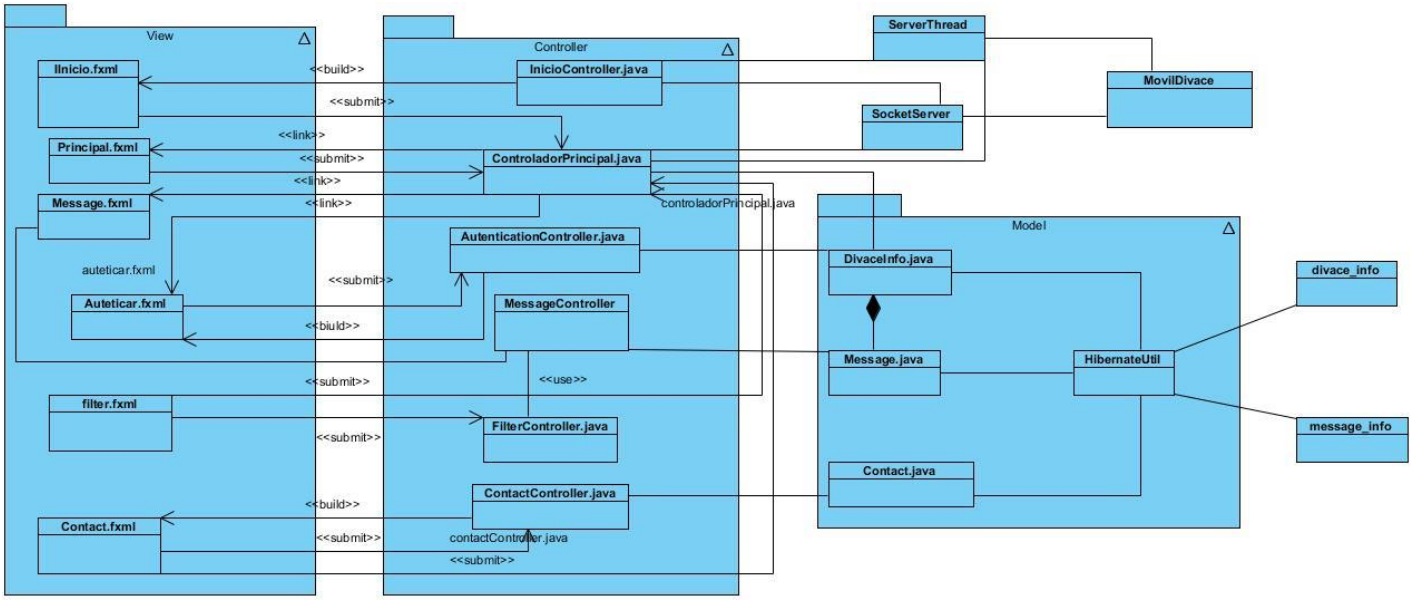


Ilustración 2. Diagrama de clases del diseño

2.4.1. Arquitectura

La arquitectura de software es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema (Reynoso, 2004).

El objetivo principal de la arquitectura del software es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. Para conseguirlo, la arquitectura del software construye abstracciones, materializándolas en forma de diagramas comentados (Reynoso, 2004).

Por la forma en que se encuentra concebido el sistema, fue aplicado el estilo arquitectónico: la arquitectura de llamada y retorno.

Arquitectura de llamada y retorno:

Estilo clásico desde los años 1960. Descomposición jerárquica en subrutinas (componentes) que solucionan una tarea o función definida. Los datos son pasados como parámetros y el manejador principal proporciona un ciclo de control sobre las subrutinas. Reflejan la estructura del lenguaje de programación. Permite al diseñador del software construir una estructura de programa relativamente fácil de modificar y ajustar a escala. Se basan en la bien conocida abstracción de procedimientos/funciones/métodos (David, 2016).

Características (David, 2016)**:**

- Hilo de control simple soportado por los lenguajes de programación.
- Usa una estructura implícita de subsistemas.
- Razonamiento jerárquico, cambios en una subrutina implican cambios en las subrutinas que hacen la invocación.
- Pretenden incrementar el desempeño distribuyendo el trabajo en múltiples procesadores.

Ventajas (David, 2016)**:**

- Utilizados en grandes sistemas de software.
- La descomposición en módulos disminuye la complejidad.
- Persiguen escalabilidad y modificabilidad.

Desventajas (David, 2016)**:**

- Dependencia y acoplamiento entre módulos.
- La reutilización y el mantenimiento son difíciles.

Este estilo fue aplicado en la investigación por la necesidad existente de separar las funciones de la aplicación, principalmente las que se utilizan entre las interfaces y el modelo de datos, para lograr optimizar las peticiones que se realicen por parte de los usuarios. Además, permite que una interfaz de usuario pueda mostrar múltiples vistas de los mismos datos simultáneamente.

2.4.2. Patrón Arquitectónico Modelo-Vista-Controlador

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo (Pablo, 2018).

- El **Modelo** que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- La **Vista**, o interfaz de usuario, que contiene la información que se envía al cliente y los mecanismos de interacción con éste.
- El **Controlador**, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

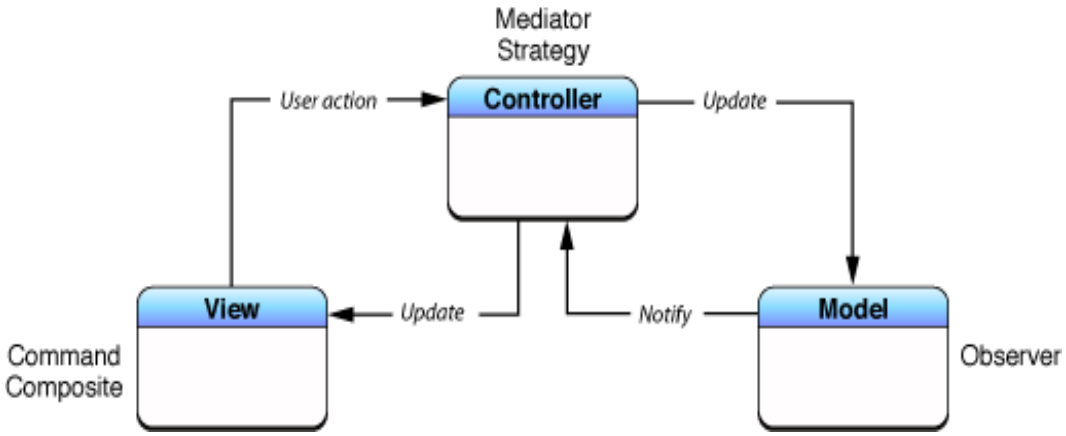


Ilustración 3. Descripción del patrón MVC

Por muy sencillo que parezca un proyecto, siempre es bueno que esté asentado sobre una base sólida y estable sobre el que desarrollarlo. Hay que anticiparse a los posibles cambios y expansiones de una aplicación, de esta forma, se puede organizar la estructura para que sea fácil de modificar y añadir nuevas funcionalidades.

Un proyecto debe ser modular, debe estar conformado por diferentes módulos independientes. Esto permite que la aplicación sea más fácil de mantener, ya que los cambios en una parte causarán un menor impacto en las demás partes al estar aisladas unas de otras. Además, el patrón MVC favorece la escalabilidad de la aplicación. La escalabilidad es el potencial de un sistema para adaptarse y permitir su crecimiento. Cuando se cuenta con una base sólida y estructurada, es más sencillo añadir nuevas funcionalidades. Se sabe exactamente en qué parte de la aplicación se tiene que añadir el código y cómo organizarlo. (Pablo, 2018)

Una de las razones por las que se utiliza el patrón Modelo – Vista – Controlador (MVC) es por el nivel de organización que provee en las aplicaciones informáticas, hace posible que la envergadura con que se desarrolla un sistema informático fluya de forma organizada (Pablo, 2018).

Por esta razón se decide utilizar este patrón en la investigación.

2.4.3. Patrones de diseño

Los patrones de diseño representan soluciones a problemas que surgen cuando se desarrolla un software en un contexto particular. “Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular” (Pressman, 2010). Para realizar el diseño del sistema se han utilizado algunos de los patrones existentes, los cuales se mencionan a continuación:

Patrones GRASP

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP, por sus siglas en inglés), son parejas de problema y solución, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Javier G. , 2014). Los patrones GRASP utilizados en la investigación son:

Experto: Permite asignar una responsabilidad al experto en información, a la clase que cuenta con la información necesaria para cumplir la responsabilidad (Javier G. , 2014). En la aplicación este patrón se pone de manifiesto en las clases *SocketServer* y *ServerThread.*

Bajo Acoplamiento: Este patrón plantea que la dependencia entre las clases que conforman la arquitectura del sistema debe ser mínima, debido a que, a la hora de realizar modificaciones en una clase, no afecte la estructura de las restantes (Javier G. , 2014). En el sistema se concibió mantener separadas las clases pertenecientes al modelo de datos, las vistas de usuario y las clases controladoras.

Alta cohesión: Las responsabilidades que almacena una clase deben ser pequeñas y enfocadas. Los métodos y atributos deben ser sencillos para implementar dichas responsabilidades (Javier G. , 2014) . En el sistema se aplica en las clases HibernateUtil y Autenticar.

Controlador: Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado (Javier G. , 2014) . Este patrón se evidencia en la clase ControladorPrincipal.

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento (Javier G. , 2014). Este patrón fue aplicado en la clase filtroController en los métodos crear_Pagina() y crear_AP_Message().

2.4.4. Modelo de datos

Un modelo de datos es un modelo abstracto que organiza elementos de datos y estandariza cómo se relacionan entre sí y con las propiedades del mundo real. El mismo tiene el propósito de garantizar que los datos persistentes sean almacenados de manera coherente y eficaz, así como definir el comportamiento que debe ser implementado en la base de datos (Pérez Porto Julián, 2016).

A continuación, se muestra el modelo entidad-relación:

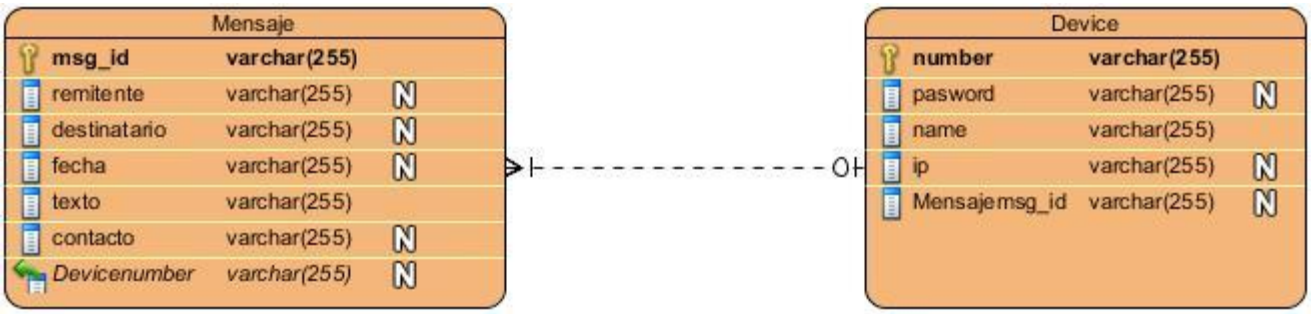


Ilustración 4. Modelo entidad-relación

Descripción de las tablas del modelo entidad-relación:

Tabla 4. Descripción de la tabla Mensaje de la base de datos

Mensaje		
Se corresponde con la clase Message		
Atributo	**Tipo**	**Descripción**
msg_id	String	Es el identificador del mensaje
remitente	String	Es de donde se envía el mensaje
destinatario	String	Es donde se recibe el mensaje
fecha	String	Es la fecha del mensaje
texto	String	Es el cuerpo del mensaje
contacto	String	El nombre del que envió el mensaje

Tabla 5. Descripción de la tabla Device de la base de datos

Device		
Se corresponde con la clase DeviceInfo		
Atributo	**Tipo**	**Descripción**
number	String	Es el número del dispositivo móvil.
pasword	String	Es la contraseña que crea el usuario al conectar su dispositivo a la aplicación de escritorio.
name	String	Es el nombre que se crea el usuario.
ip	String	Es la dirección ip que tiene el dispositivo.

2.5. Conclusiones del capítulo

En el presente capítulo se elabora el diagrama de modelo de dominio que facilita identificar los principales elementos correspondientes al dominio del sistema. Con los requisitos funcionales y no funcionales identificados por el cliente se logra el desarrollo de un sistema de filtrado de SMS que resuelve el problema planteado en la presente investigación. Las historias de usuarios permiten una mejor comprensión de los requisitos funcionales, facilitando la implementación de estos. Con el patrón arquitectónico MVC, se logra separar las clases implementadas para una mejor organización a la hora de realizar algún cambio necesario.

Capítulo 3 Implementación y pruebas

Introducción

En este capítulo se documenta el proceso de implementación de los requisitos identificados durante la realización del diseño. También, se modelan los diagramas de componentes y despliegue. Además, se muestran los resultados de las pruebas de calidad realizadas al sistema, permitiendo de esta manera que las funcionalidades desarrolladas cumplan con los requisitos establecidos y el sistema pueda ser entregado sin ninguna deficiencia a los comentaristas deportivos cubanos.

3.1 Implementación

El modelo de implementación describe cómo los elementos del diseño se implementan en componentes. El propósito principal de este flujo de trabajo es desarrollar la arquitectura y el sistema como un todo. Describe también la organización de los componentes según los mecanismos de estructuración y modularización disponibles en el entorno de desarrollo, el lenguaje de programación utilizado, y la dependencia entre componentes. (Sanz, 2009).

3.1.1 Estándares de codificación utilizados

Los estándares de código, son parte de las llamadas buenas prácticas o mejores prácticas, estas son un conjunto no formal de reglas, que han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo y las cuales, bien aplicadas, pueden incrementar la calidad del código (Merkury, 2017). Seguidamente se especifican algunos estándares de codificación utilizados en la solución (Scott):

- El tamaño máximo de las líneas de código debe ser de cien a ciento veinte caracteres aproximadamente, de manera tal que se garantice la completa visibilidad de las líneas de código sin necesidad de realizar desplazamiento horizontal. Ejemplo:

```
public int compareTo( Message message )
{
    long fecha_Milisec = Long.parseLong(message.getDate());
    Date date_Otro     = new Date(fecha_Milisec);

    fecha_Milisec = Long.parseLong(date);
    Date Mi_date = new Date(fecha_Milisec);
    return date_Otro.compareTo(Mi_date);
}
```

Ilustración 5. Ejemplo de empleo de estándar de codificación

- Los nombres de los atributos, variables y parámetros tendrán todas las letras en minúsculas, en casos de que el nombre de la variable esté compuesto por más de una palabra se adopta el estilo UpperCamelCase y usarán el guión bajo como delimitador entre palabras. Ejemplos:

```
public class Contact {

    String name;
    String number;
```

Ilustración 6. Ejemplo de empleo de estándar de codificación para atributos

```
long fecha_Milisec = Long.parseLong(message.getDate());
Date date_Otro     = new Date(fecha_Milisec);
```

Ilustración 7. Ejemplo de empleo de estándar de codificación para variables

- Sentencias

Cada línea debe contener como máximo una sentencia. Ejemplo:

```
Session session = sessionFactory.getCurrentSession();
session.beginTransaction();
session.saveOrUpdate(message);
```

Ilustración 8. Ejemplo de empleo de estándar de codificación

Las sentencias pertenecientes a un bloque de código estarán tabuladas un nivel más a la derecha con respecto a la sentencia que las contiene. Ejemplo:

```
for ( int i = 0; i < size; i++ )
{
    VBox disp = (VBox) dispositivos.get(i);
    if ( disp.getId().equals(deviceInfo.toString()) )
    {
        int posDisp = i;
        Platform.runLater(() -> dispositivos.remove(posDisp));
        return;
    }
}
```

Ilustración 9. Ejemplo de empleo de estándar de codificación

El carácter inicio de bloque "{" debe situarse en una nueva línea debajo de la línea que inicia el bloque. El carácter final de bloque "}" debe situarse en una nueva línea tras la última línea del bloque y alineada con respecto al primer carácter de dicho bloque. Ejemplo:

```
public static void eliminarMensajes( List<Message> sms )
{
    Session session = sessionFactory.getCurrentSession();
    session.beginTransaction();
    for ( Message m : sms )
        session.delete(m);
    session.getTransaction().commit();
}
```

Ilustración 10. Ejemplo de empleo de estándar de codificación

Todas las sentencias de un bloque deben encerrarse entre llaves "{...}", aunque el bloque conste de una única sentencia. Esta práctica permite añadir código sin cometer errores accidentalmente al olvidar añadir las llaves. Ejemplo:

```
for ( Message message : messages )
{
    session.saveOrUpdate(message);
}
```

Ilustración 11. Ejemplo de empleo de estándar de codificación

La sentencia "try/catch" siempre debe tener el formato siguiente:

```
try
{
    sentencias;
} catch (ClaseException e)
{
    sentencias;
}
```

Ejemplo:

```
try
{
    Session session = sessionFactory.getCurrentSession();
    session.beginTransaction();
    for ( Message message : messages )
    {
        session.saveOrUpdate(message);
    }
    session.getTransaction().commit();
} catch ( Exception e )
{
    System.out.println(e.getMessage());
}
```

Ilustración 12. Ejemplo de empleo de estándar de codificación

En el bloque "catch" siempre se imprimirá una traza de error indicando el tipo de excepción generada y posteriormente se elevará dicha excepción al código invocante, salvo que la lógica de ejecución de la aplicación no lo requiera.

3.1.2. Diagrama de componentes

La realización de este diagrama permite mostrar como el sistema está dividido en componentes y las dependencias entre ellos. Proveen una vista arquitectónica de alto nivel del sistema ayudando a tomar decisiones respecto a las tareas de implementación y a visualizar el camino de la implementación. Un componente es una parte modular, desplegable y reemplazable de un sistema que encapsula implementación y expone un conjunto de interfaces (Fabio, 2011).

A continuación, se muestra el diagrama de componentes del Sistema de filtrado de SMS para los comentaristas deportivos cubanos, el cual se diseña antes de la implementación de dicho sistema.

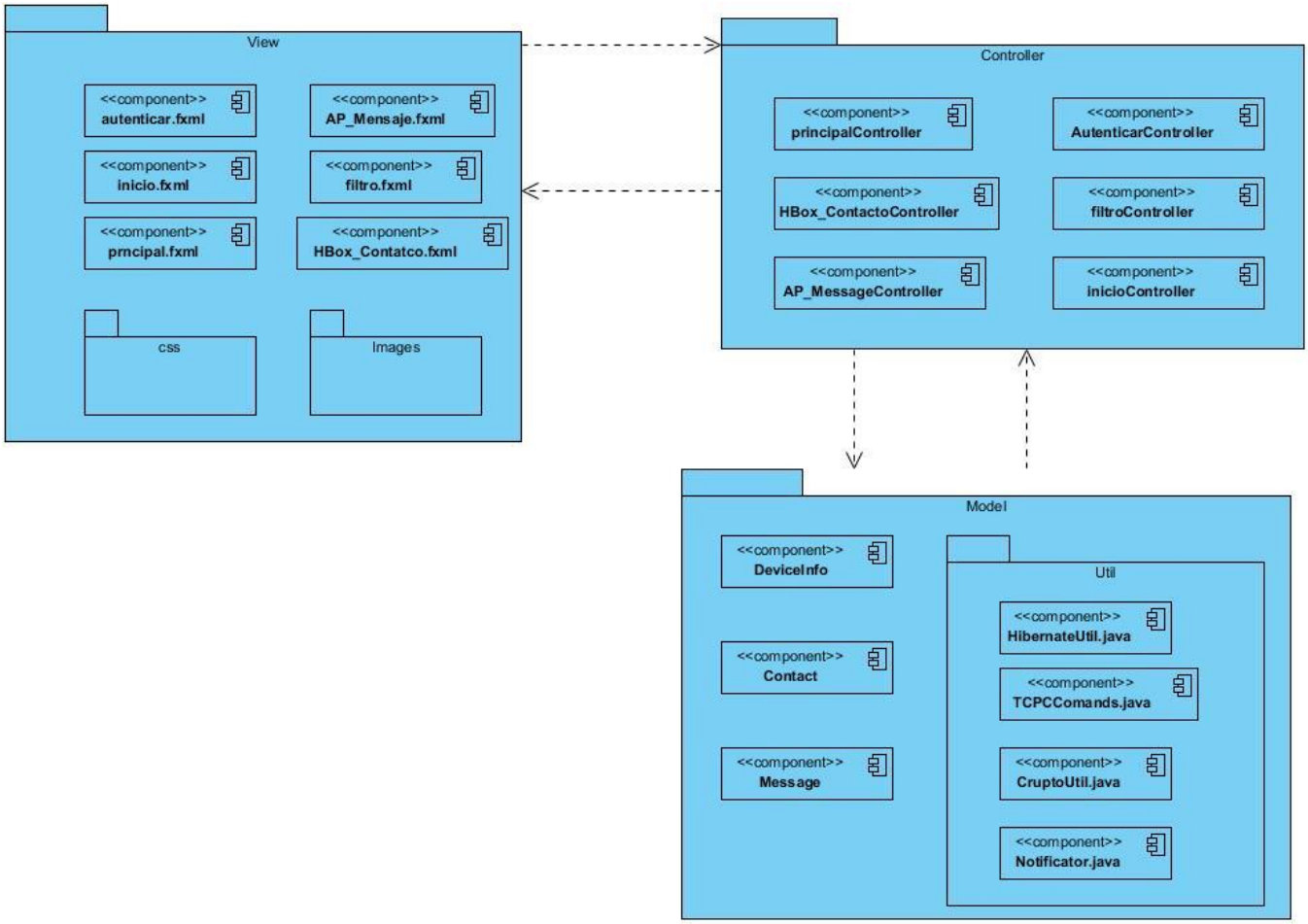


Ilustración 13. Diagrama de componentes

3.1.3. Diagrama de despliegue

El diagrama de despliegue es un diagrama modelado en UML que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Este diagrama muestra las relaciones físicas entre los componentes de hardware y software en el sistema final. Describe la topología del sistema: la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos (Johana, 2013).

A continuación, se muestra el diagrama de despliegue del Sistema de filtrado de SMS para los comentaristas deportivos cubanos, el cual se diseña antes de la implementación de dicho sistema:

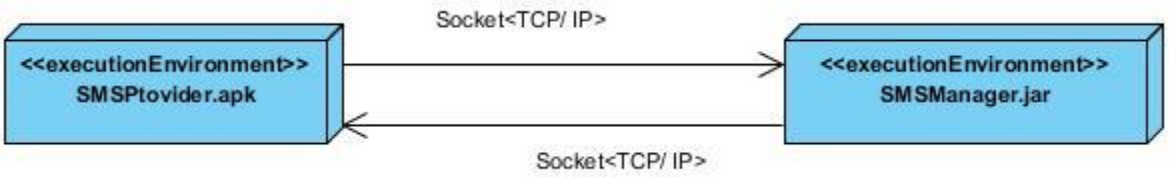


Ilustración 14. Diagrama de despliegue

3.2. Principios para la aplicación de pruebas

Para obtener un producto terminado con la calidad requerida y que cumpla con las expectativas del cliente, en el desarrollo de un software, un aspecto fundamental es la realización de las pruebas. Su objetivo es localizar el mayor número de deficiencias lo antes posible para corregirlas, ahorrando tiempo y recursos en el desarrollo (Testing, 2015). Por lo analizado anteriormente, en el presente capítulo se desarrollarán pruebas, a través de la aplicación de casos de pruebas, de caja negra y de caja blanca con el objetivo de verificar y validar los resultados de la implementación.

A continuación, se exponen diez de los principios de realización de pruebas más relevantes que se tienen en cuenta a la hora de diseñar y aplicar pruebas de software a un sistema. Estos principios son necesarios tenerlos en cuenta en el momento de diseñar los casos de pruebas ya que orientan y guían a los responsables de las pruebas (Javier Z. , 2013).

1. La prueba puede ser usada para mostrar la presencia de errores, pero nunca de su ausencia.
2. La principal dificultad del proceso de prueba es decidir cuándo parar.
3. Evitar casos de pruebas no planificados, no reusables y triviales a menos que el programa sea verdaderamente sencillo.
4. Una parte necesaria de un caso de prueba es la definición del resultado esperado.
5. Los casos de pruebas tienen que ser escritos no solo para condiciones de entrada válidas y esperadas sino también para condiciones no válidas e inesperadas.

6. Los casos de pruebas tienen que ser escritos para generar las condiciones de salida deseadas.
7. El número de errores sin descubrir es directamente proporcional al número de errores descubiertos.
8. Las pruebas deberían empezar por “lo pequeño” y progresar hacia “lo grande”.
9. Con la excepción de las pruebas de unidad e integración, un programa no deberá ser probado por la persona u organización que lo desarrolló.
10. Las pruebas deberían ser realizadas por un equipo independiente.

3.3. Niveles de Pruebas

La estrategia que se ha de seguir a la hora de evaluar dinámicamente un sistema de software debe permitir comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Más concretamente, los pasos a seguir son: pruebas unitarias, pruebas de integración, pruebas del sistema y pruebas de aceptación (Ruiz, 2010).

Pruebas Unitarias: centra el proceso de verificación en la menor unidad del diseño del software: el componente de software o módulo. Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa que está siendo probada (Ruiz, 2010). Durante el desarrollo se aplica este tipo de pruebas al comprobar el correcto funcionamiento de cada uno de los requisitos implementados.

Pruebas del Sistema: está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas (Ruiz, 2010). Estas pruebas fueron aplicadas posterior a la implementación, utilizando la técnica de partición de equivalencia con diseños de casos de prueba.

3.4. Métodos de prueba

Existen dos métodos de pruebas fundamentales: el método de caja negra y el de caja blanca. El método de caja negra consiste en pruebas que se realizan sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software sean operativas. Esta prueba examina algunos aspectos del sistema sin tener en cuenta la estructura interna del software. Además, permite obtener un conjunto de

condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Tipos de pruebas de software, s.f.).

Por otro lado, en las pruebas de caja blanca se comprueban los caminos lógicos del software proponiendo casos de prueba donde se ejerciten conjuntos específicos de condiciones y/o bucles. Además, se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con lo esperado o mencionado (Brian, s.f.) (Tipos de pruebas de software, s.f.).

3.4.1. Prueba de caja negra

Las pruebas de caja negra no consideran la codificación dentro de sus parámetros a evaluar, es decir, no están basadas en el diseño interno del programa. El objetivo que tienen estas pruebas es el de revelar el incorrecto o incompleto funcionamiento del sistema, así como los errores de interfaz, rendimiento y errores de inicialización y terminación. Estas pruebas se enfocan en los requerimientos establecidos y en las funcionalidades del sistema (Gustavo T. , 2017).

A través de esta técnica, los casos de prueba desarrollados en el presente trabajo pretenden demostrar que las funciones que ofrece la aplicación son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Los resultados de esta prueba generalmente han sido satisfactorios. Los casos de prueba para las historias de usuario se encuentran en el anexo #2.

A continuación, se muestran ejemplos de los casos de prueba que se ejecutaron:

Tabla 6. Diseño de caso de prueba correspondiente al RF17

ID del escenario	Escenario	Respuesta del sistema	Flujo central
EC 1.1 Filtrar SMS.	El usuario escribe en la barra de búsqueda el filtro	El sistema muestra todos los SMS almacenados que coincidan con el filtro. Por cada SMS muestra, además: número en caso de que no esté registrado, o en caso de que si el nombre de contacto, el texto del mensaje y la fecha y hora en que fue recibida.	Inicio/Teclear filtro

Descripción de las variables

Tabla 7. Descripción de las variables del caso de prueba correspondiente al RF17

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Filtros	Campo de texto	Si	Campo que permite seleccionar el criterio de búsqueda deseado por el usuario, puede ser introducido cualquier carácter alfanumérico y especial. - Usuario (Campo de selección). - IP (Campo de selección).

Tabla 8. Diseño de caso de prueba correspondiente al RF11

ID del escenario	Escenario	Respuesta del sistema	Flujo central
EC 1.1 Autenticación para un dispositivo registrado.	El usuario debe poder conectarse a través de una contraseña almacenada para el número del dispositivo móvil.	El sistema muestra una ventana de autenticación donde se ofrece un campo para la contraseña.	Inicio/Dispositivo/Autenticar/Principal con SMS e información**.**
EC 1.2 Autenticación para un dispositivo no registrado.	El usuario debe poder guardar una contraseña para el acceso al número de teléfono solicitado.	El sistema muestra una ventana de notificación indicando que la contraseña es incorrecta.	Inicio/Dispositivo/Registrar/Principal con SMS e información.
EC 1.3 Contraseña incorrecta.	El usuario escribe su contraseña para autenticarse, pero esta no es correcta.	El sistema muestra una ventana de notificación indicando que la contraseña es incorrecta.	Inicio/Dispositivo/Autenticar/Ventana de error.
EC 1.4 Contraseñas no coinciden**.**	La contraseña introducida en el campo para registrar no coincide con la introducida en el campo para validar.	El sistema muestra una ventana de notificación indicando que las contraseñas no coinciden.	Inicio/Dispositivo/Registrar/Ventana de error

Descripción de las variables

Tabla 9. Descripción de las variables del caso de prueba correspondiente RF11

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Contraseña	Campo de texto contraseña	No	Campo de carácter obligatorio que representa la contraseña de logueo del usuario
2	Nueva contraseña	Campo de texto contraseña	No	Campo que representa la nueva contraseña de un usuario.
3	Validar contraseña	Campo de texto contraseña	No	Campo que representa la validación de la nueva contraseña entrada por un usuario.

Clasificación de las no conformidades

Una no conformidad es un fallo en el sistema de gestión de la calidad que puede producirse por varias razones: no alcanzar el nivel de aceptación establecido en un determinado indicador y errores en la documentación del sistema. Se trata de una desviación entre lo que hay escrito y lo que ha ocurrido. Este fallo queda registrado en un informe y se establecen las acciones preventivas y correctivas necesarias para arreglar lo que no funcione y evitar que vuelva a ocurrir. Las mismas se clasifican de acuerdo al nivel de importancia en (Intranet SGC, s.f.):

1. Significativas: Son aquellas que afectan la calidad del producto o servicio de manera visible, impidiendo o no el cumplimiento de algún requisito.

2. No significativas: Son aquellas que resultan menos visibles, que no atentan el cumplimiento de algún requisito.

3. Recomendaciones: Son aquellas que quedan en función de la apreciación del probador para oportunidades de mejoras del producto o servicio.

Para corregir los errores cometidos durante la implementación de la aplicación se realizaron tres iteraciones de pruebas, en las cuales se detectaron un conjunto de no conformidades. En la primera iteración se detectaron 8 no conformidades (4 de tipo 1, 3 de tipo 2 y una de tipo 3), en la segunda iteración 5 (2 de tipo 1, 2 de tipo 2 y 1 de tipo 3) y en la tercera iteración no fue detectada ninguna. El comportamiento de las no conformidades según el tipo se puede ver en el siguiente gráfico.

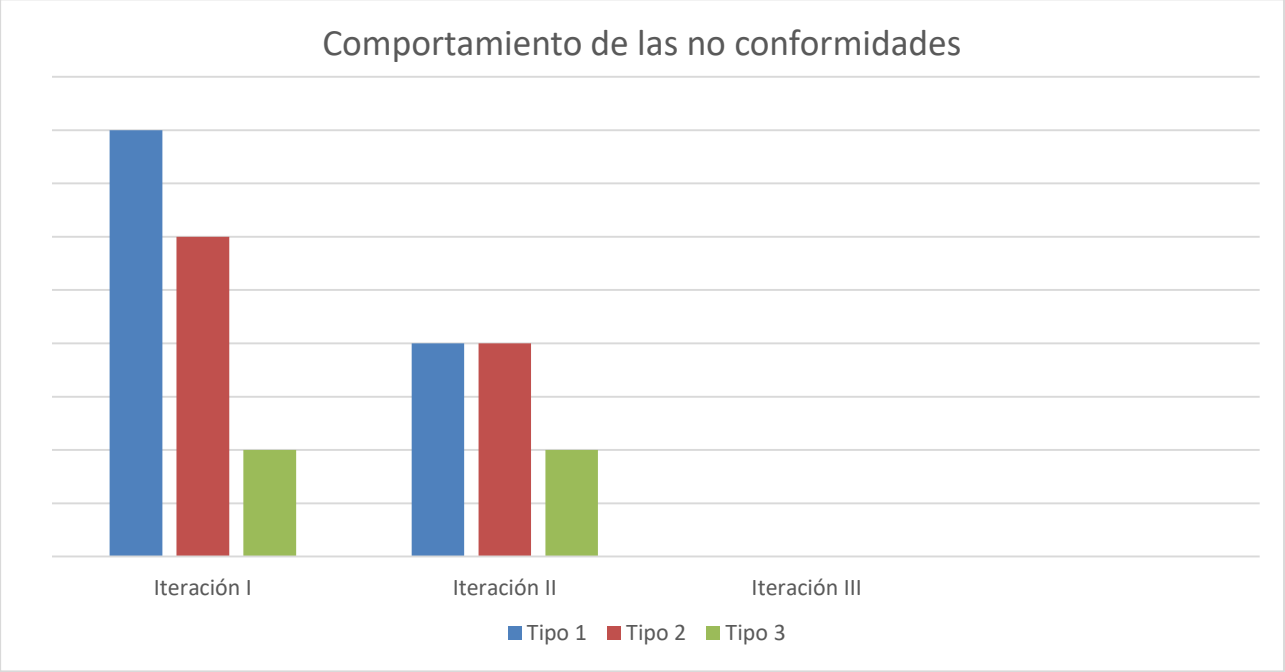


Ilustración 15. Comportamiento de las no conformidades durante las pruebas de software

3.4.2. Prueba de caja blanca

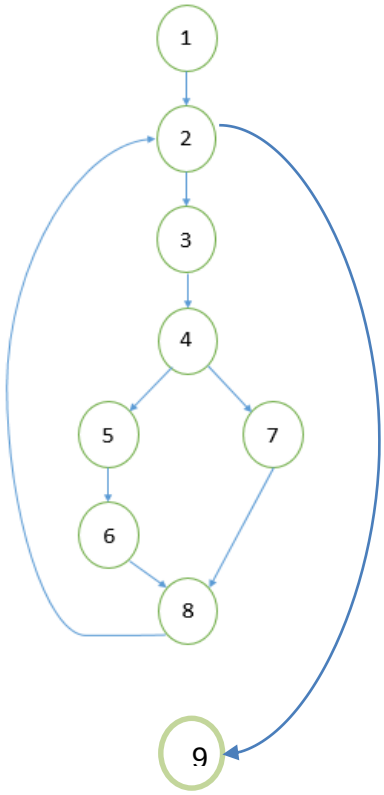
En la prueba de caja blanca se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, bucles y condiciones; y examinando el estado del programa en varios puntos (Testing de caja blanca, 2011).

En la siguiente tabla se muestra el resultado de aplicar la técnica del camino básico al método delete_Device:

Tabla 10.Técnica del camino básico al Método delete_Device.

<table>
<tr><th colspan="2">Caso de prueba unitaria: Método delete_Device.</th></tr>
<tr><td colspan="2">Descripción: Prueba para el método delete_Device.</td></tr>
<tr><td>Código:</td><td>Resultado de la prueba:</td></tr>
<tr><td>

```
//Eliminar dispositivo de la lista al desconectarse.
public static void delete_Device( DeviceInfo deviceInfo )
{
    int size = dispositivos.size();
    for ( int i = 0; i < size; i++ )
    {
        VBox disp = (VBox) dispositivos.get(i);
        if ( disp.getId().equals(deviceInfo.toString()) )
        {
            int posDisp = i;
            Platform.runLater(() -> dispositivos.remove(posDisp));
            return;
        }
    }
}
}
```

</td><td> 
</td></tr>
<tr><td>Complejidad ciclomática:</td><td>Número de regiones: 3
V(G)=aristas-nodos+2
= 10-9+2=3
V(G)=nodo_predicado+1
= 2+1=3</td></tr>
<tr><td>Caminos linealmente independientes:</td><td>1-2-9
1-2-3-4-7-8-2-9
1-2-3-4-5-6-8-2-9</td></tr>
</table>

Seguidamente se especifican los casos de prueba para cada uno de los caminos obtenidos. A medida de su ejecución se comparan los resultados esperados y se verifican si se ejecutan todas las instrucciones al menos una vez.

Tabla 11. Caso de prueba para el camino básico 1

Caso de prueba para el camino básico 1 (1-2-9)	
Descripción	Comprueba que el dispositivo se haya desconectado.
Condiciones de ejecución	El dispositivo no se ha desconectado.
Entrada	Información del dispositivo
Resultado esperado	No se elimina el dispositivo de la lista de dispositivos conectados.

Tabla 12. Caso de prueba para el camino básico 2

Caso de prueba para el camino básico 2 (1-2-3-4-7-8-2-9)	
Descripción	Comprueba que el dispositivo se haya desconectado.
Condiciones de ejecución	El dispositivo se ha desconectado.
Entrada	Información del dispositivo incorrecta
Resultado esperado	No se elimina el dispositivo del sistema, pues sus datos de entrada son incorrectos.

Tabla 13. Caso de prueba para el camino básico 3

Caso de prueba para el camino básico 3 (1-2-3-4-5-6-8-2-9)	
Descripción	Comprueba que el dispositivo se haya desconectado.
Condiciones de ejecución	El dispositivo se ha desconectado.
Entrada	Información del dispositivo
Resultado esperado	Se elimina el dispositivo de la lista de dispositivos conectados.

3.5. Conclusiones del capítulo

En el presente capítulo la utilización de estándares de código para la implementación de la propuesta de solución posibilitó adoptar una estructura homogénea que facilita la comunicación y una menor cantidad de errores; logrando un código más limpio y fácil de mantener. Se especifica cómo está construido el sistema a partir del diagrama de componentes, lo cual permite identificar con claridad la estructura y relaciones que existen entre los diferentes componentes empleados en la implementación del sistema. Se realizó el diagrama de despliegue permitiendo conocer el hardware necesario con que debe contar el cliente para que funcione el sistema implementado. Las pruebas realizadas permitieron detectar los errores presentes, corregirlos en el menor tiempo posible, para entregar al cliente un producto con calidad.

Conclusiones generales

Al finalizar el desarrollo de la aplicación se arriba a las siguientes conclusiones:

- Se logró identificar las características principales que se utilizaron en el desarrollo del sistema a través del estudio de las diferentes tecnologías, sistemas de filtrados de SMS y aplicaciones existentes.
- Se facilitó a los comentaristas deportivos cubanos la búsqueda de los SMS por varios criterios mediante el desarrollo de un sistema de filtrado de SMS que consta de una aplicación Android para dispositivos móviles inteligentes, la cual envía información a una aplicación de escritorio multiplataforma.
- Las pruebas de software realizadas al Sistema de filtrado de SMS permitieron comprobar su correcto funcionamiento.

Recomendación

- Añadir la conexión por USB como segundo tipo de conexión entre los dispositivos móviles y la aplicación de escritorio.

Referencias bibliográficas

Mercacel. Servicios. . (13 de septiembre de 2009). Obtenido de http://www.mercacel.com/servicios_mercacel.php

Nuestro Site. Plataforma de envío de mensajes SMS/MMS. (septiembre de 2009). Obtenido de http://www.nuestrosite.com.mx/productos/plataformadeenviossms

Une Internet. Servicios SMS. . (2009). Obtenido de www.uneinternet.es/servicios_sms.html

(2014). Obtenido de Visual Paradigm: https://www.visual-paradigm.com/

(29 de noviembre de 2017). Obtenido de ¿Qué tipos de filtros antispam existen?: https://www.mdirector.com/email-marketing/tipos-de-filtros-antispam.html

(4 de mayo de 2017). Obtenido de Getting Started with IntelliJ IDEA: http://kotlinlang.org/docs/tutorials/getting-started.html

Angel, R. L. (7 de marzo de 2017). Obtenido de Cómo saber si estoy en una lista negra de Spam y cómo salir de ella: https://www.teenvio.com/es/consejos/como-saber-si-estoy-en-una-lista-negra-de-spam/

Aniversario del SMS: cómo se logró enviar el primer mensaje de texto de la historia hace 25 años. (3 de diciembre de 2017). Obtenido de http://www.bbc.com/mundo/noticias-42214176

Ardévol , F. M. (octubre de 2011). *Las comunicaciones móviles y el desarrollo socioeconómico: Una perspectiva latinoamericana*. Obtenido de https://unchronicle.un.org/es/article/las-comunicaciones-m-viles-y-el-desarrollo-socioecon-mico-una-perspectiva-latinoamericana

AS. (8 de 12 de 2016). Obtenido de Android Studio: ventajas, desventajas y principales características: https://androidstudiofaqs.com/conceptos/ventajas-desventajas-android-studio

Brian, M. (s.f.). *New Models for test development*. Obtenido de http://www.exampler.com/testing-com/writings/new-models.pdf.

Bruno, C. (25 de Agosto de 2017). *Requerimientos, Técnicas de Especificación y metodologías Ágiles* . Obtenido de Historias de Usuario: https://gist.github.com/brunocascio/09ba6c90842948bfb9ad

Carlos, G. P. (9 de mayo de 2006). Obtenido de https://www.adictosaltrabajo.com/tutoriales/hsql/

César, K. (2016). Obtenido de ¿Qué es y para qué sirve UML? Versiones de UML (Lenguaje Unificado de Modelado). Tipos de diagramas UML.: https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46&Itemid=163

David, Z. (4 de octubre de 2016). *Estilo de artquitectura de llamada y retorno*. Obtenido de https://prezi.com/gjtimc-be3c_/estilo-de-artquitectura-de-llamada-y-retorno/

Esendex. (2017). Obtenido de https://www.esendex.es/blog/post/repasamos-nuestros-mejores-momentos-del-2017/

Fabio, C. Q. (7 de abril de 2011). *Diagrama de componentes* . Obtenido de https://es.slideshare.net/uitron/diagrama-de-componentes-7551535

Fernando, G. (17 de enero de 2017). *Curso Completo de Java.* Obtenido de https://www.tutellus.com/tecnologia/desarrollo-web/curso-completo-de-java-2017-13080

Garino, G. C. (15 de julio de 2016). Obtenido de http://www.unidiversidad.com.ar/las-tic-su-importancia-en-la-actualidad-y-el-mercado-laboral

Gustavo, R. (25 de octubre de 2013). *MOTORES DE BASE DE DATOS.* Obtenido de https://prezi.com/ry9ckaivktcx/motores-de-base-de-datos/

Gustavo, T. (26 de febrero de 2017). *Pruebas de Caja Negra y un enfoque práctico*. Obtenido de https://testingbaires.com/pruebas-caja-negra-enfoque-practico/

Intranet SGC. (s.f.). Obtenido de https://sites.google.com/a/iesprimerodemayo.com/sgc/preguntas-as-frecuentes/%C2%BFqueesunanoconformidad.

Jacobson, I. &. (2000). Obtenido de El Lenguaje Unificado de Modelado.Manual de Referencia. .

Jaime, R. (8 de septiembre de 2014). *Métodos Empíricos*. Obtenido de https://es.slideshare.net/jaimericardorodriguez5/metodos-empiricos

Javier, G. (5 de agosto de 2014). *Los patrones GRASP*. Obtenido de http://www.javiergarzas.com/2014/08/los-patrones-grasp.html

Javier, S. (3 de 12 de 2012). Obtenido de Las diez mejores aplicaciones Android de comunicación : http://sevilla.abc.es/mobility/las_mejores_app/android/las-mejores-app-android/las-diez-mejores-aplicaciones-android-de-comunicacion/

Javier, S. (18 de enero de 2017). Obtenido de Las mejores aplicaciones con filtros artísticos para tu Android : evilla.abc.es/mobility/las_mejores_app/android/las-mejores-app-android/las-mejores-aplicaciones-con-filtros-artisticos-para-tu-android/

Javier, Z. (2013). *Principios Fundamentales del Proceso de Pruebas*. Obtenido de https://pruebasdelsoftware.wordpress.com/2013/01/07/principios-fundamentales-del-proceso-de-pruebas/.

Jennifer, F. (25 de octubre de 2013). *MÉTODO ANALÍTICO-SINTÉTICO*. Obtenido de https://prezi.com/aj7ol61na2bb/metodo-analitico-sintetico/

Jiménez, A. (16 de septiembre de 2016). Obtenido de Spam. ¿Cómo funcionan los filtros bayesianos?: https://www.xatakaciencia.com/matematicas/spam-como-funcionan-los-filtros-bayesianos

Johana, S. (12 de abril de 2013). *UML: Diagrama de Despliegue* . Obtenido de http://umldiagramadespliegue.blogspot.com/

Juan, G. (29 de diciembre de 2016). *Android en 2017 y lo que esperamos en un futuro más lejano.* Obtenido de https://www.cnet.com/es/noticias/android-2017-futuro-novedades-android-8/

Julio, C. A. (11 de noviembre de 2016). Redes sociales y Tecnologías de la Información y la Comunicación en Educación. *Revista de Educación a Distancia.* , pág. http://www.um.es/ead/red/.

King Gavin, B. C. (15 de septiembre de 2010). Obtenido de HIBERNATE - Persistencia relacional para Java idiomático: https://docs.jboss.org/hibernate/orm/3.5/reference/es-ES/html/

Luis, O. (enero de 2018). *7 aplicaciones para bloquear llamadas y evitar acosos*. Obtenido de https://www.androidpit.es/aplicaciones-bloquear-llamadas

Merkury. (12 de enero de 2017). *Estándares de codificación*. Obtenido de https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/

Metodología de Desarrollo de Software. (2017). Obtenido de https://www.uladech.edu.pe/images/stories/universidad/documentos/2018/metodologia-desarrollo-software-v001.pdf

Pablo, D. (23 de marzo de 2018). *Descubre la arquitectura MVC*. Obtenido de https://openclassrooms.com/courses/contribuye-a-proyectos-de-codigo-abierto-en-github/el-patron-modelo-vista-controlador-mvc

Paul, F. (26 de noviembre de 2013). Obtenido de Diferentes tipos de filtros anti-spam: http://blog.es.mailify.com/envio-emailing/diferentes-tipos-de-filtros-anti-spam/

Pérez Porto Julián, G. A. (2016). Obtenido de https://definicion.de/sms/

Picajoso. (4 de marzo de 2010). *35 motores de Bases de Datos*. Obtenido de https://www.muylinux.com/2010/03/04/35-motores-de-bases-de-datos-open-source/

Pressman, R. (2010). Obtenido de Ingenería de software.Un enfoque práctico.
Rajeev, K. S. (1 de junio de 2017). *Writing your first desktop application in JavaFX.* Obtenido de https://www.callicoder.com/javafx-desktop-application-development-tutorial/
Reynoso, C. B. (2004). *Introducción a la Arquitectura de Software.* Buenos Aires.
Ronald, H. (2017). *Teleios Systems Limited*.
Ruiz, R. T. (2010). *Las pruebas de software y su importancia en las organizaciones.*
Salomón, P. O. (12 de diciembre de 2017). *La telefonía celular en Cuba cumple 26 años*. Obtenido de http://www.cubadebate.cu/especiales/2017/12/12/la-telefonia-celular-en-cuba-cumple-26-anos/
Santiago, S. (13 de marzo de 2018). Obtenido de Cómo bloquear SMS y llamadas?: http://www.androidjefe.com/bloquear-sms-llamadas-samsung/
Sanz, L. (2009). *Proceso Unificado: Implementación.*
Scott, H. (s.f.). *Convenciones de Código para el lenguaje de programación Java.*
SMS empresas. Servicios SMS para empresas. (s.f.). Obtenido de https://www.altiria.com/servicios-sms-para-empresas/
Sommerville, I. (2007). *Ingenería de software.*
Sommerville, I. (2007). *Ingenería de software(8ed).*
Standby. (4 de febrero de 2017). *SMS Spoofing*. Obtenido de Amino: https://aminoapps.com/c/world-hacking/page/blog/sms-spoofing/bo3p_LnhouBP8KEGxZ7rDkEkmV33lkM0QQ
Support. (2017). Obtenido de Organizar tus mensajes utilizando filtros: https://support.mozilla.org/es/kb/organizar-tus-mensajes-utilizando-filtros
Tamara, R. S. (6 de marzo de 2015). *Metodología de desarrollo para la Actividad productiva de la UCI. .*
Testing de caja blanca. (11 de mayo de 2011). Obtenido de https://jummp.wordpress.com/2011/05/21/testing-de-caja-blanca-white-box-testing/
Testing, P. (11 de febrero de 2015). *Software QA – ¿Cuáles son los tipos de pruebas software?* . Obtenido de https://www.panel.es/blog/software-qa-cuales-son-los-tipos-de-pruebas-software/
Thorin, K. (13 de julio de 2015). Obtenido de La mejor aplicación para gestionar contactos en el iPhone: https://es.gizmodo.com/la-mejor-aplicacion-para-gestionar-contactos-en-el-ipho-1717556108
Tipos de pruebas de software. (s.f.). Obtenido de http://es.slideshare.net/GuillermoLemus/tipos-de-pruebas-de-software.
todoteach. (2016). Obtenido de NQ Call Blocker: https://www.todotech.com/android/apps/n45/nq-call-blocker.html
Valeria, R. (25 de febrero de 2015). *Método teórico histórico-logico*. Obtenido de https://prezi.com/l1trei4ha4w6/metodo-teorico-historico-logico/
Vlad, L. (s.f.). *Bloqueador de llamadas y SMS - Calls Blacklist APK*. Obtenido de https://apkpure.com/es/calls-blacklist-call-blocker/com.vladlee.easyblacklist
Vodafone España. (2017). Obtenido de https://www.vodafone.es/c/conocenos/es/vodafone-espana/

Anexos

Anexo #1: Historias de usuario

Tabla 14. Descripción de la Historia de usuario del RF1

Número: RF1	**Nombre del requisito:** Permitir conexión entre dispositivo móvil y servidor a través de red Wifi.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: Fallo en la Wifi	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir establecer una conexión entre un dispositivo móvil y un servidor desde la aplicación de escritorio a través de la red wifi. **2- Acciones para lograr el objetivo (precondiciones y datos):** - Debe estar activa una red Wifi y debe haber algún dispositivo conectado. **4- Flujo de la acción a realizar:** Al cargar la aplicación se le lista al usuario los dispositivos móviles conectados. El usuario selecciona el dispositivo con el cual desea establecer la conexión. Se establece la conexión.	
Observaciones:	
Prototipo de interfaz: No requiere prototipo de interfaz	

Tabla 15. Descripción de la Historia de usuario del RF2

Número: RF2	**Nombre del requisito:** Permitir varias conexiones simultáneas.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir el manejo de varias conexiones al mismo tiempo **2- Acciones para lograr el objetivo (precondiciones y datos):** - Debe haber más de un dispositivo móvil conectado. **3- Flujo de la acción a realizar:** Al mostrarse la vista se listan los dispositivos móviles conectados a la aplicación.	
Observaciones:	
Prototipo de interfaz:	

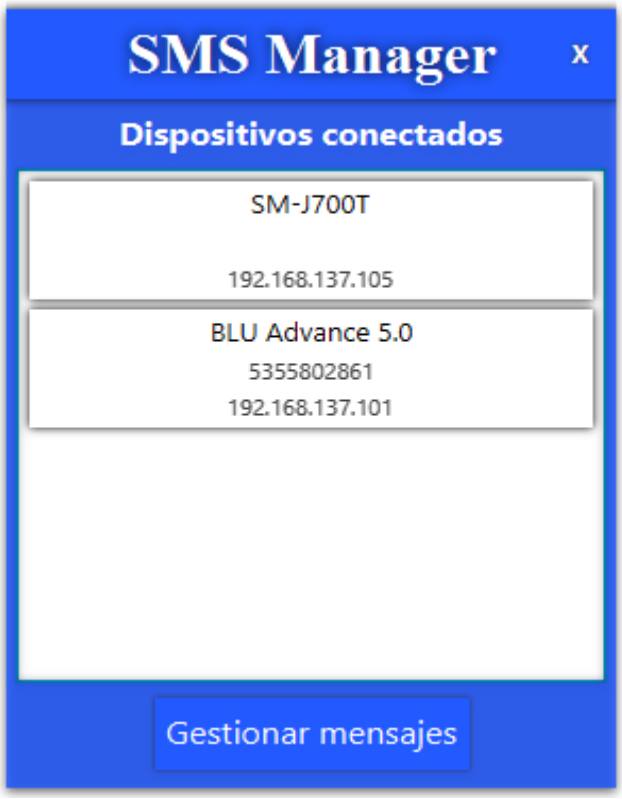

SMS Manager
x
Dispositivos conectados
SM-J700T
192.168.137.105
BLU Advance 5.0
5355802861
192.168.137.101
Gestionar mensajes

Tabla 16. Descripción de la Historia de usuario del RF3

Número: RF3	**Nombre del requisito:** Enviar información del dispositivo conectado.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Enviar al servidor los datos del dispositivo móvil. **2- Acciones para lograr el objetivo (precondiciones y datos):** - El dispositivo móvil debe estar conectado a la misma wifi que el servidor. **3- Flujo de la acción a realizar:** El usuario, de la aplicación de escritorio debe seleccionar, una vez conectados, el botón Gestionar mensajes, simultáneamente el sistema mostrará una notificación detallando el proceso.	
Observaciones:	
Prototipo de interfaz: No presenta prototipo de interfaz.	

Tabla 17. Descripción de la Historia de usuario del RF4

Número: RF4	**Nombre del requisito:** Enviar contactos del dispositivo conectado.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Enviar al servidor los contactos contenidos en el móvil. **2- Acciones para lograr el objetivo (precondiciones y datos):** - El dispositivo móvil debe estar conectado a la misma wifi que el servidor. **3- Flujo de la acción a realizar:** El usuario, de la aplicación de escritorio debe seleccionar, una vez conectados, el botón Gestionar mensajes, simultáneamente el sistema mostrará una notificación detallando el proceso.	
Observaciones:	
Prototipo de interfaz: No requiere prototipo de interfaz	

Tabla 18. Descripción de la Historia de usuario del RF5

Número: RF5	**Nombre del requisito:** Enviar mensajes del dispositivo conectado.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** - Enviar al servidor los contactos contenidos en el móvil. **2- Acciones para lograr el objetivo (precondiciones y datos):** - El dispositivo móvil debe estar conectado a la misma wifi que el servidor. **3- Flujo de la acción a realizar:** El usuario, de la aplicación de escritorio debe seleccionar, una vez conectados, el botón Gestionar mensajes, simultáneamente el sistema mostrará una notificación detallando el proceso.	
Observaciones:	
Prototipo de interfaz: No requiere prototipo de interfaz	

Tabla 19. Descripción de la Historia de usuario del RF6

Número: RF6	Notificar al servidor de la llegada de mensajes nuevos, además de enviarlos.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Notificar al servidor la llegada de mensajes nuevos al dispositivo. **2- Acciones para lograr el objetivo (precondiciones y datos):** -El dispositivo debe estar conectado al servidor. -El dispositivo debe recibir un mensaje nuevo. **3- Flujo de la acción a realizar:** Al recibir un mensaje nuevo se envía al servidor una notificación además de enviar el mensaje al servidor.	
Observaciones:	
Prototipo de interfaz: No presenta interfaz visual	

Tabla 20. Descripción de la Historia de usuario del RF7

Número: RF7	**Nombre del requisito:** Encriptar información enviada a través de la conexión Wifi.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Encriptar los mensajes que son enviados a través de la Wifi. **2- Acciones para lograr el objetivo (precondiciones y datos):** - El dispositivo debe estar conectado al servidor mediante la wifi. - Debe realizarse algún envío de mensajes. **4- Flujo de la acción a realizar:** Previo al envío de mensajes estos son encriptados.	
Observaciones:	
Prototipo de interfaz: No presenta interfaz visual	

Tabla 21. Descripción de la Historia de usuario del RF8

Número: RF8	**Nombre del requisito:** Permitir búsqueda de servidor disponible en la subred del dispositivo móvil.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permite en el dispositivo móvil seleccionar la wifi del servidor al cual conectarse **2- Acciones para lograr el objetivo (precondiciones y datos):** - Debe haber alguna conexión wifi disponible en el servidor. **4- Flujo de la acción a realizar:** El usuario selecciona la aplicación de conexión automática. Se le muestra la conexión wifi del servidor disponible, el usuario selecciona y presiona conectar.	
Observaciones:	
Prototipo de interfaz:	

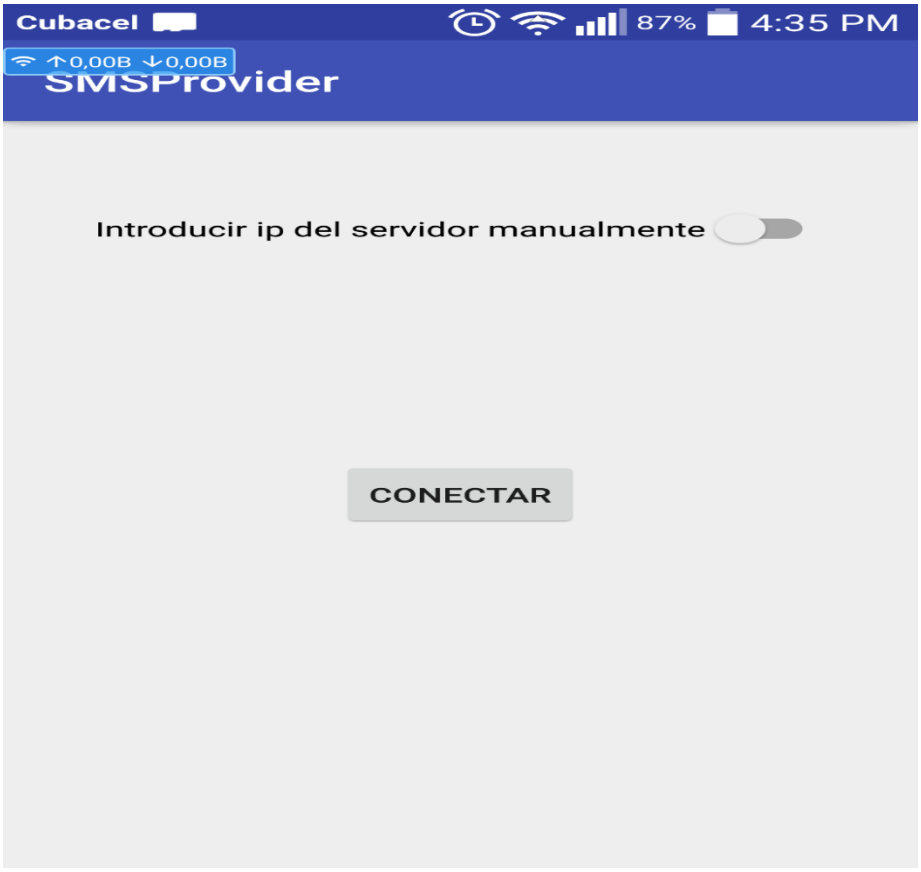

Cubacel
87% 4:35 PM
↑0,00B ↓0,00B
SMSProvider
Introducir ip del servidor manualmente
CONECTAR

Tabla 22. Descripción de la Historia de usuario del RF9

Número: RF9	**Nombre del requisito:** Permitir al usuario especificar directamente el ip del servidor.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir al usuario conectarse manualmente desde el móvil al servidor mediante una dirección IP. **2- Acciones para lograr el objetivo (precondiciones y datos):** - Para buscar los SMS, debe haber algún dispositivo servidor disponible. **3- Comportamientos válidos y no válidos (flujo central y alternos):** El usuario activa la conexión manual, completa el campo con una dirección IP, y presiona la opción conectar. Después recibe la notificación de que está conectado. En caso de que la dirección IP no sea correcta muestra un mensaje de error. En caso de que no encuentre la dirección IP muestra una notificación de que el IP no fue encontrado. **4- Flujo de la acción a realizar:** El usuario selecciona la opción de conexión manual, llena el campo de la dirección IP, y presiona el botón conectar.	
Observaciones:	
Prototipo de interfaz:	

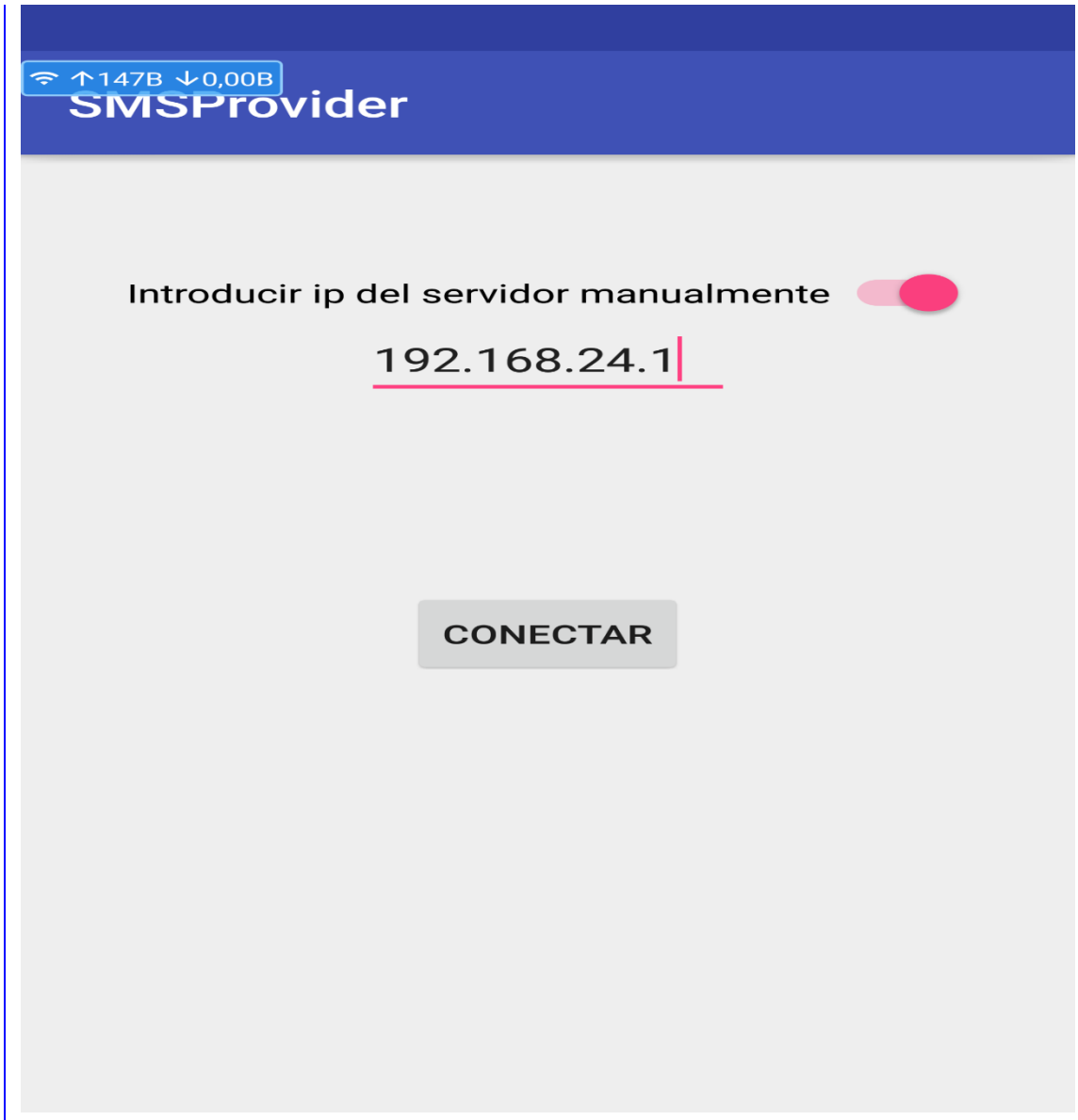

↑147B ↓0,00B
SMSProvider
Introducir ip del servidor manualmente
192.168.24.1
CONECTAR

Tabla 23. Descripción de la Historia de usuario del RF10

Número: RF10	**Nombre del requisito:** Listar dispositivos almacenados y conectados.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Listar los dispositivos cuyos datos han sido previamente almacenados estén o no conectados. **2- Acciones para lograr el objetivo (precondiciones y datos):** - Debe haber algún dispositivo almacenado o conectado. **3- Flujo de la acción a realizar:** Cuando el usuario accede a la vista principal se listan los dispositivos almacenados en la base de datos o que estén conectados.	
Observaciones:	
Prototipo de interfaz:	

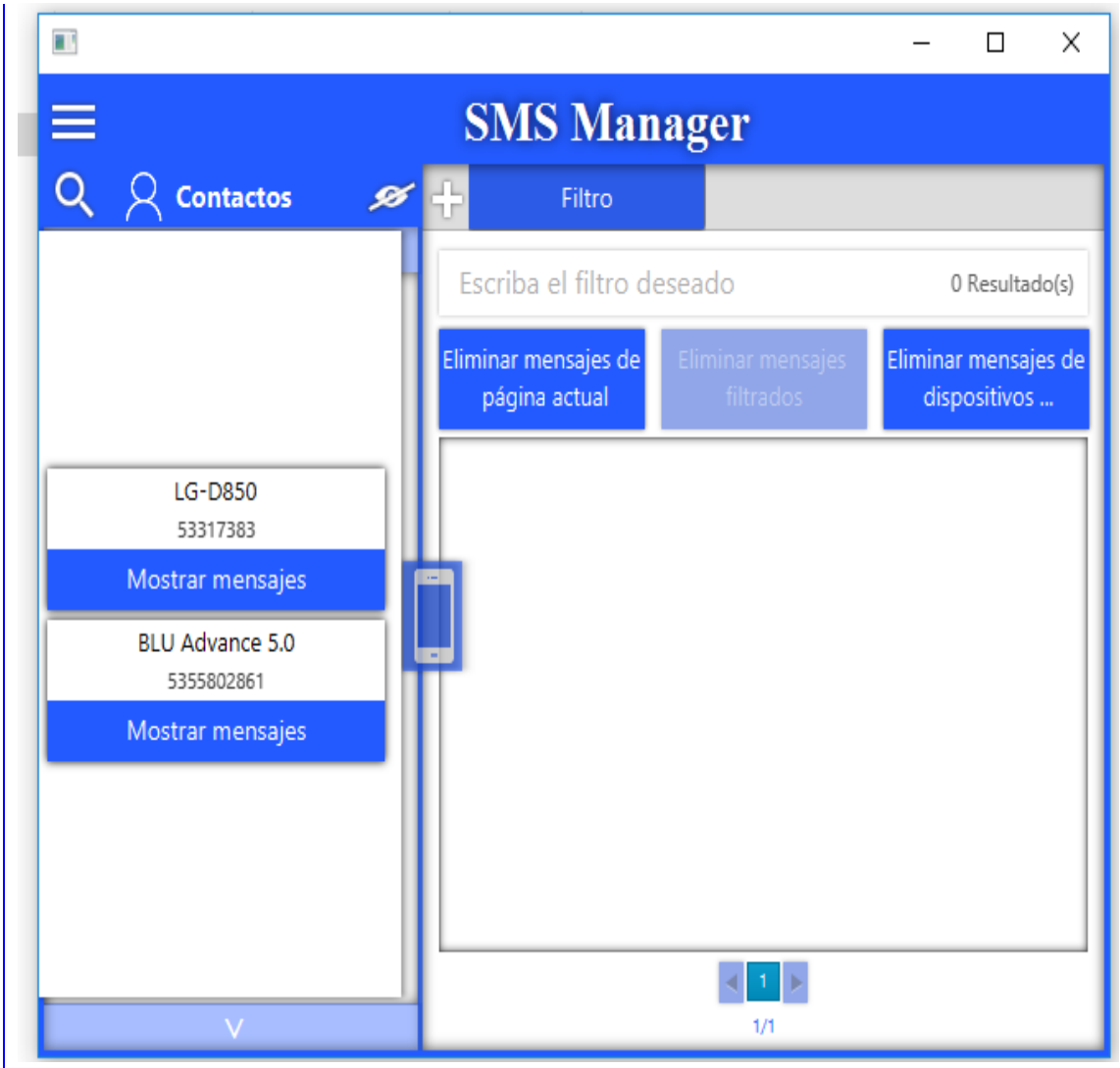

SMS Manager
Contactos
Filtro
Escriba el filtro deseado
0 Resultado(s)
Eliminar mensajes de página actual
Eliminar mensajes filtrados
Eliminar mensajes de dispositivos ...
LG-D850
53317383
Mostrar mensajes
BLU Advance 5.0
5355802861
Mostrar mensajes
1
1/1

Tabla 24. Descripción de la Historia de usuario del RF11

Número: RF11	**Nombre del requisito:** Solicitar autenticación al conectarse un dispositivo.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir que un usuario pueda autenticarse para conectarse a un dispositivo. **2- Acciones para lograr el objetivo (precondiciones y datos):** - Debe haber algún dispositivo móvil conectado. **3- Comportamientos válidos y no válidos (flujo central y alternos):** EL usuario selecciona el dispositivo móvil, si el dispositivo móvil está registrado se pide autenticación por un campo de contraseña. Si la contraseña es correcta se procede a la carga de los mensajes contenidos en ese dispositivo. En caso de que la contraseña sea incorrecta se muestra un mensaje indicando el error. Si el dispositivo móvil no se ha conectado anteriormente se le solicita al usuario que ingrese una nueva contraseña, y la confirme en dos campos. En caso de que las contraseñas no coinciden se muestra un mensaje de error. **4- Flujo de la acción a realizar:** El usuario selecciona el dispositivo, se autentica o registra según corresponda, y posteriormente se hace la carga con los mensajes correspondientes	
Observaciones:	
Prototipo de interfaz:	


Autenticar
Introduzca la contraseña para ver los sms del dispositivo
Usuario: 53643040
Aceptar
Cancelar

Tabla 25. Descripción de la Historia de usuario del RF12

Número: RF12	**Nombre del requisito:** Solicitar autenticación al querer mostrar los mensajes de un dispositivo almacenado.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir la autenticación de un usuario mediante una solicitud. **2- Acciones para lograr el objetivo (precondiciones y datos):** - Debe haber algún dispositivo móvil conectado. **3- Comportamientos válidos y no válidos (flujo central y alternos):** EL usuario selecciona el dispositivo móvil, si el dispositivo móvil está registrado se pide autenticación por un campo de contraseña. Si la contraseña es correcta se procede a la carga de los mensajes contenidos en ese dispositivo. En caso de que la contraseña sea incorrecta se muestra un mensaje indicando el error. Si el dispositivo móvil no se ha conectado anteriormente se le solicita al usuario que ingrese una nueva contraseña, y la confirme en dos campos. En caso de que las contraseñas no coinciden se muestra un mensaje de error. **4- Flujo de la acción a realizar:** El usuario selecciona el dispositivo, se autentica o registra según corresponda, y posteriormente se hace la carga con los mensajes correspondientes.	
Observaciones:	
Prototipo de interfaz:	

Tabla 26. Descripción de la Historia de usuario del RF13

Número: RF13	**Nombre del requisito:** Mostrar mensajes de los dispositivos autenticados.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir visualizar los SMS de los dispositivos móviles conectados. **2- Acciones para lograr el objetivo (precondiciones y datos):** - Debe haber un dispositivo móvil conectado y además autenticado. **3- Flujo de la acción a realizar:** Cuando el sistema muestra los datos de los dispositivos móviles conectados o previamente conectados. Se pide para cualquier dispositivo que el usuario debe estar autenticado.	
Observaciones:	

Prototipo de interfaz:

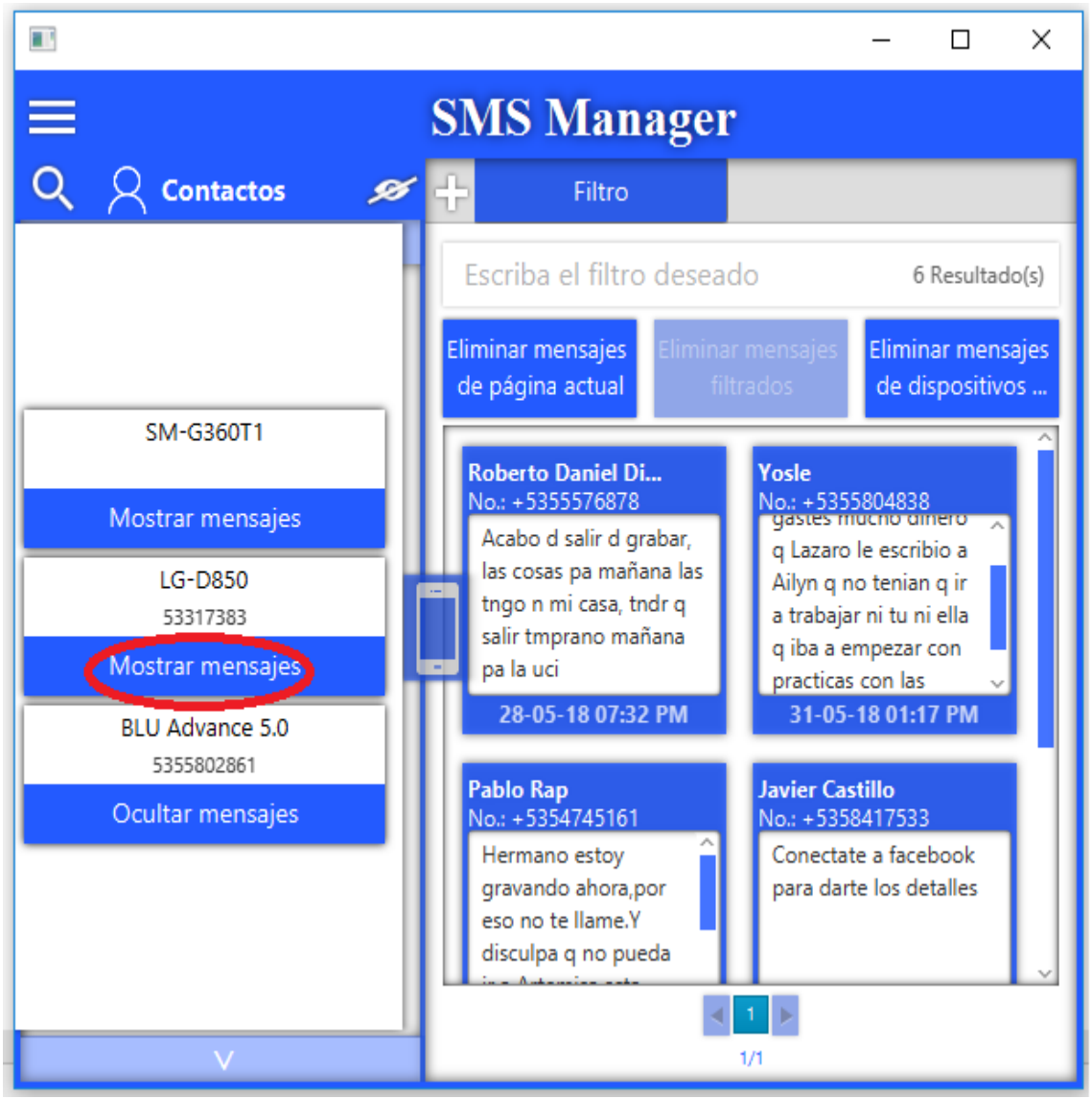


Tabla 27. Descripción de la Historia de usuario del RF14

Número: RF14	**Nombre del requisito:** Ocultar mensajes propios al desconectar un dispositivo
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Ocultar los mensajes propios cuando un dispositivo se desconecta. **2- Acciones para lograr el objetivo (precondiciones y datos):** - Un dispositivo debe estar conectado. **3- Flujo de la acción a realizar:** Una vez que el dispositivo se desconecta, el sistema muestra una ventana pidiendo confirmar si desea seguir viendo los mensajes o no, en caso de que la opción sea sí, debe introducir nuevamente la contraseña, y en caso de que sea no, los mensajes de la base de datos se mantienen ocultos.	
Observaciones:	
Prototipo de interfaz:	

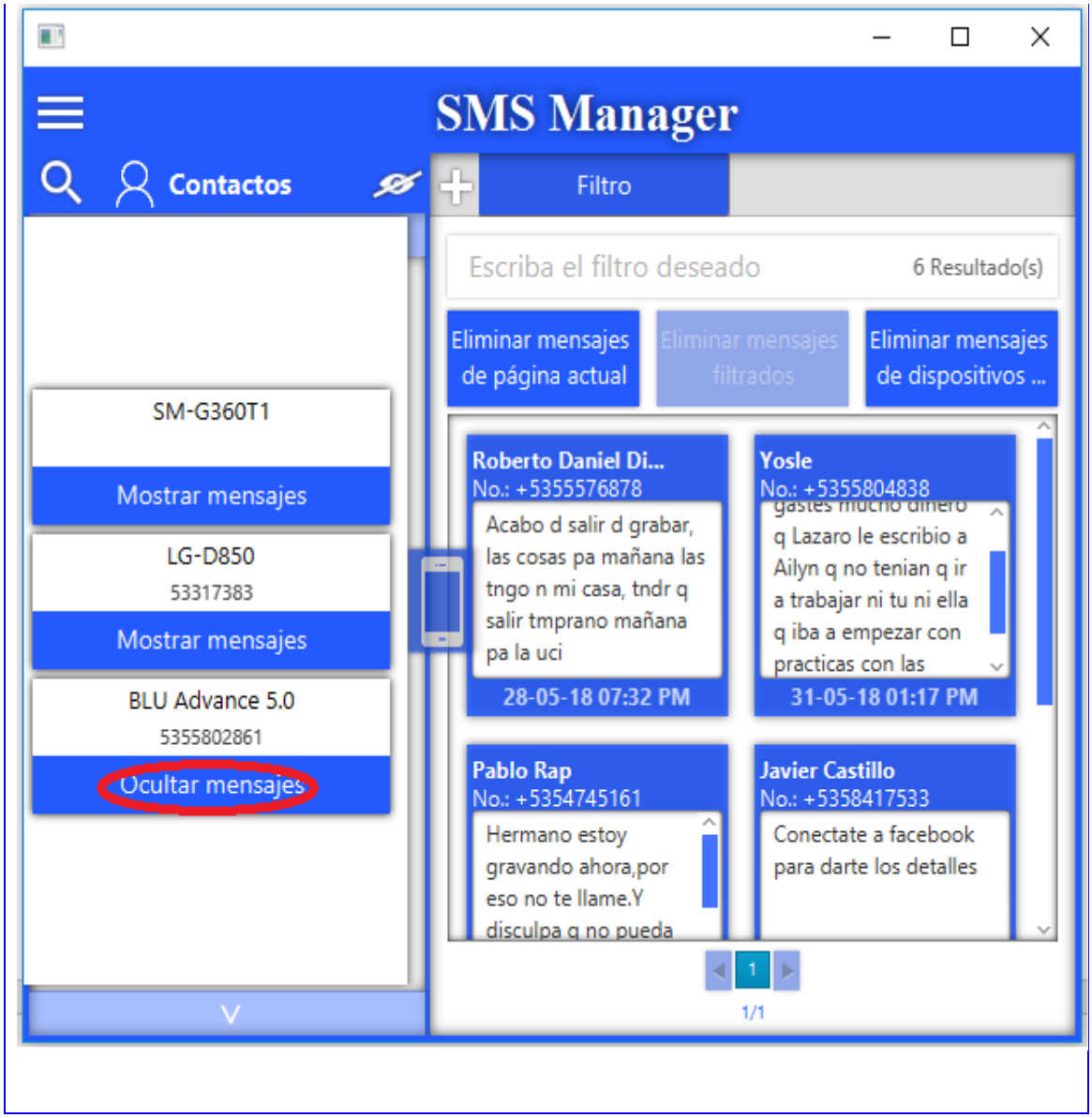

SMS Manager
Contactos
Filtro
Escriba el filtro deseado
6 Resultado(s)
Eliminar mensajes de página actual
Eliminar mensajes filtrados
Eliminar mensajes de dispositivos ...
SM-G360T1
Mostrar mensajes
LG-D850
53317383
Mostrar mensajes
BLU Advance 5.0
5355802861
Ocultar mensajes
Roberto Daniel Di...
No.: +5355576878
Acabo d salir d grabar, las cosas pa mañana las tngo n mi casa, tndr q salir tmprano mañana pa la uci
28-05-18 07:32 PM
Yosle
No.: +5355804838
q Lazaro le escribio a Ailyn q no tenian q ir a trabajar ni tu ni ella q iba a empezar con practicas con las
31-05-18 01:17 PM
Pablo Rap
No.: +5354745161
Hermano estoy gravando ahora,por eso no te llame.Y disculpa q no pueda
Javier Castillo
No.: +5358417533
Conectate a facebook para darte los detalles
1
1/1

Tabla 28. Descripción de la Historia de usuario del RF15

Número: RF15	**Nombre del requisito:** Mostrar contactos de los dispositivos conectados.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Visualizar los contactos de un dispositivo conectado. **2- Acciones para lograr el objetivo (precondiciones y datos):** - Debe haber al menos un dispositivo conectado. **4- Flujo de la acción a realizar:** El usuario selecciona un dispositivo, la aplicación muestra los contactos almacenados en ese dispositivo.	
Observaciones:	
Prototipo de interfaz:	

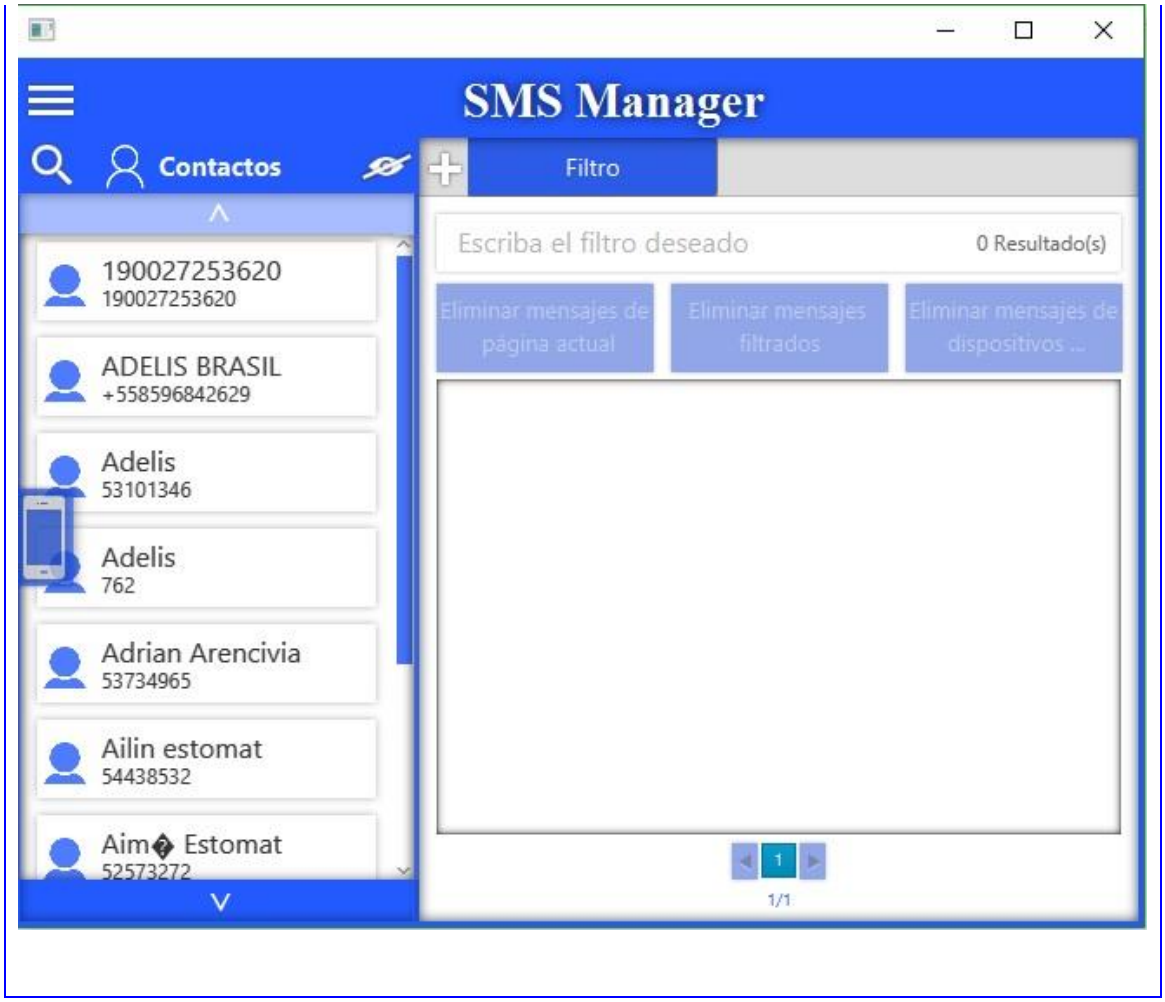

SMS Manager
Contactos
Filtro
Escriba el filtro deseado
0 Resultado(s)
Eliminar mensajes de pagina actual
Eliminar mensajes filtrados
Eliminar mensajes de dispositivos ...
190027253620
190027253620
ADELIS BRASIL
+558596842629
Adelis
53101346
Adelis
762
Adrian Arencivia
53734965
Ailin estomat
54438532
Aim Estomat
52573272
1
1/1

Tabla 29. Descripción de la Historia de usuario del RF16

Número: RF16	**Nombre del requisito:** Mostrar mensajes de un contacto seleccionado.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir buscar los SMS del dispositivo móvil a partir de un criterio determinado.

2- Acciones para lograr el objetivo (precondiciones y datos):

- El dispositivo móvil que posee los contactos debe haber estado conectado en el sistema.

3- Comportamientos válidos y no válidos (flujo central y alternos):

EL campo es obligatorio.

- (*) Filtro: permite todo tipo de caracteres.

4- Flujo de la acción a realizar:

Una vez que el usuario selecciona un contacto. El sistema muestra todos los SMS correspondientes a ese contacto.

Observaciones:

Prototipo de interfaz:


SMS Manager
Contactos
Filtro
Escriba el filtro deseado
6 Resultado(s)
Eliminar mensajes de página actual
Eliminar mensajes filtrados
Eliminar mensajes de dispositivos ...
Roberto Daniel Di...
No.: +5355576878
Acabo d salir d grabar, las cosas pa mañana las tngo n mi casa, tndr q salir tmprano mañana pa la uci
28-05-18 07:32 PM
Yosle
No.: +5355804838
q Lazaro le escribio a Ailyn q no tenian q ir a trabajar ni tu ni ella q iba a empezar con practicas con las
31-05-18 01:17 PM
Pablo Rap
No.: +5354745161
Hermano estoy gravando ahora,por eso no te llame.Y disculpa q no pueda
Javier Castillo
No.: +5358417533
Conectate a facebook para darte los detalles
1
1/1

Tabla 30. Descripción de la Historia de usuario del RF17

Número: RF17	**Nombre del requisito:** Buscar mensajes que coincidan con un criterio de búsqueda.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Permitir buscar los SMS del dispositivo móvil a partir de un criterio determinado.

2- Acciones para lograr el objetivo (precondiciones y datos):

- Para buscar los SMS, debe haber algún dispositivo móvil conectado, y además el usuario debe haber creado algún filtro previamente.

3- Comportamientos válidos y no válidos (flujo central y alternos):

EL campo es obligatorio.

- (*) Filtro: permite todo tipo de caracteres.

4- Flujo de la acción a realizar:

Una vez que el usuario introduzca los datos en el campo filtro, el sistema muestra los SMS que coincidan con los datos introducidos hasta el momento.

Observaciones:

Prototipo de interfaz:

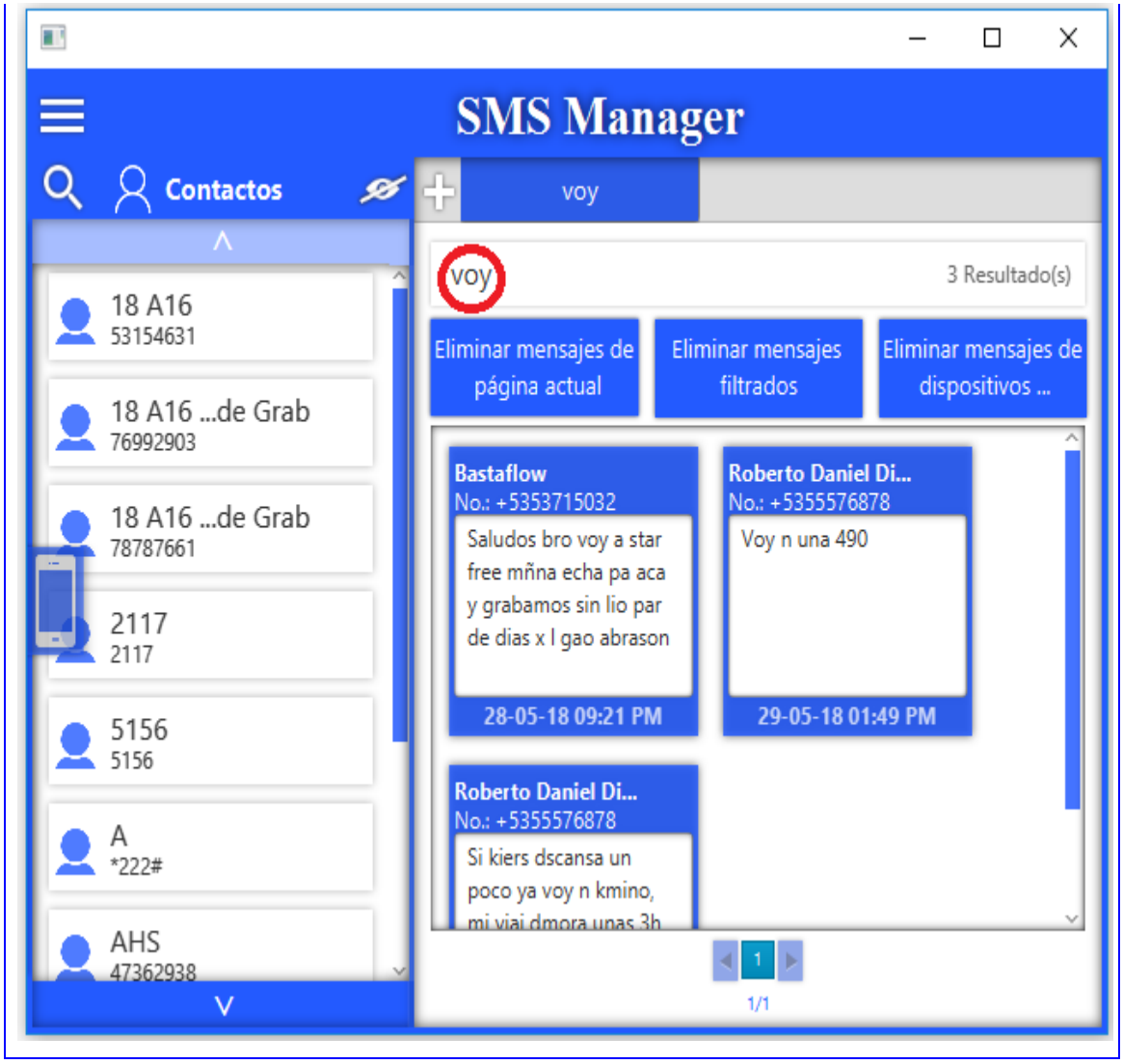

SMS Manager
Contactos
18 A16
53154631
18 A16 ...de Grab
76992903
18 A16 ...de Grab
78787661
2117
2117
5156
5156
A
*222#
AHS
47362938
voy
voy
3 Resultado(s)
Eliminar mensajes de página actual
Eliminar mensajes filtrados
Eliminar mensajes de dispositivos ...
Bastaflow
No.: +5353715032
Saludos bro voy a star free mñna echa pa aca y grabamos sin lio par de dias x l gao abrason
28-05-18 09:21 PM
Roberto Daniel Di...
No.: +5355576878
Voy n una 490
29-05-18 01:49 PM
Roberto Daniel Di...
No.: +5355576878
Si kiers dscansa un poco ya voy n kmino,
1
1/1

Tabla 31. Descripción de la Historia de usuario del RF18

Número: RF18	**Nombre del requisito:** Permitir tener activas varias búsquedas a la vez.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir mantener activas varias búsquedas por diferentes criterios al mismo tiempo. **4- Flujo de la acción a realizar:** Una vez que el usuario comience a introducir los datos, el sistema muestra los SMS que contengan el filtro tecleado hasta el momento. El usuario presiona la tecla más en la parte superior izquierda, la cual adiciona una nueva pestaña que permite introducir un nuevo filtro. Esto no afecta la búsqueda realizada anteriormente.	
Observaciones:	
Prototipo de interfaz:	

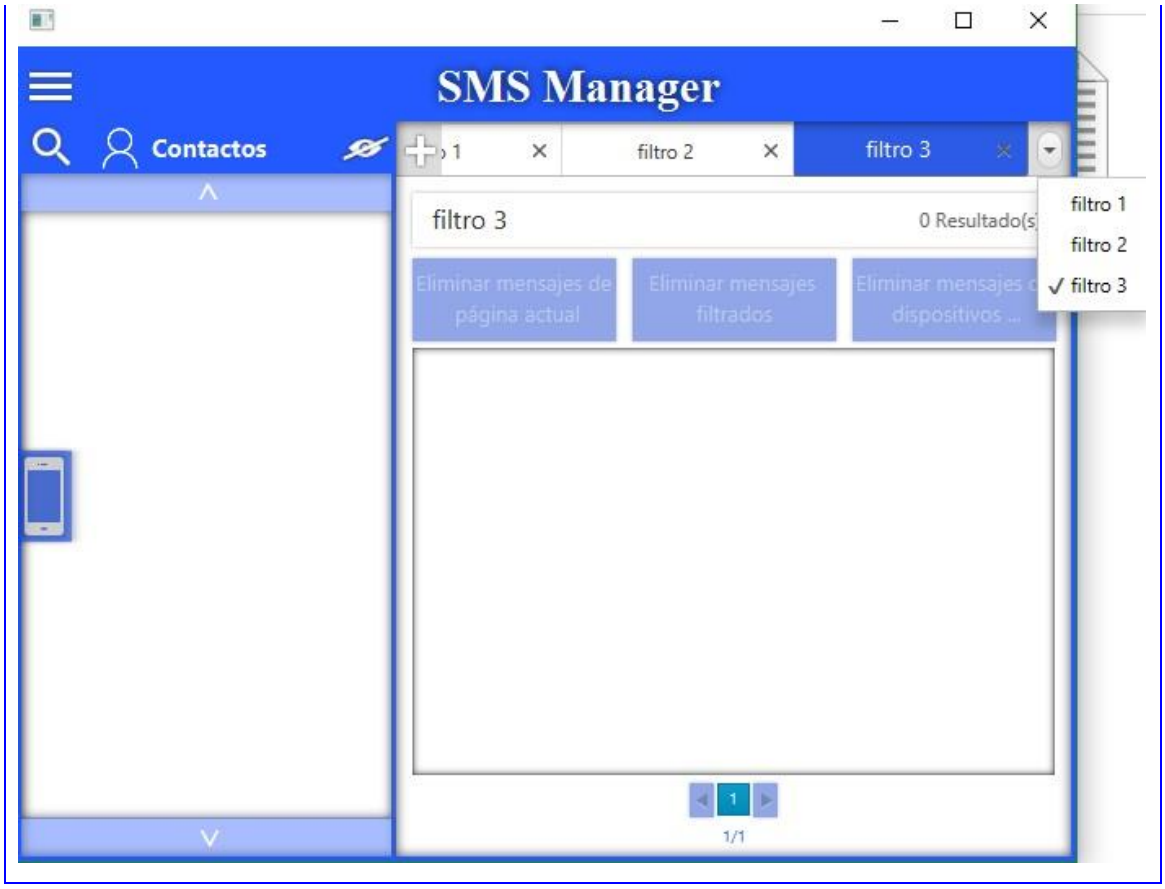

SMS Manager
Contactos
filtro 2
filtro 3
filtro 3
0 Resultado(s)
filtro 1
filtro 2
filtro 3
Eliminar mensajes de página actual
Eliminar mensajes filtrados
1/1

Tabla 32. Descripción de la Historia de usuario del RF19

Número: RF19	**Nombre del requisito:** Eliminar mensaje de la base de datos.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir al usuario eliminar directamente un mensaje de la base de datos. **2- Acciones para lograr el objetivo (precondiciones y datos):** - Para buscar los SMS, debe haber algún dispositivo móvil conectado, y además el usuario debe haber almacenado algún SMS. **3- Flujo de la acción a realizar:** El usuario selecciona el SMS y presiona Eliminar. Eso hace que el SMS sea borrado de la base de datos de forma permanente.	
Observaciones:	
Prototipo de interfaz:	

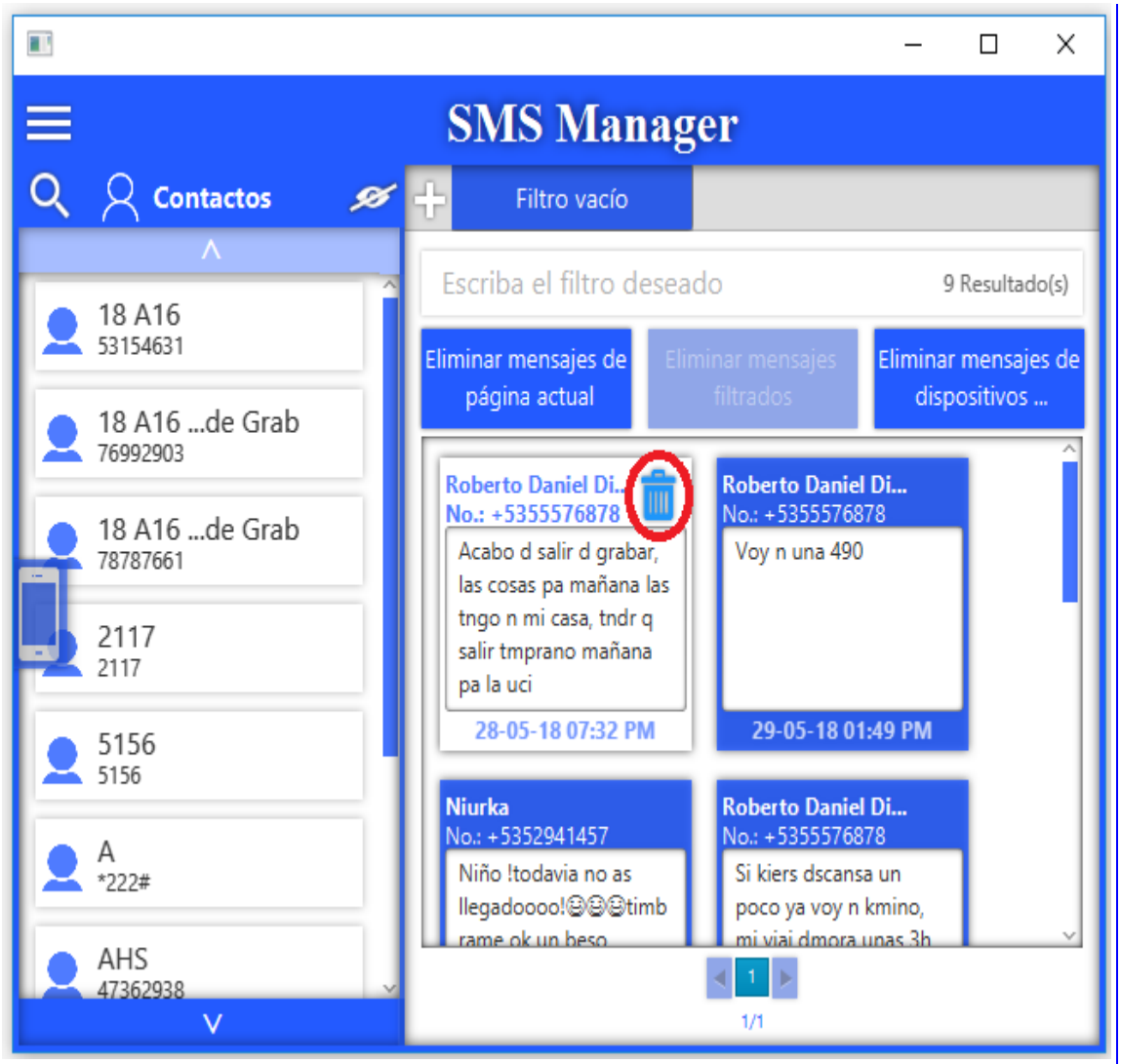

SMS Manager
Contactos
Filtro vacío
Escriba el filtro deseado
9 Resultado(s)
Eliminar mensajes de página actual
Eliminar mensajes filtrados
Eliminar mensajes de dispositivos ...
18 A16
53154631
18 A16 ...de Grab
76992903
18 A16 ...de Grab
78787661
2117
2117
5156
5156
A
*222#
AHS
47362938
Roberto Daniel Di...
No.: +5355576878
Acabo d salir d grabar, las cosas pa mañana las tngo n mi casa, tndr q salir tmprano mañana pa la uci
28-05-18 07:32 PM
Roberto Daniel Di...
No.: +5355576878
Voy n una 490
29-05-18 01:49 PM
Niurka
No.: +5352941457
Niño !todavia no as llegadoooo!timb
Roberto Daniel Di...
No.: +5355576878
Si kiers dscansa un poco ya voy n kmino,
1
1/1

Tabla 33. Descripción de la Historia de usuario del RF20

Número: RF20	**Nombre del requisito:** Eliminar los mensajes que coincidan con el criterio de búsqueda. .
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir al usuario eliminar aquellos mensajes que coincidan con un criterio de búsqueda determinado. **2- Acciones para lograr el objetivo (precondiciones y datos):** - El usuario introduce un criterio de búsqueda, posteriormente selecciona la opción Eliminar mensajes filtrados, para borrar los SMS que coincidan con ese criterio. **3- Flujo de la acción a realizar:** Una vez que el usuario introduce el criterio de búsqueda, el sistema muestra los SMS.	
Observaciones:	
Prototipo de interfaz:	

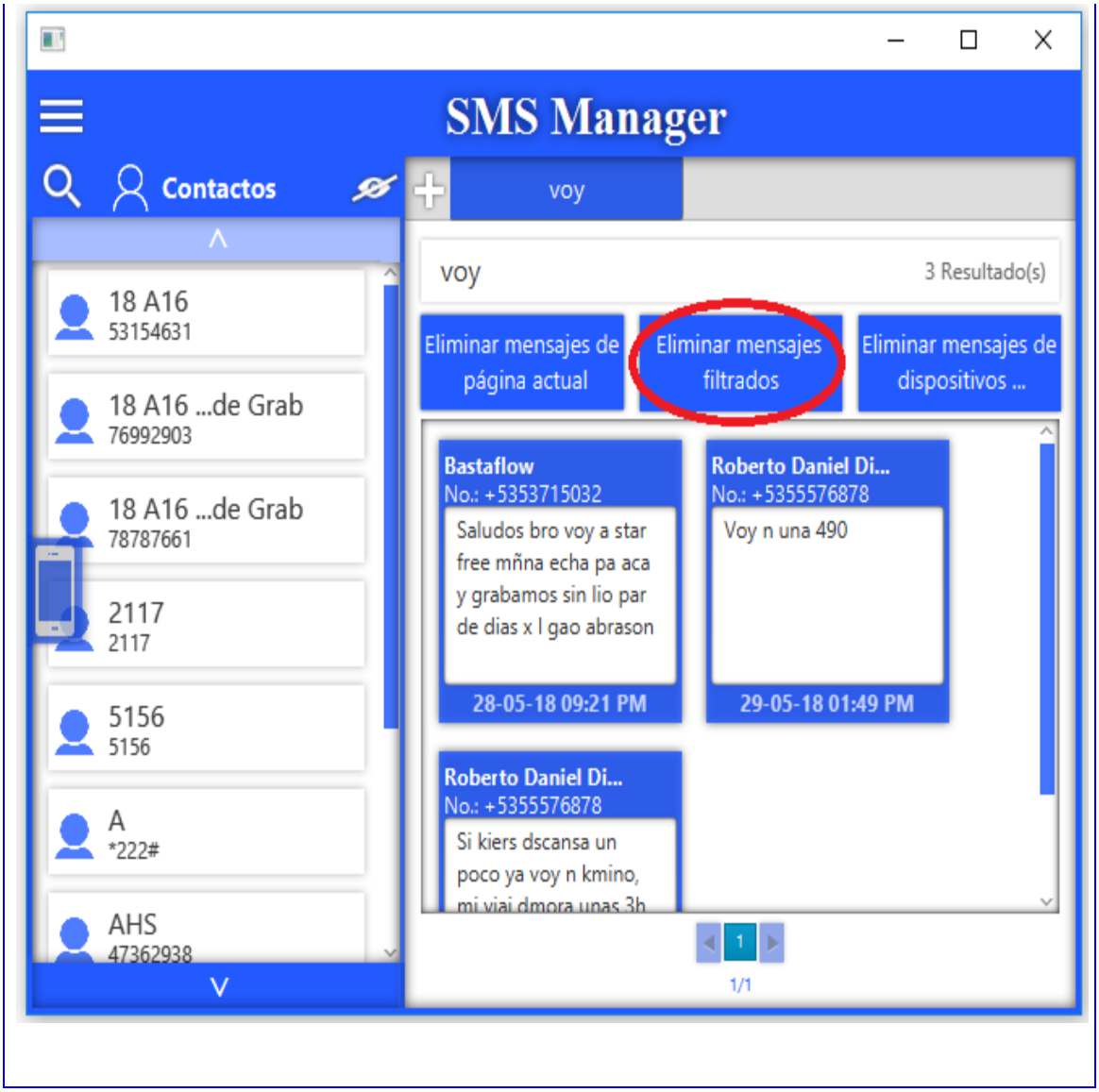

SMS Manager
Contactos
18 A16
53154631
18 A16 ...de Grab
76992903
18 A16 ...de Grab
78787661
2117
2117
5156
5156
A
*222#
AHS
47362938
voy
voy
3 Resultado(s)
Eliminar mensajes de página actual
Eliminar mensajes filtrados
Eliminar mensajes de dispositivos ...
Bastaflow
No.: +5353715032
Saludos bro voy a star free mñna echa pa aca y grabamos sin lio par de dias x l gao abrason
28-05-18 09:21 PM
Roberto Daniel Di...
No.: +5355576878
Voy n una 490
29-05-18 01:49 PM
Roberto Daniel Di...
No.: +5355576878
Si kiers dscansa un poco ya voy n kmino,
1
1/1

Tabla 34. Descripción de la Historia de usuario del RF21

Número: RF21	**Nombre del requisito:** Actualizar interfaz al recibir nuevos mensajes.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir que la interfaz se actualice de forma automática cuando el dispositivo recibe un nuevo mensaje. **2- Flujo de la acción a realizar:** Cuando el dispositivo recibe un nuevo SMS automáticamente se actualiza la interfaz mostrando los datos de ese mensaje.	
Observaciones:	
Prototipo de interfaz:	

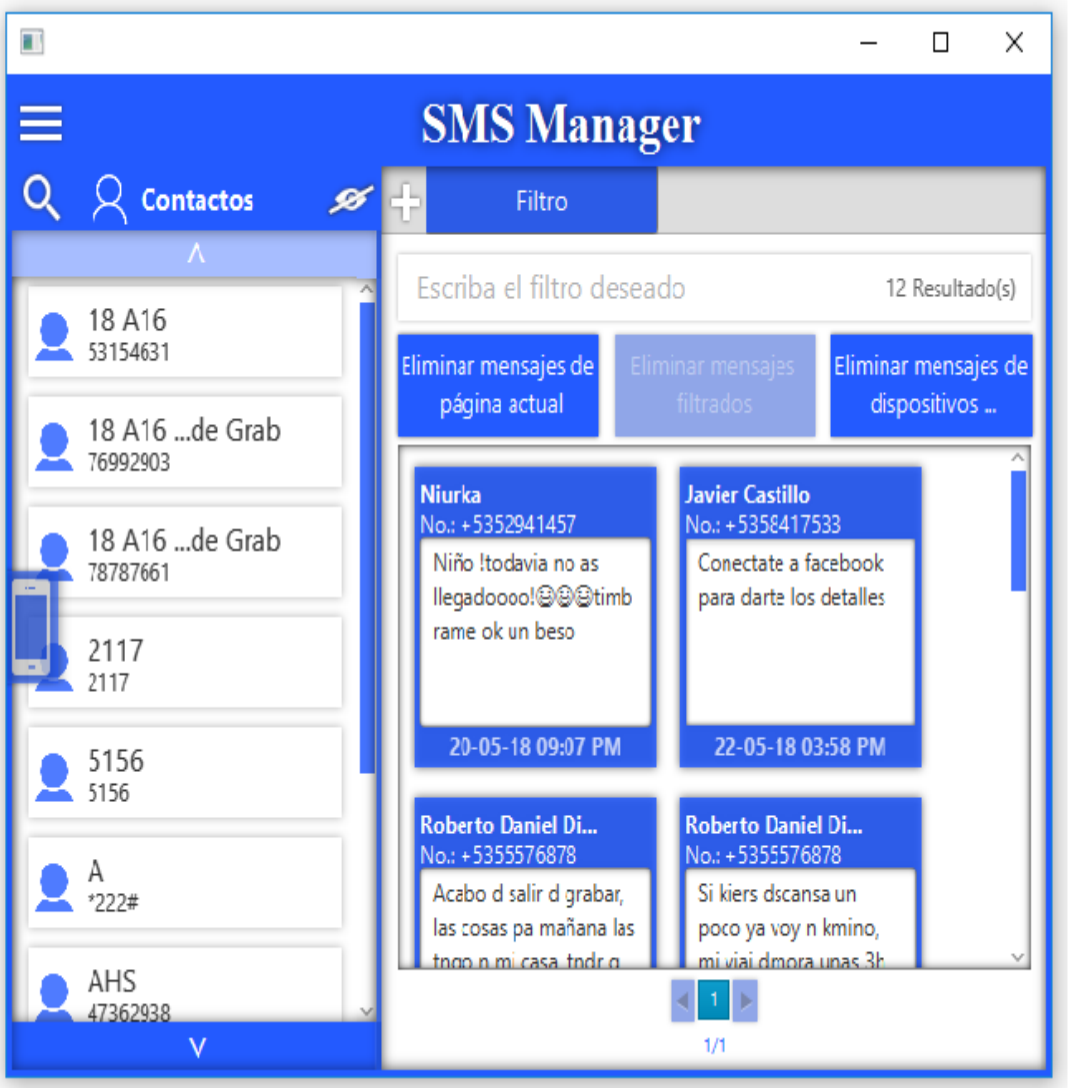

SMS Manager
Contactos
Filtro
Escriba el filtro deseado
12 Resultado(s)
Eliminar mensajes de página actual
Eliminar mensajes filtrados
Eliminar mensajes de dispositivos ...
18 A16
53154631
18 A16 ...de Grab
76992903
18 A16 ...de Grab
78787661
2117
2117
5156
5156
A
*222#
AHS
47362938
Niurka
No.: +5352941457
Niño !todavia no as llegadoooo!timb rame ok un beso
20-05-18 09:07 PM
Javier Castillo
No.: +5358417533
Conectate a facebook para darte los detalles
22-05-18 03:58 PM
Roberto Daniel Di...
No.: +5355576878
Acabo d salir d grabar, las cosas pa mañana las
Roberto Daniel Di...
No.: +5355576878
Si kiers dscansa un poco ya voy n kmino,
1
1/1

Tabla 35. Descripción de la Historia de usuario del RF22

Número: RF22	**Nombre del requisito:** Mostrar los 5 contactos más activos.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir al usuario identificar los cinco contactos más activos. **2- Flujo de la acción a realizar:** El usuario selecciona en la barra superior izquierda la opción Mostrar los 5 contactos más activos. Se le muestra en pantalla los datos de cada uno de ellos, así como la cantidad de mensajes.	
Observaciones:	
Prototipo de interfaz:	

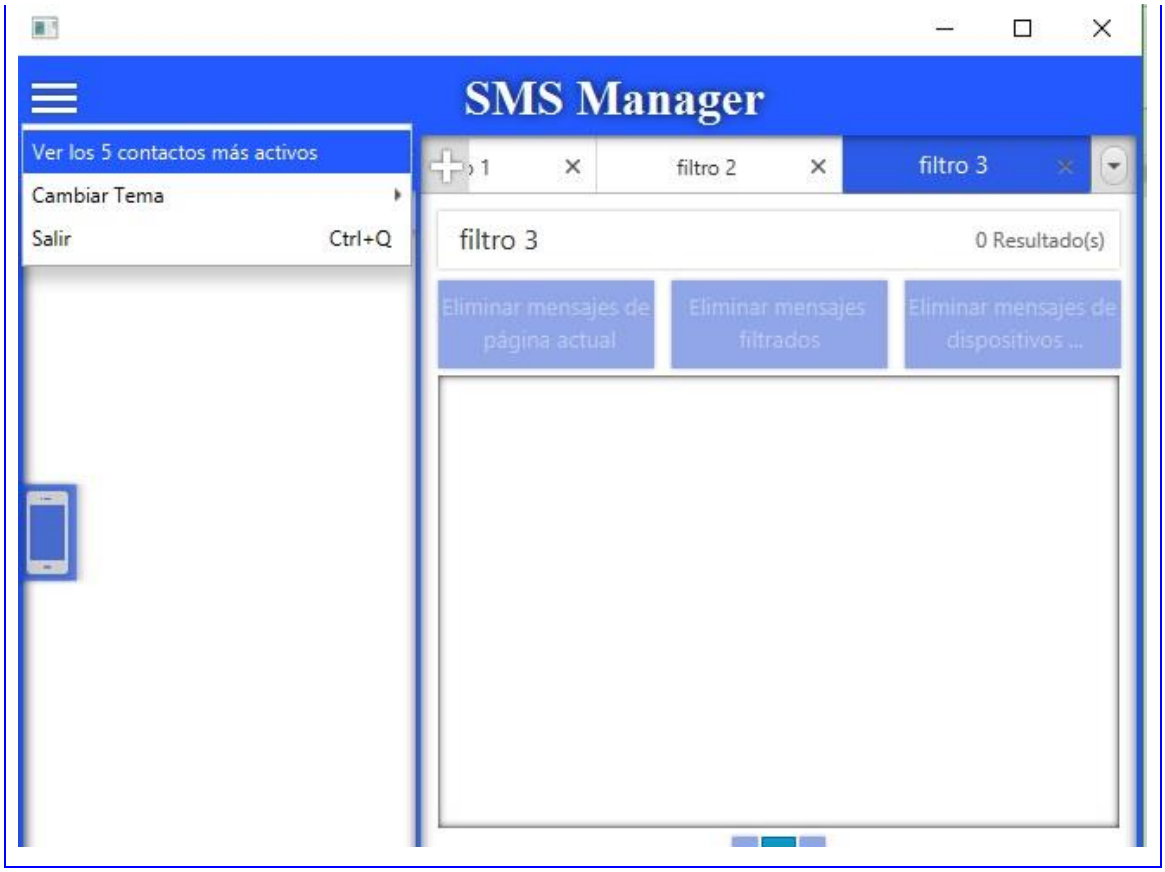
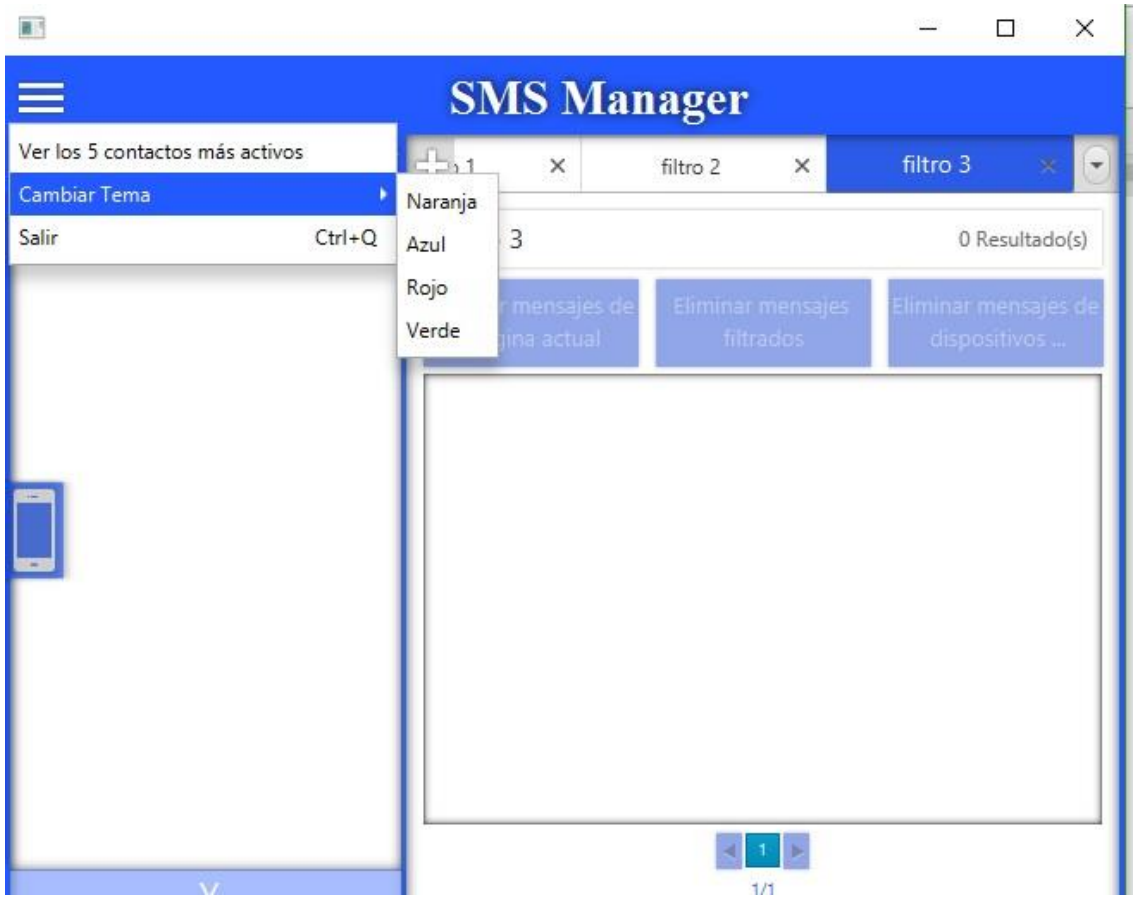

SMS Manager
Ver los 5 contactos más activos
Cambiar Tema
Salir
Ctrl+Q
1
filtro 2
filtro 3
filtro 3
0 Resultado(s)
Eliminar mensajes de página actual
Eliminar mensajes filtrados
Eliminar mensajes de dispositivos ...

Tabla 36. Descripción de la Historia de usuario del RF23

Número: RF23	**Nombre del requisito:** Seleccionar un tema.
Programador: Jorge Antonio Cabrera Cruz	**Iteración Asignada:** 1
Prioridad: Baja	**Tiempo Estimado:** 20h
Riesgo en Desarrollo: No se identificaron riesgos.	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir al usuario un tema de interfaz entre varias opciones. **2- Flujo de la acción a realizar:** El usuario selecciona la opción cambiar tema en la barra superior. Se le despliega una lista con los colores de tema disponibles. El selecciona un tema, y automáticamente es aplicado.	
Observaciones:	
Prototipo de interfaz:	


SMS Manager
Ver los 5 contactos más activos
Cambiar Tema
Salir
Ctrl+Q
Naranja
Azul
Rojo
Verde
filtro 2
filtro 3
0 Resultado(s)
Eliminar mensajes filtrados
Eliminar mensajes de dispositivos ...
1
1/1

Anexo #2: Casos de prueba del método de caja negra

Tabla 37. Diseño de caso de prueba correspondiente al RF10

ID del escenario	Escenario	Respuesta del sistema	Flujo central
EC 1.1 Listar dispositivos almacenados y conectados.	El usuario accede a la vista principal.	El sistema muestra un listado con todos los dispositivos conectados, y almacenados en la base de datos.	Inicio/Principal/Menú lateral/Dispositivos.

Tabla 38. Diseño de caso de prueba correspondiente al RF13

ID del escenario	Escenario	Respuesta del sistema	Flujo central
EC 1.1 Mostrar mensajes de los dispositivos.	El usuario selecciona un dispositivo, se autentica y posteriormente se le muestra un listado con los SMS contenidos en el dispositivo.	El sistema muestra un listado con la información de los SMS del dispositivo seleccionado.	Inicio/Principal/Menú lateral/Dispositivos/Listar SMS.

Tabla 39. Diseño de caso de prueba correspondiente al RF15

ID del escenario	Escenario	Respuesta del sistema	Flujo central
EC 1.1 Mostrar contactos del dispositivo conectado.	El usuario selecciona un dispositivo y se autentica.	El sistema muestra un listado en la columna izquierda con la información de los contactos almacenados en ese dispositivo.	Inicio/Principal/Menú lateral/Dispositivos/Contactos

Anexo #3: Diagramas de clases del diseño

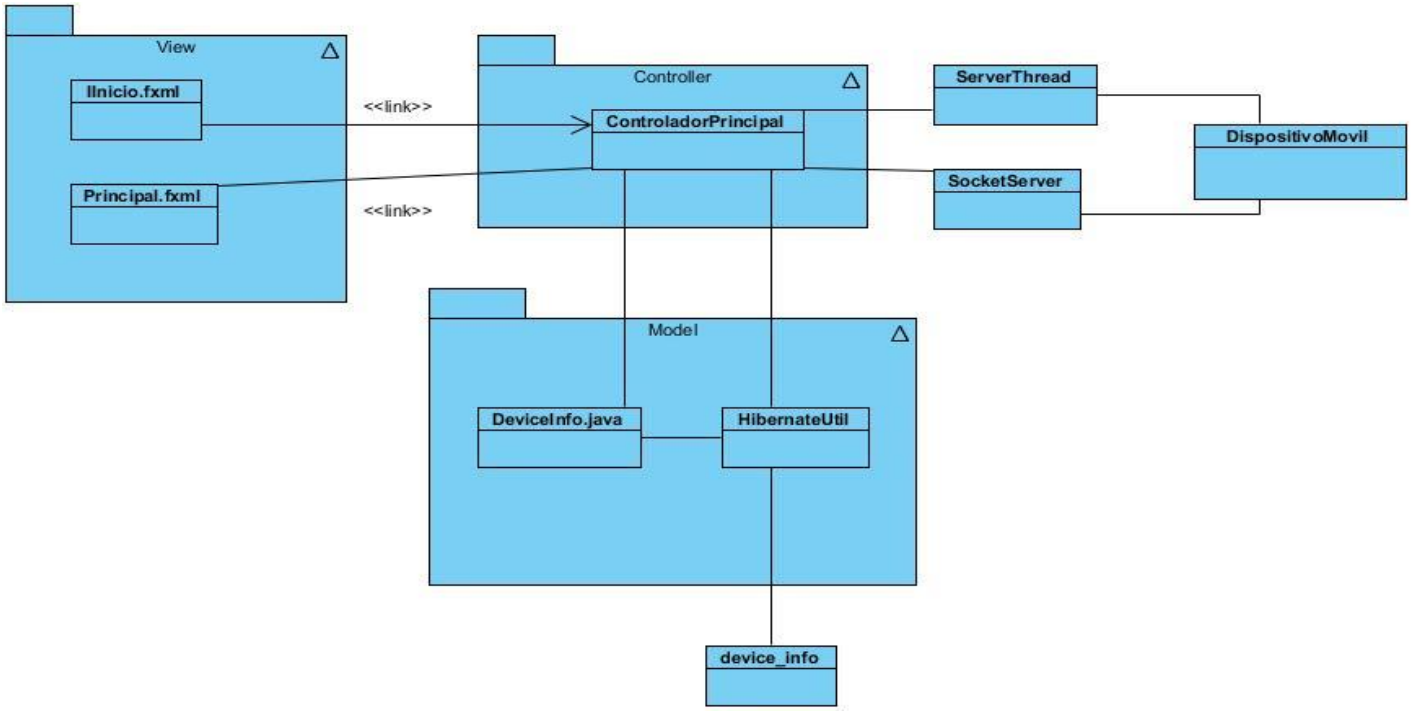


Ilustración 16. Diagrama de clases del diseño del RF1

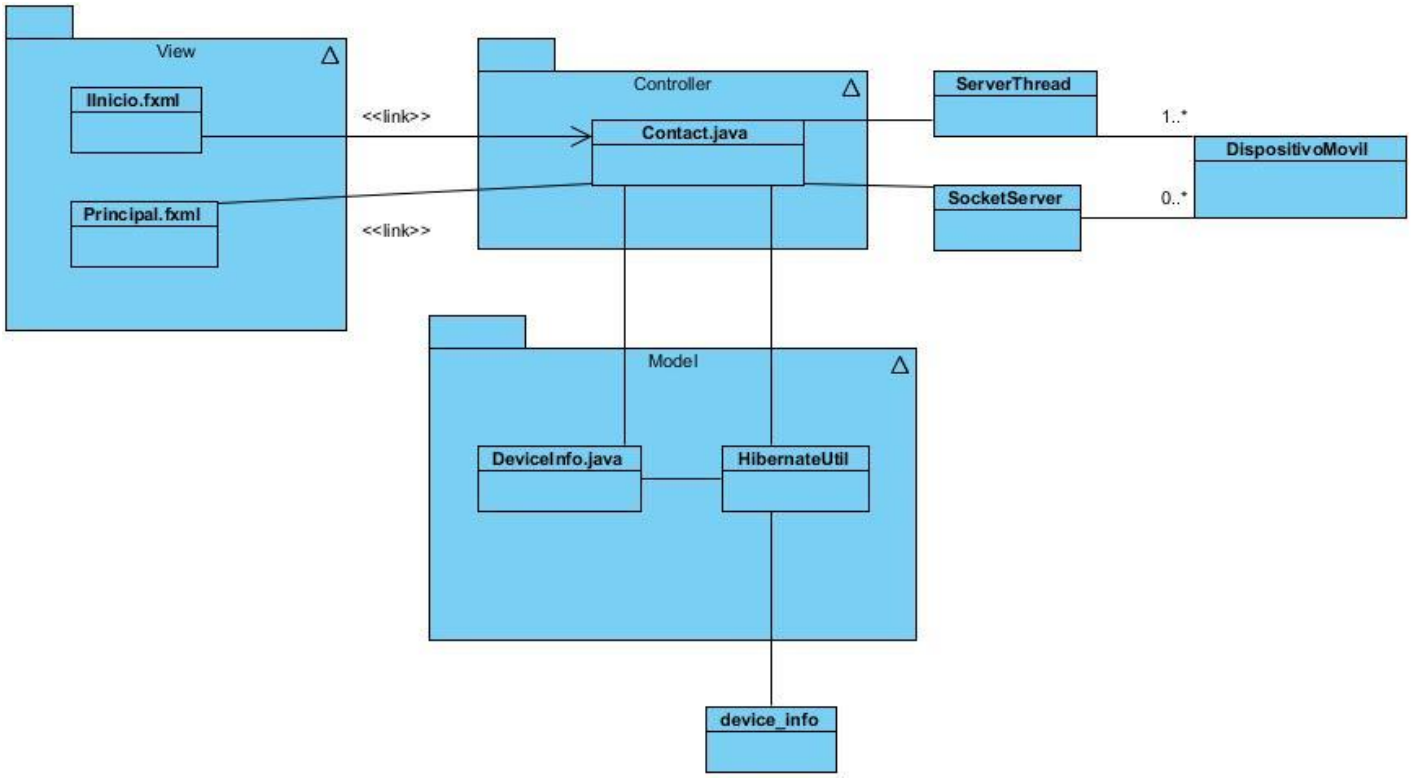


Ilustración 17. Diagrama de clases del diseño del RF2

.

.