Trabajo de Fin de Grado

Aplicación Anti-Ciberacoso

Anti-Cyberbullying Aplication

Anael Eneas Melián Baute

La Laguna, 5 de julio de 2016

D.ª **Pino Caballero Gil**, 45.534.310-Z Catedrática de Universidad Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

D.ª **Jezabel Molina Gil**, con N.I.F. 78.507.682-B Profesora Contratada Laboral Interina de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutora

C E R T I F I C A N

Que la presente memoria titulada:

"*Aplicación Anti-Ciberacoso*"

ha sido realizada bajo su dirección por D. **Anael Eneas Melián Baute**, con N.I.F. 79.061.604-R.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de julio de 2016

Agradecimientos

Me gustaría mostrar mis agradecimientos a mis padres, no solo en la realización de este trabajo, sino todo lo que ha llevado a él, desde el colegio hasta la universidad.

También me gustaría agradecer a mi novia Sonia, a mi hermana Lorelay y a mis amigos Emmanuel e Iñaki por el apoyo constante y sus palabras de ánimo durante la realización de este proyecto.

Por otra parte, me gustaría mucho agradecer tanto a la tutora Pino, como a la cotutora Jezabel, como al resto de integrantes del grupo CryptULL, por todo el apoyo, la ayuda y la involucración en el desarrollo de este TFG.

Por último, que no menos importantes, mis agradecimientos para aquellas personas que contribuyeron de alguna forma en la liberación de estrés durante la realización de este proyecto.

Licencia

Resumen

El objetivo de este trabajo ha sido crear una aplicación para dispositivos móviles con sistema operativo Android, que permite detectar casos de ciberacoso, o ciberbullying según terminología anglosajona. El funcionamiento principal de esta aplicación se basa en recoger y analizar el texto de las notificaciones que llegan al dispositivo, emitidas por aplicaciones de mensajería instantánea, tales como Telegram, WhatsApp, etc. Este texto pasa por un filtro definido internamente en la aplicación, enfocado a detectar posibles casos de ciberacoso. Una vez detectada la posibilidad de estar sufriendo ciberacoso, se envía inmediatamente un aviso de que el usuario del móvil (normalmente un menor) puede estar siendo víctima de ciberbullying, a la persona indicada en la aplicación (padre, tutor, profesor, etc.). La aplicación está pensada para ser instalada de manera que no sea posible detectar su existencia con objeto de proteger, de este modo, a la posible víctima. Por lo tanto, todo su funcionamiento se realiza completamente en segundo plano.

Palabras clave: Ciberacoso, Bullying, Notificación, Aplicación, Android.

Abstract

The aim of this work has been to create an application for mobile devices with Android operating system, which allows detecting cyberbullying cases. The main function of this application is based on collecting and analyzing the text of the notifications that reach the device, issued by instant messaging applications like WhatsApp, Telegram, etc. This text passes through a filter internally defined in the application, which is focused on detecting possible cases of cyberbullying. Once the possibility of suffering cyberbullying is detected, a message to warn that the mobile user (usually a child) might be being a victim of cyberbullying is immediately sent to the person indicated in the application (parent, guardian, teacher, etc.). The application is designed to be installed so that it is not possible to detect its existence, in order to protect, in this way, the potential victim. Therefore, all its operation is performed completely in the background.

Keywords: *Cyberbullying, Bullying, Notification, Application, Android.*

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción al trabajo

En este trabajo se ha decidido utilizar el género masculino del lenguaje, en general, por generalización y economía lingüística.

1.1. Motivación

En está sección se expondrán los motivos que han propiciado la realización de este trabajo, basados en estadísticas y artículos con gran fiabilidad. Estos irán acompañados de una referencia por si es de interés para alguien, que dicha persona pueda acceder a los artículos completos.

- Según las estadísticas, en el caso de España, un 37 % de los jóvenes ha sufrido ciberacoso. De ese porcentaje, un 17 % admite recibir un trato poco amistoso, un 13 % ser objeto de burlas y un 19 % ser insultado. Además, el 63 % de los niños encuestados aseguró saber mucho o algo del ciberacoso y el 81 % reconoció estar preocupado al respecto [1].

- Por otra parte, según la comunidad escolar, el acoso escolar se multiplica por cuatro en niños de 7 u 8 años y disminuye progresivamente hasta el bachillerato.

- Los expertos en ciberbullying coinciden en que el acoso en las redes sociales ya no lo protagonizan sólo los adolescentes, sino también niños de la tierna Primaria. Además, Según el Instituto Nacional de Estadística (INE), la edad media en la que los críos comienzan a navegar por Internet es «por debajo de los 10 años» [2].

- España es uno de los países donde más ciberacoso sufren los menores, en especial los adolescentes de 13 años, según un informe de la Organización Mundial de la Salud (OMS). Entrado en mayor profundidad, entre los chicos, el momento de máxima incidencia son los 11 años y en las chicas entre los 11 y los 13 años, y en ambos casos decae a los 15 años [3].

- España se sitúa en séptima posición en el ránking de países donde los niños de 13 años han recibido amenazas o insultos, por ejemplo a través de servicios de mensajería móvil como Whatsapp o de las redes sociales como Facebook o Twitter, al menos dos o tres veces al mes, por detrás de países como Lituania, Rusia o Bulgaria [3].

Teniendo en cuenta estos datos y estadísticas, la lógica nos invita a pensar que la mejor forma de erradicar el ciberacoso es detectando a tiempo a los acosadores. así, se podrá tomar medidas para evitar que estos vuelvan a acosar en un futuro. En consiguiente, como cada vez nos invade más la tecnología y se consolida la era de los smartphones, la idea de este trabajo ha sido crear una aplicación móvil adaptada a niños que puedan empezar a sumergirse en el mundo del ciberacoso, también conocido por el correspondiente término anglosajón ciberbullying. En la **figura 1.1** podemos apreciar los tipos de acoso escolar que existen.

Figura 1.1: Tipos de acoso escolar

1.2. Objetivos

El objetivo principal de este proyecto es crear una aplicación, llamada BulStapp, para niños y/o pre-adolescentes, capaz de detectar casos de ciberacoso sin que el usuario protegido tenga consciencia de esta. No obstante, este objetivo es bastante general, por lo que se ha dividido en objetivos más específicos, que son los siguientes:

- Reducir los casos de ciberacoso, principalmente en España y si esta tuviera éxito, se realizarían los cambios necesarios para adapatar al resto de países afectados por este tipo de violencia.
- Ofrecer una herramienta que haga que los padres, madres y tutores legales estén más seguros con respecto a la seguridad de los más pequeños de la familia.
- Reducir la posible sobreprotección de los padres o tutores legales hacia los niños, preadolescentes y adolescentes, permitiendo que estos puedan desarrollar una vida normal como los niños y no les discriminen por no hacerlo. Este objetivo, se debe a que a veces los adultos, por miedo, controlan excesivamente a los menores a su cargo. Este objetivo se pretende conseguir reduciendo el miedo y la preocupación de los padres y tutores, de que sus pequeños puedan estar sufriendo en silencio un caso de caso propiciado por las nuevas tecnologías.
- Detectar y frenar conductas de acoso, previniendo que estos niños se conviertan en "**acosadores**.[en] potencia y esta conducta sea más difícil de erradicar en otras etapas más evolucionadas de sus vidas.
- Concienciar a los padres, madres y/o tutores con respecto al acoso en cualquiera de sus formas, dado que muchos adultos no contemplan la posibilidad de que sus los niños puedan estar siendo acosados tanto en el colegio u otros lugares donde pueda socializar con otros niños, como en redes sociales, aplicaciones de mensajería instantánea y demás tipos de violencia cibernética.
- Proteger a los niños que puedan ser víctimas de ciberacoso para que este no afecte negativamente a sus vidas.
- Actuar por los jóvenes que son víctimas de ciberbullying y no lo reportan por miedo a las consecuencias y a las represalias que puedan tomar contra ellos los propios acosadores. Muchos jóvenes, no reportan estas conductas tanto como si son ellos las víctimas como si no, por miedo a que les adjudiquen adjetivos como "**chivatos**" o "**acusicas**".

1.3. Fases del desarrollo

A continuación, se nombrarán y describirán brevemente las diferentes fases que han compuesto el desarrollo de este trabajo. Estas han sido las siguientes:

- **Estudio del campo de estudio:** En esta fase, lo principal ha sido recopilar información sobre el tema del trabajo, en este caso el ciberacoso. Esta fase, ha influido en gran parte en las motivaciones que se han visto en la **sección 1.1**.

- **Estudio de mercado:** También conocido por estudio del arte, esta fase ha consistido en buscar estudios y artículos y aplicaciones similares que están actualmente en el mercado y sobretodo que estén relacionadas con nuestros objetivos vistos en la sección anterior.

- **Estudio y adaptación al software y los lenguajes a utilizar:** Llegados a este punto, ya se disponía de la información suficiente para trabajar, sin embargo tanto la herramienta utilizada (Android Studio), como los lenguajes de programación usados (Java y XML) eran completamente nuevas para el desarrollador. Por este motivo, fue necesario dedicar tiempo a conocer y familiarizarse con esta herramienta de desarrollo, así como con los lenguajes de programación utilizados por esta.

- **Desarrollar código fuente.** Esta fase, sin duda fue la más larga, ya que ha abarcado la gran parte del tiempo y se ha tardado varios meses en realizarse. Esta, consistía en implementar tanto la parte visual como el funcionamiento interno, no solo de las ideas que ya se tenían previamente, sino de las que iban apareciendo.

- **Testeo de la aplicación:** Este punto, consiste en realizar pruebas de nuestra aplicación con las funcionalidades que tuviera implementadas, en distintos dispositivos con diferentes versiones de Android para comprobar que todo está funcionando correctamente. Esta fase apareció varias veces a lo largo del desarrollo, pues cada vez que se implementaba alguna funcionalidad o alguna mejora había que probar que funciona, junto con lo que ya estaba implementado para asegurar las funcionalidades no habían entrado en conflicto.

- **Supervisión con las tutoras:** Una vez terminadas las dos fases anteriores, debemos contar con la aprobación del tutor y que este certifique que todo funciona correctamente, antes de dar el trabajo por finalizado. En caso de que el tutor encontrase algún error o sugiriese alguna modificación o nueva implementación, habría que volver a realizar las dos fases anteriores y volver a pasar por esta.

- **Elaboración de la memoria:** Cuando se haya superado la fase anterior con éxito, significará que el trabajo ya estará completamente terminado. Por lo que la siguiente fase, en la que nos encontramos actualmente, será redactar un informe final de máximo cincuenta páginas más apéndices, que englobe todo el trabajo realizado en este proyecto desde su inicio hasta la fase actual, explicando desde los detalles más pequeños hasta los más complejos.

- **Exposición del trabajo:** La última fase de este proyecto, realizada una vez que hayan sido superadas todas las demás con éxito, será exponer ante un tribunal, formado por tres personas, todo el trabajo realizado. Esto se realizará en formato de exposición, la cual deberá llevarse a cabo durante quince minutos aproximadamente y esta será evaluada junto con la memoria por el presente tribunal.

1.4. Antecedentes

Actualmente, existen varias aplicaciones en el mercado y estudios que luchan contra el **bullying**. A continuación se procede a listar algunas de estas con una pequeña descripción:

- **Stop It:** Esta aplicación permite informar anónimamente de un caso de **bullying** en el que no tienes que ser necesariamente la víctima, el usuario podría informar de algún caso habiendo sido testigo de este [4]. Además, está disponible para dispositivos móviles con sistema operativo Android o iOS y es gratuita.

- **KnowBullying:** Esta es una aplicación diseñada para padres, que facilita las conversaciones con sus hijos, ofreciendo a estos información sobre cómo empezar dichas conversaciones. Además también provee recordatorios especializados para llevarlas acabo [5]. Esta aplicación se encuentra disponible para Android e iOS y es gratuita.

- **Bully Button:** Bully Button Permite grabar incidentes con la cámara del móvil y enviar estos a algún adulto con un solo click [6]. Se encuentra disponible para terminales con sistema operativo Android o iOS y tiene un coste de $0.99.

- **My Mobile Watchdog:** Esta aplicación permite a los padres supervisar las interacciones de sus hijos con sus móviles sin tener que entrar en él [7]. My Mobile Watchdog está disponible para Android e iOS y tiene un coste de $4.95.

- **Net Nanny:** Permite a los padres filtrar y supervisar las actividades de sus hijos [8]. Está disponible para Apple, Android y Windows y cobra por licencias y suscripciones.

- **Bully Tag:** Es una aplicación que permite a quienes han sido testigos de algún caso de bullying, enviar información anónima a las autoridades [9]. Está disponible para Android e iOS y es gratuita.

- **Anonymous Alerts:** Esta app no solo permite informar sobre los casos de bullying vividos en primera persona, sino que además aporta facilidad para que los testigos mantengan conversaciones anónimas y bidireccionales con las autoridades de los colegios [10]. Está disponible para Android, iOS y el navegador web Chrome y es gratuita.

- **Cyberbullying Blocker:** Aplicación para dispositivos móviles Android, que detecta automáticamente posible contenido nocivo en el texto [11]. Aunque a priori pueda parecer el mismo objetivo de nuestra aplicación, el concepto y la forma de trabajar son diferentes. Esta aplicación aún no puede encontrarse en la tienda de Android.

- **PocketGuardian:** Esta app ayuda a los padres a controlar los smartphones y las redes sociales de sus hijos sin invadir su privacidad. Estos, reciben notificaciones cuando ciberbullying o sexting son detectados [12]. Está disponible para iOS y Android. Por otra parte, PocketGuardian no está disponible en España.

- **ReThink:** Se trata de un producto no intrusivo, innovador, pendiente de patente de software, que detiene el ciberacoso antes de que el daño esté hecho. Cuando un adolescente intenta enviar un mensaje ofensivo en los medios sociales, ReThink utiliza su tecnología de filtrado sensible al contexto para determinar si es o no ofensivo y da al adolescente una segunda oportunidad para reconsiderar su decisión [13]. Esta aplicación se encuentra disponible para iOS y Android.

- **NearParent:** Esta es una aplicación inteligente que permite a las familias crear listas de contactos seguros, zonas de seguridad y establecer una configuración para realizar un seguimiento de las ubicaciones de los demás implicados, tanto los niños o los contactos de seguridad como otros familiares [14]. NearParent está disponible para dispositivos con sistema operativo Android de manera gratuita.

Capítulo 2

Introducción a la aplicación

Como ya se comentó en el capítulo anterior, la finalidad del proyecto es la creación de una aplicación que permita detectar posibles casos de ciberacoso. En este apartado, se procede a comentar el funcionamiento y las características de esta, los permisos de los que precisará para su correcto funcionamiento, como la ocultaremos de cara a los menores y las bases de datos que utiliza.

2.1. Funcionamiento

En esta sección, describiremos el funcionamiento general de la aplicación y se entrará en más detalle en el **capítulo 4**. Este, en la teoría, es sencillo de entender y se puede definir en una serie de pasos definidos a continuación:

1. Recoger todas las notificaciones desde un servicio en segundo plano y extraer el mensaje solo de aquellas provenientes de aplicaciones de mensajería directa como "**WhatsApp**" o "**Telegram**".

2. Una vez extraído el mensaje, este pasa por un filtro interno formado por palabras y frases clave, almacenadas en una base de datos y definidas previamente. Las palabras y frases clave, son completamente desconocidas por los usuarios.

3. Mientras se pasa el filtro, si se detecta alguna coincidencia dentro del mensaje con alguna palabra o frase clave, se guarda completo en una base de datos junto con el emisor y la aplicación por la que ha sido enviado. Esta base de datos, será definida en la **sección 2.5.1**, en la que podremos ver un breve resumen de esta, sus atributos y sus claves.

4. A continuación, se comprueba cuántos mensajes del mismo emisor existen en la base de datos. El valor obtenido, es comparado con un número fijado previamente que supondrá el número límite de mensajes sospechosos permisibles.

5. En este punto, pueden suceder dos cosas dependiendo del resultado obtenido en el punto anterior. Si el valor obtenido es menor que el límite, simplemente se almacena el mensaje en la base de datos. Si por el contrario, el valor es igual que el límite, se envía un mensaje de aviso a los números de teléfono que se encuentren en una base de datos definida más adelante en la **sección 2.5.2**. Estos números de teléfono, son especificados previamente por el usuario, la manera de hacerlo se especificará más adelante, en el **capítulo 4**.

6. Finalmente, si el emisor ha llegado al límite de mensajes permitidos, no se volverá a enviar un aviso a no ser que vuelva a sobrepasarlo. Dicho de otro modo, sería como si el contador de mensajes sospechosos de contener ciberacoso volviera a empezar de cero cada vez que alcanza el límite.

Este solo es el funcionamiento general. Algunos aspectos como los números de teléfono especificados previamente y las bases de datos existentes, se explicarán más adelante en otros apartados.

2.2. Características

En este apartado, se explican brevemente las características principales que tiene la aplicación. Estas características son las siguientes:

- La idea principal de esta aplicación, es que el usuario no sea el propietario del móvil, que en principio será un niño o un preadolescente, quien instale la aplicación; sino alguno de sus tutores legales. Por ello, sus principales características están basadas en la idea de que el portador del dispositivo móvil no debe saber de la existencia de esta aplicación en su terminal.

- Teniendo en cuenta el punto anterior, una de las características más importantes de esta aplicación es que debe estar oculta para el usuario, pero quien la haya instalado debe poder hacerla visible y configurarla con los ajustes que esta permite.

- Tiene que existir una contraseña, que solo deben conocer los tutores legales, para acceder a las opciones.

- Se necesita un **permiso especial** para acceder a las notificaciones, el cual se tiene que asignar manualmente.

- A pesar de tener interfaz gráfica y opciones dentro de la aplicación, su ejecución es totalmente en segundo plano.

- Se inicia el servicio, en el cual se realizará todo el funcionamiento principal, automáticamente al abrir la aplicación o al encender el móvil.

- Posee dos bases de datos modificables, explicadas en este capítulo en la **sección 2.5**, una automáticamente y la otra manualmente por el usuario.
- Se pueden añadir números de teléfono a una de las bases de datos directamente desde la agenda de contactos del dispositivo móvil.

2.3. Permisos

En esta sección se nombran y explican los permisos que deberán proporcionarse al dispositivo móvil para que la aplicación funcione correctamente.

2.3.1. Permisos requeridos

- **Acceso a notificaciones:** Permite a la aplicación acceder a las notificaciones entrantes y a toda su información. Este es un permiso especial, que apareció con la versión de Android 4.4 kitkat, que a diferencia de los otros permisos, debe darse manualmente desde los **ajustes de seguridad** del teléfono móvil. La **figura 2.1** muestra la pantalla donde aparecen las aplicaciones que necesitan este permiso. Para dárselo, debemos tocar el círculo que aparece rodeado en rojo y darle al botón de aceptar al diálogo que se mostrará, como podemos ver en la **figura 4.2**. Finalmente veremos el círculo de la **figura 2.1** en verde, lo que quiere decir que dicha aplicación tiene permiso para acceder a las notificaciones.
- **Redireccionar llamadas salientes:** Permite que la aplicación detecte cuando se realiza una llamada y pueda acceder al número marcado con la opción de interferir en ella, por ejemplo redirigiendo esta a otro número o cancelar la misma.
- **Acceso completo a red:** permite que la aplicación cree sockets de red y utilice protocolos de red personalizados. El navegador y otras aplicaciones proporcionan los medios necesarios para el envío de datos a Internet, por lo que no hace falta utilizar este permiso para eso.
- **Consultar tus contactos:** Permite que la aplicación consulte información sobre contactos almacenados en el teléfono, incluida la frecuencia con la que los has llamado, les has enviado un email o te has puesto en contacto con ellos de otro modo. Este permiso permite guardar los datos de los contactos. Nótese que las **aplicaciones malintencionadas** pueden utilizarlo para compartir datos de contactos de usuario sin su consentimiento.

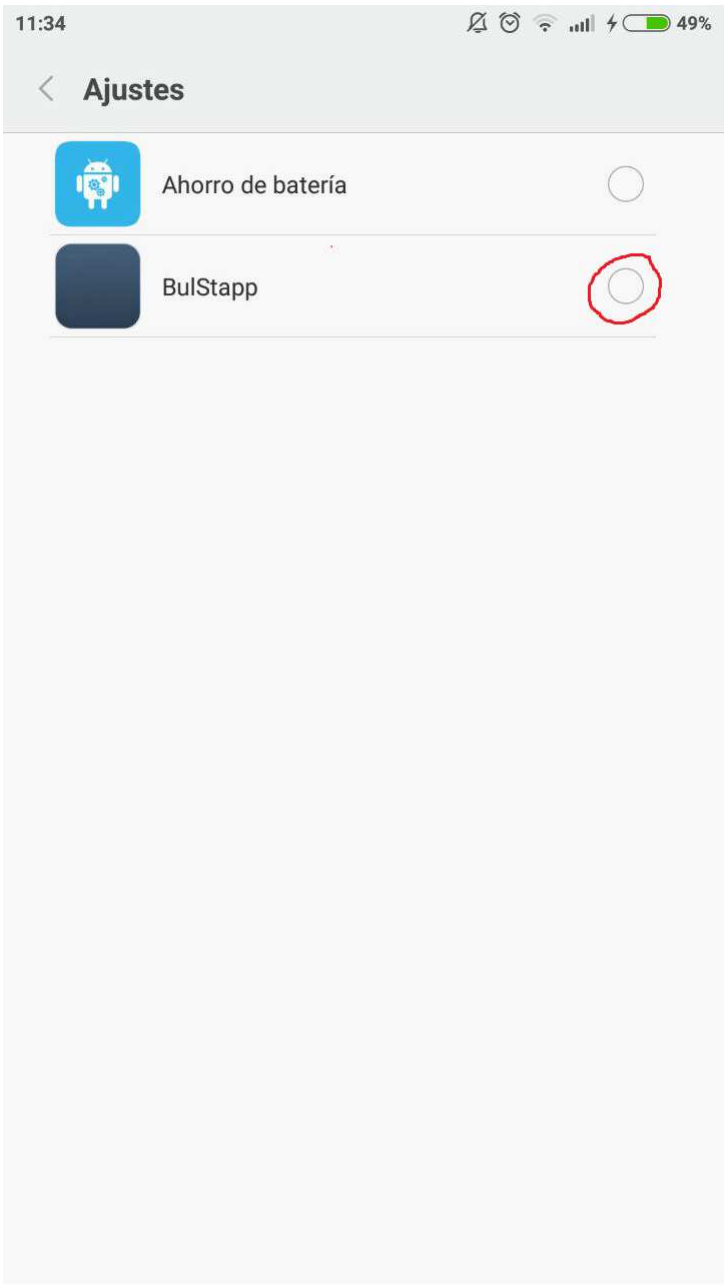

Figura 2.1: Pantalla para permitir permiso de notificaciones

2.3.2. Uso de los permisos

Para concluir esta sección, se procede a explicar para qué acción se necesita cada permiso.

- el **acceso a notificaciones** se necesita para poder extraer el mensaje de las notificaciones entrantes.
- El segundo permiso sirve para hacer visible la aplicación mediante la llamada a un número establecido previamente, lo que se explicará con detalle más adelante.

- El tercer permiso, es requerido para comunicar nuestra aplicación con los teléfonos que aparecen en la base de datos, que veremos en la sección **2.5.2**, cuando se detecte un posible caso de ciberacoso. Este proceso de comunicación se explica con todo detalle en el **capítulo 3**.

- Por último, el permiso de consultar los contactos es necesario para poder añadir contactos a la base de datos de la aplicación. Estos serían los contactos a los que se enviaría el mensaje de aviso en caso de detectar síntomas de ciberacoso.

En la **figura 2.1** aparecía el permiso especial de las notificaciones, pero faltan por ver el resto de permisos. Estos permisos, como se puede ver en la **figura 2.2**, están divididos en categorías: seguridad, privacidad y otros. En este caso, nuestra aplicación necesita dos permisos de seguridad y uno de privacidad. Los permisos de seguridad son el de acceso completo a red y el de redireccionar llamadas recientes y el de privacidad es el de consultar contactos.

Figura 2.2: Permisos básicos que requiere la aplicación

2.4. Ocultar y descubrir la aplicación

Para empezar, recordemos que una de las características descritas en el **apartado 2.2** es que la aplicación debe estar oculta para el usuario. Sin embargo, debe ser visible para quien instala la aplicación.

Por consiguiente, esta sección está dividida en dos subsecciones, la primera explica cómo se oculta la aplicación y qué efectos conlleva, y la segunda explica cómo descubrir o hacer visible la aplicación cuando esta está oculta para el usuario.

2.4.1. Ocultar

Para ocultar nuestra aplicación, lo primero que debemos hacer es ejecutar el lanzador para llegar a la pantalla principal, reflejada en la **figura 4.1**. En esta pantalla principal deberemos pulsar un botón encargado de realizar esta acción, llamado "**ocultar APP**", cuyos detalles podemos ver en la **sección 4.1.3**. Véase también el posicionamiento de este botón en la figura **4.1**.

El efecto que tiene el ocultar la aplicación es que esta no tendrá un lanzador en el menú de aplicaciones ni ningún acceso directo. sin embargo, se podrá ver en el menú de aplicaciones instaladas y se podrá desinstalar y/o borrar sus datos, pero dado que los usuarios son niños, esto no supondrá un problema. También, se sigue ejecutando el servicio en segundo plano, tal y como se refleja en la **figura 2.3**. Esta es una captura de pantalla de un móvil real, Xiaomi MI4C de cinco pulgadas, con la aplicación oculta.

2.4.2. Descubrir

Para **descubrir** la aplicación, lógicamente deberemos haberla ocultado previamente como indica el paso anterior. Cuando esta esté oculta, debemos llamar al número que nos aparece en el mensaje de aviso que aparece en el **diálogo emergente** de la **figura 4.3**, en este caso el **1234**. Cuando realicemos la llamada, nuestra aplicación comprobará si el número es el establecido. Si es así, esta cortará la llamada **automáticamente** y volverá a hacer visible nuestra aplicación.

2.5. Bases de datos

A continuación se concretan algunos detalles relacionados con las bases de datos que se utilizan en este proyecto. Se trata de dos bases de datos distintas, una exclusiva para los mensajes sospechosos y otra para almacenar los contactos a los que se enviará el mensaje de aviso en caso de que algún emisor haya sobrepasado el límite de mensajes sospechosos. En las siguientes secciones, se detalla todo lo relacionado con estas bases de datos por separado.

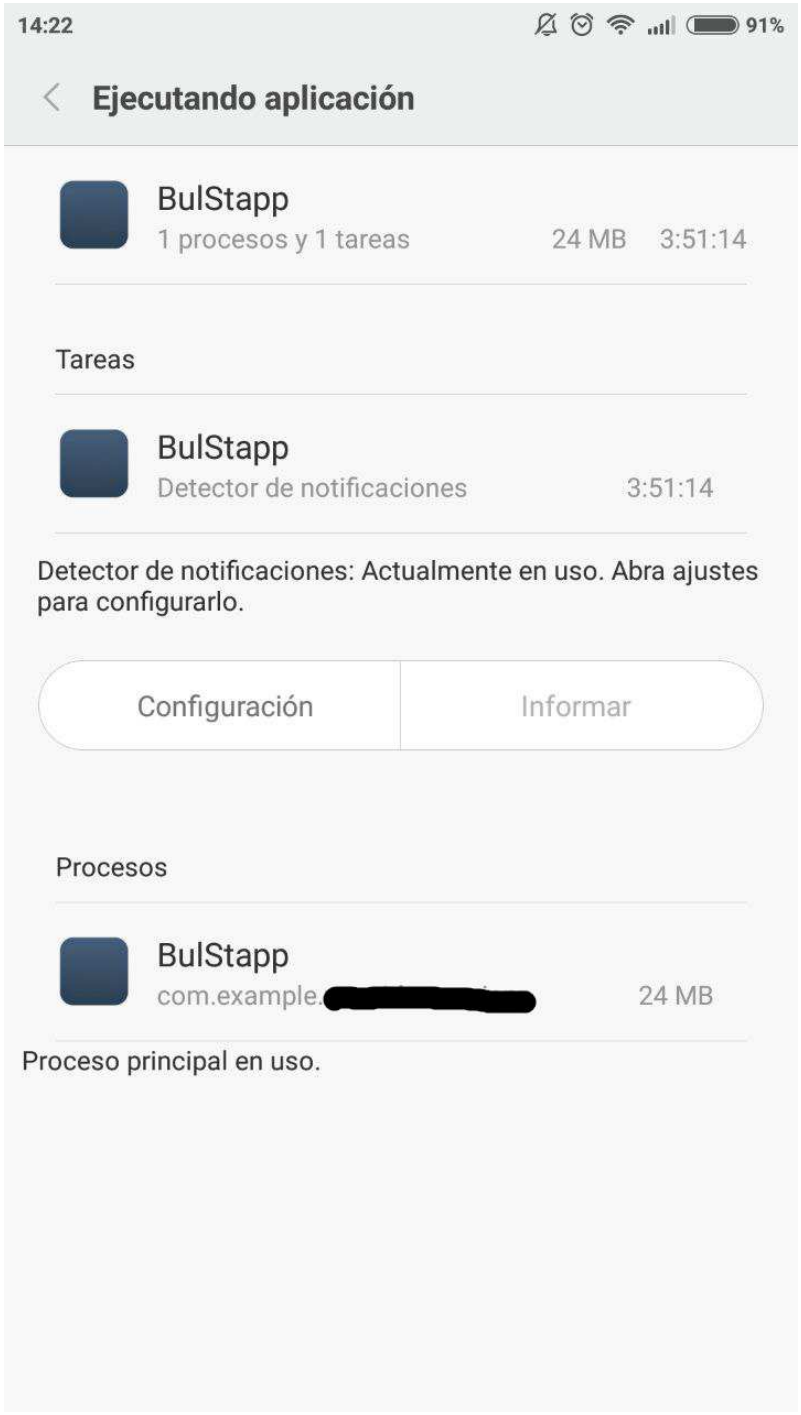

Figura 2.3: Proceso en ejecución con la aplicación oculta

2.5.1. Base de datos de notificaciones

A continuación, se proporcionan diferentes detalles de la base de datos de notificaciones utilizadas en forma de tablas. Los atributos en la tabla de datos no llevan tildes para evitar problemas internos, por lo que en estas tablas tampoco llevan.

- **Atributos**

nombre	**Tipo**	**Descripción**
id	INT	Identificador de la notificación.
aplicacion	TEXT	Aplicación que ha lanzado la notificación.
mensaje	TEXT	Texto que contiene el mensaje contenido en la notificación.
emisor	TEXT	Persona que envía el mensaje.

Cuadro 2.1: Tabla de atributos de la base de datos de notificaciones

- **Claves**

Atributo	**Clave primaria**	**Clave ajena**	**Clave alternativa**
id	X		
aplicacion	X		
mensaje	X		
emisor			X

Cuadro 2.2: Tabla de claves de la base de datos de notificaciones

2.5.2. Base de datos de contactos

A continuación, se proporcionan diferentes detalles de la base de datos de contactos utilizadas en forma de tablas. Los atributos en la tabla de datos no llevan tildes para evitar problemas internos, por lo que en estas tablas tampoco llevan.

- **Atributos**

nombre	**Tipo**	**Descripción**
nombre	TEXT	Representa el nombre del contacto.
numero	TEXT	Número de teléfono del contacto.

Cuadro 2.3: Tabla de atributos de la base de datos de contactos

- **Claves**

Atributo	**Clave primaria**	**Clave ajena**	**Clave alternativa**
nombre			X
numero	X		

Cuadro 2.4: Tabla de claves de la base de datos de contactos

Capítulo 3

Comunicaciones

A continuación se explica cómo se realiza todo el proceso de comunicación y las herramientas utilizadas para ello, entre los usuarios de BulStapp menores de edad y padres, tutores legales o cualquier otro contacto que haya sido vinculado desde la aplicación. Para ello se ha creado otra aplicación para los padres llamada **BulStapp Parent**, encargada de recibir las notificaciones cuando se detecte un posible caso de ciberacoso. Primero se explicará el funcionamiento interno de esta nueva aplicación. Posteriormente se explicará como se lleva a cabo la comunicación entre las dos aplicaciones. Por último se incluye un apartado que recoge toda la información sobre la seguridad aplicada a estas comunicaciones.

3.1. BulStapp Parent

En esta sección se explica el funcionamiento de la aplicación BulStapp Parent y las herramientas utilizadas para llevarlo a cabo. En la **figura 3.1** podemos apreciar el aspecto de la pantalla principal de esta aplicación y los elementos que la componen. Además, observándola podemos decir a simple vista que es bastante intuitiva y sencilla de utilizar, sobretodo en comparación con la aplicación principal, BulStapp.

3.1.1. Funcionamiento general

Básicamente, el funcionamiento de esta aplicación consiste en generar notificaciones cada vez que se detecta un caso de ciberacoso desde un móvil que tenga BulStapp instalada y vinculado a esta, en la base de datos de contactos visto en la **sección 2.5.2**, el número con el que nos hayamos registrado en esta aplicación. Para llevarlo a cabo, utilizaremos la herramienta **Firebase**, la nueva y mejorada plataforma de desarrollo móvil en la nube de Google, disponible para diferentes plataformas (Android, iOS, web). Además, provee una API para sincronizar datos en la nube en tiempo real [15]. Esta última funcionalidad, es de la que nos aprovecharemos en nuestra aplicación, dado que podremos

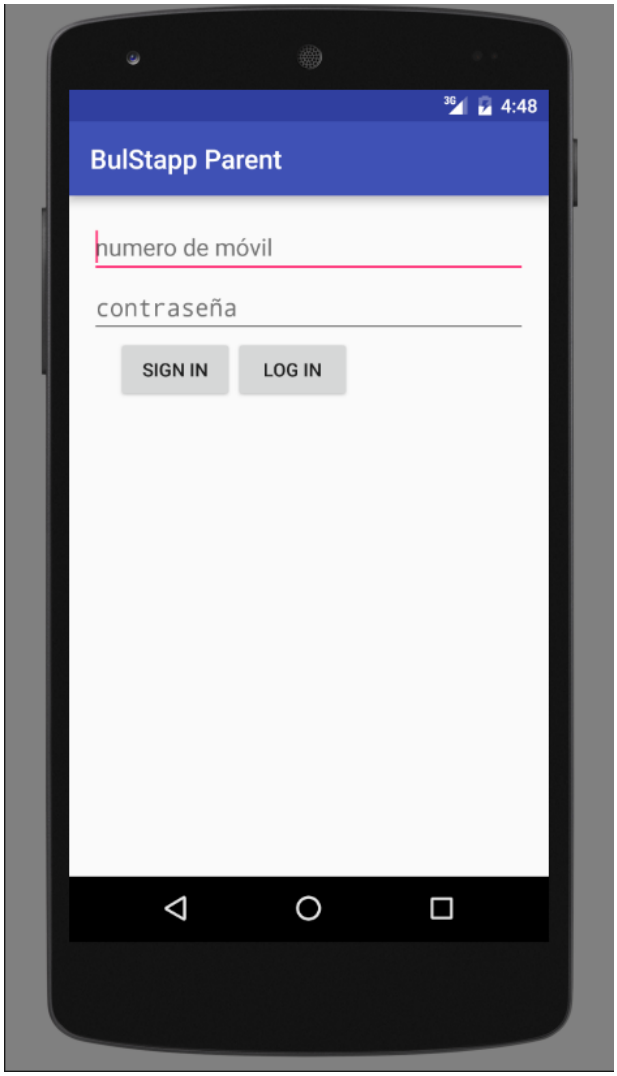

Figura 3.1: BulStapp Parent launcher

escribir en esta API desde BulStapp e inmediatamente recibir la información en BulStapp Parent.

Esta aplicación, funciona con un registro dentro de la misma y un logueo. Los usuarios se registrarán con su número de teléfono y una contraseña personal e intransferible en el launcher de esta aplicación. Aunque aparentemente se lleve a cabo un registro local, en realidad internamente nos estaremos registrando en Firebase, lo que se explicará en la **sección 3.1.2** del botón sign in. Análogamente, el logueo también será internamente en Firebase y se explicará en el **apartado 3.1.3** del botón Log In. La clave de su funcionamiento reside en un servicio ejecutándose en segundo plano, que además se inicia automáticamente cada vez que se enciende el dispositivo móvil. Este servicio utiliza el número con el que se loguea el usuario, que es el que debería recibir las notificaciones cuando un móvil con la aplicación Bulstapp detecta un caso de bullying. Este servicio se encarga de .[es]cuchar.[en] la plataforma Firebase, por si esta recibe algún aviso desde nuestra otra app para menores. Además, también se encarga de crear las

notificaciones que veremos en nuestros terminales. No obstante, esto es solo un resumen, ya que a lo largo de este capítulo se explicará todo con más detalle.

3.1.2. Botón Sign In

Este botón, como su propio nombre traducido indica, se encarga de llevar a cabo los procesos adecuados para registrarse en nuestra aplicación. Para ello debemos haber introducido nuestro número de teléfono, que tiene que coincidir con alguno que se haya añadido en la aplicación de los niños, BulStapp y una contraseña. Aunque este registro, como se comentó en el apartado anterior, parezca interno de nuestra aplicación, en realidad el registro se está llevando a cabo en Firebase. Para esto, se utiliza unos de los **modos de autenticación** propios de esta plataforma, que consta de un correo y contraseña. Este registro, nos da un nombre de usuario interno a Firebase con el que podemos controlar quién puede y quién no puede acceder a los datos, utilizando las reglas que nos facilita el entorno de Firebase.

Teniendo en cuenta el **párrafo anterior**, aparentemente algo no cuadra, pues el registro en nuestra aplicación se lleva a cabo con un número de teléfono, mientras que el modo de registrarse en Firebase precisa de un correo. Esto ocurre porque el correo de registro se crea desde dentro de la aplicación utilizando nuestro numero de teléfono, seguido de "@firebase.com". La forma de llevarlo a cabo la podemos observar en el fragmento de código que se muestra en la **figura 3.2**. Este está compuesto por dos funciones que son llamadas cuando se pulsa el botón del que hablamos, y se llama a una u otra dependiendo del contexto. Para este apartado nos interesa la primera, llamada "**signInToFirebase**". La segunda será comentada en el siguiente párrafo. Dicha función llama a una función propia de Firebase (la variable que podemos observar, myFirebaseRef, es un objeto de tipo Firebase) llamada "**createUser**"que recibe dos parámetros importantes en cada registro, el nombre de usuario y la contraseña. El primer parámetro que recibe es el nombre de usuario de tipo string, que si nos fijamos en la figura, podemos apreciar es una cadena que está formada por la concatenación entre el número de móvil introducido y "@firebase.com", como mencionamos en el apartado anterior. El segundo parámetro que nos interesa, es la contraseña, que si observamos la imagen se aprecia que se utiliza una variable llamada "passwd", que contiene la contraseña que hemos introducido en la aplicación antes de pulsar el botón.

Por otra parte, este botón no sirve solo para registrarse, sino que también sirve para eliminar a un usuario de la plataforma Firebase. Cuando un usuario es registrado en la aplicación, automáticamente cambia el texto del botón a "Sign Out". Cuando se pulsa el botón en este modo, este llama a la segunda función que se implementa en la **figura 3.2**, llamada "**signOutToFirebase**". Esta función es bastante parecida a la primera, que ya ha sido comentada

en el apartado anterior. La diferencia principal entre estas es la función a la que llaman desde dentro, que en este caso es a la función "**removeUser**"de la clase Firebase, que recibe los mismos parámetros que la función de Firebase comentada en el párrafo anterior. En resumen, a pesar de que ambas son propias de Firebase, una efectúa la acción contraria a la otra. No obstante, para que este paso tenga éxito, la contraseña y el email, formado por la concatenación del número de teléfono y la cadena "@firebase.com", como se explica en el párrafo anterior y como se puede observar en la figura, deben coincidir con los de algún usuario previamente registrado. Finalmente, en caso de que no coincida alguno de estos dos campos, no se llevará a cabo el borrado del usuario.

```
public void signInToFirebase(String num, String passwd) {

    myFirebaseRef.createUser(num+"@firebase.com",
            passwd,
            new Firebase.ValueResultHandler<Map<String, Object>>() {
        @Override
        public void onSuccess(Map<String, Object> result) {
            System.out.println("Successfully created user account with uid: "
                    + result.get("uid"));
        }
        @Override
        public void onError(FirebaseError firebaseError) {
            Log.e("signInToFirebase",firebaseError.toString());
        }
    });
}

private void signOutToFirebase(String num, String passwd){

    myFirebaseRef.removeUser(num+"@firebase.com",
            passwd,
            new Firebase.ResultHandler() {
        @Override
        public void onSuccess() {
            // user removed
        }
        @Override
        public void onError(FirebaseError firebaseError) {
            // error encountered
        }
    });
}
```

Figura 3.2: Función de registro en Firebase

3.1.3. Botón Log In

En esta sección, se explica el funcionamiento del botón "Log In", que aparece en la **figura 3.1**. Este sirve loguear al usuario dentro de nuestra aplicación con

los mismos datos que previamente se ha registrado. Además, al igual que pasaba al registrarse, el usuario aparentemente se loguea en nuestra aplicación, aunque sin embargo internamente se está logueando en Firebase. Además, tenemos el problema de registrarnos con un número de móvil y necesitar una dirección de correo electrónico, como pasaba con el botón de registrarse, y la solución es la misma que en el caso anterior, que es utilizar el número de teléfono móvil seguido de "@firebase.com"para simular una cuenta de correo electrónico.

Además, tal y como pasaba en el caso anterior, este botón también tiene una doble funcionalidad que cambia según lo usemos. Por defecto, el texto de este elemento es "Log In", tal y como se muestra en la **figura 3.1** y se utiliza la llamada a la función llamada "**logInToFirebase**", cuya implementación podemos observar en la **figura 3.3**. Esta implementación, a simple vista se puede estimar que es algo más compleja que la anterior, aunque en este caso el funcionamiento es casi el mismo. En primer lugar, realizamos una llamada al método "**authWithPassword**", perteneciente a la clase Firebase. Esta función es la encargada de comprobar si existe la pareja de nombre de usuario y contraseña, pasados como los dos primeros argumentos. Si no ocurre ningún error, el flujo de ejecución continua dentro de la función "**onAuthenticated**". Una vez dentro de esta, almacenamos el id que Firebase asigna al usuario. También se cambia el texto del botón de "Log In" a "Log Out" y se niega el contenido de la variable booleana login, que se pone a "true", porque previamente debe ser "false". El motivo de estas dos últimas acciones lo veremos en el siguiente apartado. A continuación, en el código se aprecia una condición que comprueba si el número con el que se está llevando a cabo el logueo es el mismo que el último número que se ha logueado desde el mismo móvil. Si este da positivo, se almacenará internamente el nuevo número y se invocará de nuevo el servicio, que tal y como se explica en el **punto 3.1.1**, se encargará de .ºcuchar" en la plataforma de Firebase, por si existe alguna incidencia vinculada a ese número. Además, este servicio también se encarga de dejar de escuchar en el número introducido anteriormente.

No obstante, este botón también tiene otra funcionalidad, como ocurría con el botón "Sign In". Dicha funcionalidad es la de desloguear al usuario previamente logueado, acción que se lleva a cabo una vez que el usuario se haya logueado. Tal y como vimos en el punto anterior, cuando un usuario se loguea en la aplicación, cambia el texto de "Log In" a "Log Out". Esta es la parte que ve el usuario y que sirve para dar feedback. Sin embargo, internamente existe una variable booleana llamada "login", que ya vimos en el punto anterior y que podemos apreciar en la **figura 3.3**. Esta variable es inicializada como "false" cuando se inicia la aplicación y una vez el usuario se loguea con éxito, su valor cambia a "true"Èsto lo que le indica al programa es a qué función llamar cuando se pulsa el botón del que hablamos en esta sección. En este caso, si el valor de la variable "logIn" es "true", se invoca a la función llamada "**logOutFirebase**", que encontraremos

en la parte baja de la **figura 3.3**. En esta podemos observar que simplemente llama a un método que desloquea de firebase, sea cual sea el usuario, pues no hace falta que coincida número de teléfono ni usuario. Además, esta función también cambia el texto del botón a "Log In", como aparecía anteriormente y vuelve a cambiar el valor de la variable "logIn" a "false". De este modo, la próxima vez que se pulse el botón se llevará a cabo el logueo nuevamente.

```
public void logInFirebase(final String num, String passwd) {

    myFirebaseRef.authWithPassword(num+"@firebase.com",
            passwd, new Firebase.AuthResultHandler() {
        @Override
        public void onAuthenticated(AuthData authData) {
            System.out.println("User ID: " + authData.getUid() +
                    ", Provider: " + authData.getProvider());

            uid=authData.getUid();
            bt_LogIn.setText("Log Out");
            logIn=!logIn;

            if (!settings.getString("numero","").equals(num)){

                Log.d("MAINACTIVITY","num= "+ num+", bd= "+
                        settings.getString("numero",""));
                editor = settings.edit();
                editor.putString("numero", num);
                editor.commit();

                Intent intent = new Intent(MainActivity.this,
                        ServiceListener.class);
                startService(intent);
            }
        }
        @Override
        public void onAuthenticationError(FirebaseError firebaseError) {
            Log.e("logInFirebase",firebaseError.toString());
        }
    });
}
public void logOutFireBase(){
    myFirebaseRef.unauth();
    bt_LogIn.setText("Log in");
    logIn=!logIn;
}
```

Figura 3.3: Función de logueo en Firebase

3.2. Comunicación

En este apartado se explica la comunicación entre nuestras dos aplicaciones, BulStapp y BulStapp Parent, y cuándo esta se lleva a cabo.

3.2.1. ¿Cuándo se necesita?

Recordando un poco el funcionamiento de nuestra aplicación, el objetivo principal es que los padres, madres y/o turores legales, puedan instalar esta aplicación a sus hijos o menores a su cargo y detectar posibles casos de ciberacoso en sus móviles, basándose en el texto de sus notificaciones y un filtro dentro de la aplicación. Además, los usuarios que han instalado esta aplicación deberían poder tener constancia de que se ha detectado una amenaza. Para ello, recordemos que en el **apartado 2.5** se comentó la existencia de bases de datos y se explicó que una de ellas almacena contactos a los que se les enviará el aviso de amenaza.

3.2.2. Antecedentes

Fueron varias las ideas y variaciones pensadas antes de llegar a la solución definitiva, entre las que destacan las siguientes:

- **SMS:** Dado que dentro de la aplicación se proporcionaba el número de móvil de los contactos, se pensó en esta sencilla opción. Finalmente fue descartada dado el coste de dinero que suponía el envío de un SMS y porque sería más complicado añadirle nuestros propios mecanismos de seguridad.

- **Bluetooth:** Esta opción tenía muchos contras y pocos puntos a favor, por ejemplo, era muy complicado establecer el algoritmo de identificación de usuarios, la víctima y los padres debían estar cerca cuando se detectara una amenaza, debía estar el Bluetooth conectado en ambos dispositivos, este punto en contra además tenía dos puntos en contra, el gasto de batería del Bluetooth encendido a todas horas y el hecho de que el menor desconectase su Bluetooth que impediría enviar el aviso.

- **Correo electrónico:** Otra de las opciones en las que se pensó para comunicar a los menores con los adultos fue el envío de un correo electrónico. Esta opción no fue llevada a cabo debido al requerimiento de uso de otras aplicaciones como Gmail y cuentas en ellas, y porque dificultaba el implementar métodos de seguridad.

- **Notificaciones Push:** La última opción en la que se pensó y la que más prometedora resultaba, fue el uso de notificaciones Push. La tecnología

Push es una forma de comunicación en la que una aplicación servidora envía un mensaje a un cliente-consumidor. Es decir, es un mensaje que un servidor envía a una persona alertándolo de que tiene una información nueva. Lo que caracteriza esta tecnología es que es siempre el servidor el que inicia esta comunicación. Aunque el cliente no tenga interés en saber si hay algo nuevo, lo comunica siempre [16]. Para implementar notificaciones Push, se requiere el uso de Google Cloud Messaging incluido en los servicios ofrecidos por Google Play Services. Este método fue implementado en su mayor medida con éxito, excepto la parte de enviar las notificaciones desde nuestra aplicación principal, para lo que se requería el uso de un servidor propio. En esta implementación también se utilizaba la plataforma Firebase ya comentada en la **sección 3.1.1**. Esta opción fue descartada tanto por la necesidad de implementar un servidor y la complejidad que requería su implementación para la funcionalidad que tendría, como por las restricciones que ofrecía su funcionamiento. Además, se descubrió un método más sencillo que ya estaba prácticamente hecho con la implementación que había de la plataforma Firebase.

Finalmente, teniendo en cuenta la funcionalidad que proporcionaba Firebase mientras se usaban notificaciones Push, se llegó a la conclusión de que se podían simular estas notificaciones Push sin el uso de Google Cloud Messaging, ni sus requisitos, como tener instalado en nuestro dispositivo móvil la aplicación Google Play Services o una versión de Android superior a la 4.2.2, ni de un servidor propio.

3.2.3. ¿Cómo se realiza?

Para llevar a cabo una comunicación entre terminales, como ya se comentó en el apartado anterior, se ha optado por utilizar la plataforma Firebase. Esta nos proporciona una API, con una jerarquía en árbol, en la cual podemos escribir y leer en tiempo real. En esta funcionalidad se basa nuestra comunicación entre aplicaciones, escribiendo desde la aplicación principal (BulStapp) y leyendo desde la aplicación secundaria o para padres (BulStapp Parent). A continuación dividimos la sección en dos, una para la escritura y otra para la lectura.

- **Escritura:** En primer lugar hacemos un pequeño resumen sobre el funcionamiento de nuestra aplicación BulStapp, visto en la sección 2.1. En dicha sección se explica que nuestra aplicación tiene acceso a las notificaciones que reciba nuestro terminal. Si estas se tratan de notificaciones enviadas por aplicaciones de mensajería instantánea, como WhatsApp o Telegram, se extrae el texto de estas, que es el mensaje que nos han enviado. Este texto extraído de la notificación, es analizado mediante un filtro programado internamente en la aplicación con palabras y expresiones clave que puedan suponer una amenaza de ciberacoso. Una vez detectada

la amenaza, si se ha llegado a un número estipulado de mensajes amenazantes, se extrae también la aplicación mediante la que se ha recibido el mensaje y el emisor de este. Estos dos últimos datos son los que interesa enviar a la otra aplicación, así que una vez extraídos nos disponemos a escribirlos en la API de Firebase.

Llegados a este punto, tenemos a nuestra disposición los datos que debemos enviar en nuestra aplicación principal y en el dispositivo móvil en el que esta está instalada, por lo que el siguiente paso es enviarlos a todos los dispositivos móviles añadidos a la tabla de datos de contactos vista en la **sección 2.5.2**. Para realizar esta acción, debemos escribir en nuestra API creada en firebase, en este caso llamada "simple-firebase". Esta se ramifica, como podemos ver en la **figura 3.4**, siguiendo una jerarquía que vamos escribiendo los usuarios. En nuestro caso de esta rama principal colgará otra rama llamada "*contactos*". En esta rama es en la que se escribirán los datos especificados de la siguiente manera:

Figura 3.4: Jerarquía en árbol en Firebase

- En primer lugar, se crea un hijo en la rama contactos por cada número de móvil en la base de datos de contactos. Cada uno de estos hijos tiene de nombre el propio número de teléfono, que determinará a la otra aplicación que en esa rama están los datos que le corresponde leer.
- A continuación, en la aplicación BulStapp creamos un objeto que contenga los elementos que se precisan (aplicación y nombre del agresor), tal y como podemos apreciar en la figura 3.4.
- Finalmente, se escribe el contenido de este objeto como hijo de la rama de cada número, o sea, se escribe la misma información en tantas ramas como números existan en nuestra base de datos de contactos.

- **Lectura:** Para realizar la lectura de los datos mencionados en el punto anterior, nos beneficiaremos de las herramientas que nos aporta Firebase. Una de ellas, es la posibilidad de .[es]cuchar" en una o varias de las ramas. Esto quiere decir que cada vez que se produzca algún cambio en la rama y se dispara un evento, esto se le comunicará a nuestra aplicación. Además, permite actuar de una manera distinta dependiendo del evento que se haya producido. En este caso, solo realizaremos alguna acción cuando el valor de los datos hijos de nuestra rama cambien su valor.

  La implementación de esta función la podemos ver en la **figura 3.5**. Para comenzar, esta figura muestra el código de la función onStartCommand del servicio que lanza la aplicación BulStapp Parent, del que ya hemos hablado anteriormente en la sección **3.1.1**. La función de onStartCommand, se invoca siempre que se lanza el servicio al que pertenezca. Guarda una diferencia con respecto a la función onCreate y es que si el servicio está activo y se vuelve a lanzar, no se ejecuta la función onCreate, y sin embargo la función onStartCommand sí. A continuación explicamos el funcionamiento de esta:

  - En esta función, lo primero que hacemos es eliminar el "listener" en caso de existir. Esto es una medida de seguridad por si se ha logueado una persona y luego se loguea otra. Para que un mismo terminal no escuche en dos ramas distintas, primero se debe borrar el "*oyente*" de otra.
  - En segundo lugar, se especifica qué rama es la que se quiere escuchar. Para ello utilizamos el número introducido como nombre de usuario al loguearnos en la aplicación. Este paso está explicado en la sección 3.1.3. La rama de la que escuchamos será aquella que tenga de nombre este número de teléfono, que equivaldría a la que está expandida, por ejemplo, en la **figura 3.4**.
  - Por último, se crea el "listener.[en] la rama que hemos seleccionado. Este nos avisará cada vez que se produzca un evento y se llamará a una función u otra dependiendo del evento que se active. En este caso, para esta aplicación solo hemos implementado la función llamada onDataChange, que se activa cuando cambia algún valor que esté bajo la rama señalada. Cuando esto ocurre, si los datos no están vacíos se llama a la función generarNotificación, que es una función propia del servicio que se encarga de generar la notificación. En la **figura 3.6**, podemos observar un ejemplo de una notificación que se recibiría en un dispositivo móvil con la aplicación BulStapp Parent. En esta se puede observar que tiene la misma estructura y los mismos datos que la rama en la API de Firebase de la que hemos escuchado.

Una vez dicho lo anterior, podemos darnos cuenta de que esta comunicación

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Firebase.setAndroidContext(this);

    if (listenerValue!=null)
        myFirebaseRef.removeEventListener(listenerValue);

    if (!settings.getString("numero","").equals("")) {

        myFirebaseRef = new Firebase(FIREBASE_URL).child(FIREBASE_CHILD).child(
                settings.getString("numero", ""));

        listenerValue = new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot snapshot) {
                if (snapshot.getValue()!=null) {
                    generarNotificacion(snapshot.getKey(), snapshot.getValue().toString());
                    System.out.println(snapshot.getValue());
                }
            }

            @Override
            public void onCancelled(FirebaseError error) {
            }
        };
        myFirebaseRef.addValueEventListener(listenerValue);

    }
    return super.onStartCommand(intent, flags, startId);
}
```

Figura 3.5: Función OnStartCommand del servicio de BulStapp Parent

no conecta un terminal con otro propiamente como cabía esperar, sino que gracias a la integración en tiempo real que nos ofrece Firebase generamos esa sensación, dado que según se escribe en una aplicación se genera el evento para que la otra aplicación pueda leer.

Para terminar esta sección, cabe mencionar que una misma rama en la API de Firebase de las que hemos hablado puede ser escrita por varias personas, pero solo puede ser leída por una. Esto es bastante útil si se tiene más de un menor a cargo de una misma persona, pues de esta forma podrá recibir aviso de cualquiera de ellos.

Figura 3.6: Ejemplo de notificación en BulStapp Parent

3.3. Seguridad

A continuación se explicará la seguridad implementada en la comunicación explicada en la sección anterior, si nos fijamos en la **figura 3.4** podemos observar que los datos están bastante accesibles para todo el mundo y se pueden exponer datos personales como nombres o números de teléfono que podrían ser usados con fines malignos. Para evitar acciones malintencionadas, se llevará acabo un cifrado al escribir los datos en la aplicación principal (BulStapp) y análogamente un descifrado en la aplicación secundaria (BulStapp Parent).

3.3.1. Algoritmo

El algoritmo utilizado ha sido el Advanced Encryption Standard (AES), también conocido como Rijndael (pronunciado Rain Doll.[en] inglés), este es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos. El AES es uno de los algoritmos más populares usados en criptografía simétrica. Por otra parte, estrictamente hablando, AES no es precisamente Rijndael (aunque en la práctica se los llama de manera indistinta) ya que Rijndael permite un mayor rango de tamaño de bloques y longitud de claves; AES tiene un tamaño de bloque fijo de 128 bits y tamaños de llave de 128, 192 o 256 bits, mientras que Rijndael puede ser especificado por una clave que sea múltiplo de 32 bits, con un mínimo de 128 bits y un máximo de 256 bits [17]. En nuestro caso, utilizaremos este cifrado AES con una llave de 256 bits.

3.3.2. Modo de operación

En criptografía, un modo de operación es un algoritmo que utiliza un cifrador por bloques para proveer seguridad a la información, como confidencialidad y autenticidad. Un cifrador por bloques en si mismo sólo es adecuado para la transformación criptográfica segura (cifrado y descifrado) de un grupo de bits de longitud fija llamado bloque (a menudo de 64 a 128 bits), en nuestro caso usaremos un bloque de 64 bits. Un modo de operación describe cómo aplicar repetidamente una operación de cifrado en un bloque simple para la transformación segura de cantidades de datos mayores que un bloque [18].

La mayoría de los modos requieren de una única secuencia binaria, usualmente llamada vector de inicialización (IV) para cada operación de encriptación. El IV no tiene que repetirse y para algunos modos es aleatorio. El vector de inicialización es utilizado para asegurar que se generen textos cifrados distintos cuando el mismo texto claro es encriptado varias veces con la misma clave. Los cifradores por bloques pueden manejar tamaños de bloque diferentes, pero durante la transformación el tamaño se mantiene siempre fijo. Los modos de

cifrado por bloques operan con bloques completos, por lo que es necesario que la última parte del bloque sea rellenada si su tamaño es menor que del actual. Sin embargo, hay modos que no necesitan el rellenado porque usan un cifrador por bloques como cifrador en flujo de forma muy efectiva [18].

Para nuestro algoritmo, utilizaremos el modo CBC (cipher-block chaining), cuyo funcionamiento es que antes de ser cifrado, a cada bloque de texto se le aplica una operación XOR con el previo bloque ya cifrado. De este modo, cada bloque cifrado depende de todos los bloques de texto claros usados hasta ese punto. Además, para hacer cada mensaje único se debe usar un vector de inicialización en el primer bloque. Este es el modo usado más a menudo. Su principal contrapartida es que es secuencial y no puede funcionar en paralelo [18]. Para entender mejor este procedimiento, tanto para cifrar como para descifrar, se puede observar la **figura 3.7**.

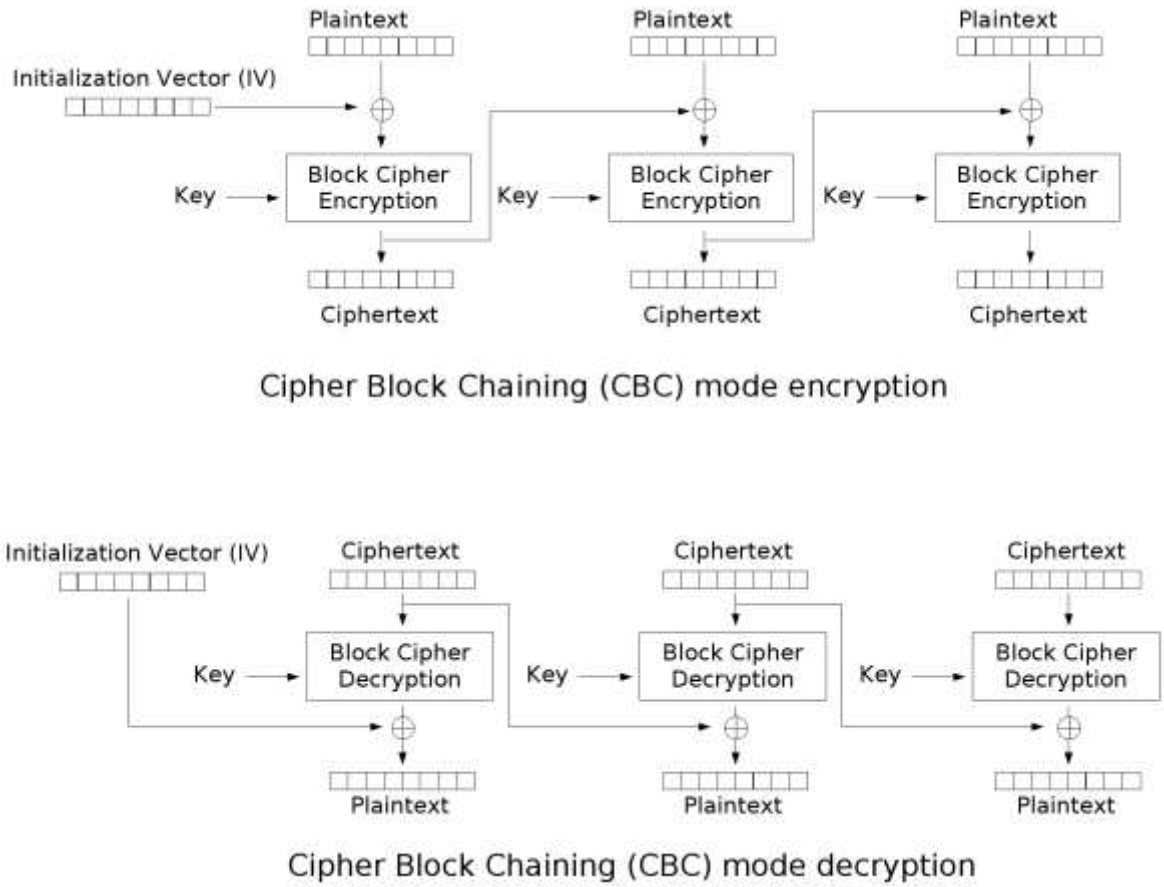

Figura 3.7: Funcionamiento del modo de cifrado CBC

3.3.3. Clave privada

Conociendo ya el algoritmo y el modo de cifrado, solo queda establecer una clave que conozcan tanto el emisor como el receptor. Para ello, se ha optado por

escoger un dato que conozcan ambas aplicaciones. En la aplicación principal, se utilizará el número de teléfono almacenado en la base de datos de contactos, dado que puede haber varios contactos, cada uno llevará un cifrado distinto que dependerá únicamente de su número de contacto, que como ya vimos en el **apartado 2.5.2** es clave primaría, ergo no habrá más de un contacto con el mismo número. Con respecto a la aplicación secundaria, la clave será el número con el que se loguea el usuario en la aplicación. Dado que ambas aplicaciones estarán conectadas mediante ese número no existirá ningún problema a la hora de cifrar o descifrar y como cada número es distinto y cada uno corresponde únicamente a una sola persona, aunque una persona intente descifrar los datos de otra no conseguiría descifrar el mensaje, pues las claves no coincidirían.

Por otra parte, los usuarios desconocen completamente el valor de la clave, dado que esta trabaja internamente en la aplicación y los usuarios no tendrán acceso al código. Esto, implica mayor seguridad, pues los datos tampoco podrán ser descifrados manualmente. Además, tampoco tendrán acceso al link de la API de Firebase, por lo que no podrán conocer los número que estarán dentro de la aplicación, protegiendo así esta información personal de los usuarios.

Para finalizar este capítulo, podremos observar un ejemplo de la estructura de nuestro árbol en la API de Firebase con los datos cifrados en la **figura 3.8**.

Figura 3.8: Datos cifrados en Firebase

Capítulo 4

BulStapp

En este capítulo veremos qué aspecto tiene la aplicación y se analizará cada uno de sus elementos. Para hacerlo, dividimos el capítulo en dos secciones, una por cada pantalla que tiene la aplicación. La primera será la **pantalla principal** y la segunda la **pantalla de ajustes**. A continuación, se expondrá una imagen de cada una con los elementos enumerados y se definirán individualmente en secciones.

4.1. Pantalla principal

En la **figura 4.1** podemos apreciar el aspecto de la **pantalla principal** y los elementos que la componen.

4.1.1. Botón Permisos

El elemento señalado con un 1 en la **figura 4.1** corresponde al botón de permisos. Este botón tiene la funcionalidad de abrir directamente la ventana que permite conceder el **permiso de notificaciones** visto en el **apartado 2.3**.

Una vez en esa ventana, para conceder este permiso solo habría que pulsar en la palanca que aparece desactivada, al lado del nombre de nuestra aplicación, y aceptar los permisos en un diálogo flotante que aparecerá.

4.1.2. Botón Escribir_BD

Este elemento tiene la función de escribir, en el espacio en blanco que aparece en la **figura 4.1**, todos los mensajes que se encuentren en la **base de datos de las notificaciones** ya comentada en el **apartado 3.1**. Este **espacio en blanco**, que en principio da la sensación de que está vacío, en realidad contiene una lista que no muestra nada hasta que se pulsa este botón. Además, si la lista no está vacía, al pulsar el botón al que nos referimos en este apartado esta se borra por completo.

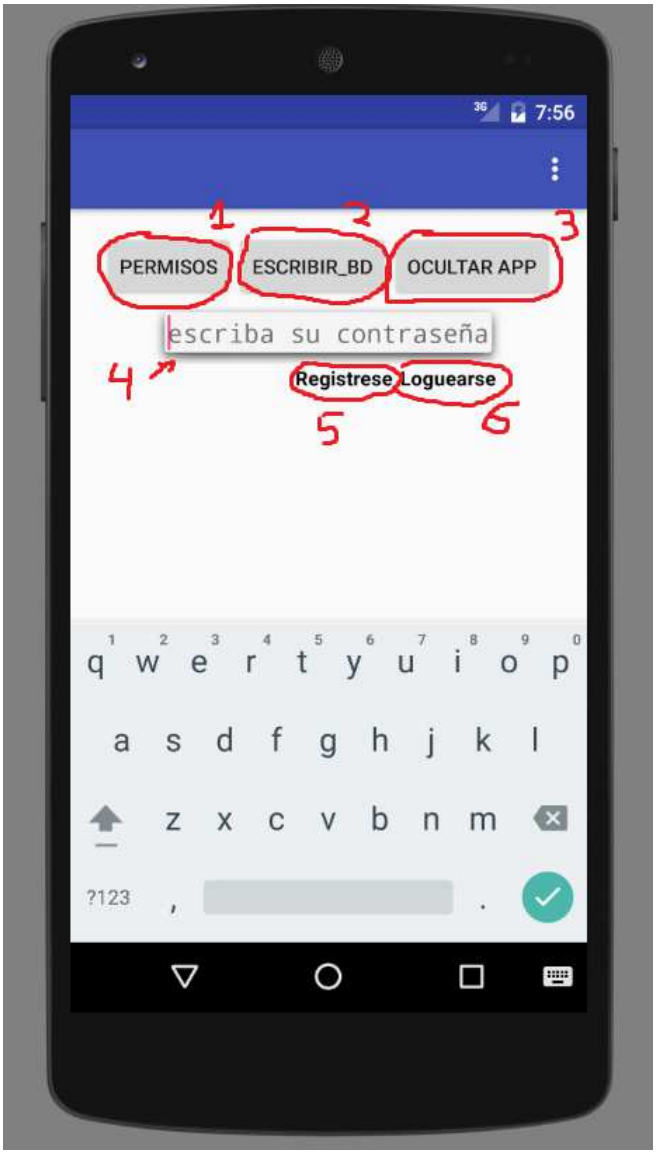

Figura 4.1: Pantalla principal de BulStapp

Por otra parte, solo encontraremos este elemento en la **fase beta** de la aplicación, por lo que si algún día esta aplicación sale al mercado sería sin este botón, o al menos en la **pantalla principal**.

4.1.3. Botón Ocultar APP

Este botón es bastante importante para el concepto de la aplicación, ya que nos va a servir para **ocultarla**, que tal y como se vio en la **sección 2.2** es una de las características más importantes. Al ser pulsado, este botón genera un diálogo flotante, que podemos observar en la **figura 4.3**. Se avisa de que se ocultará la aplicación y lo que se debe hacer para descubrirla. Como podemos observar en esta figura, el diálogo emergente posee un botón que dice "**OK**", que será el verdadero encargado de ocultar la aplicación, ergo, si no se pulsa

Figura 4.2: Diálogo para permitir permiso de notificaciones

ese botón no se ocultará la aplicación. Una vez que nos aparezca la **ventana flotante**, podemos cerrarla sin necesidad de pulsar el **botón de OK**. Para ello solo debemos pulsar el **botón de retroceso** que incorporan todos los **dispositivos** con sistema operativo Android.

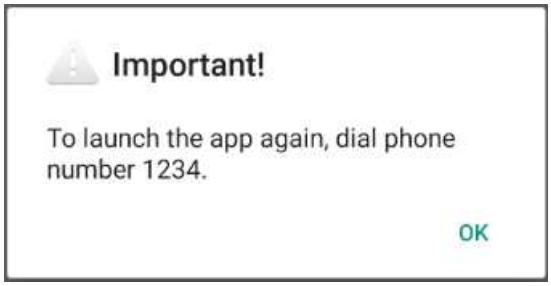

Figura 4.3: Diálogo que informa de que va a ocultar de la aplicación

4.1.4. Campo Contraseña

En este campo se escribirá la contraseña para poder acceder a los ajustes, siempre y cuando se haya registrado una contraseña previamente.

4.1.5. Botón Regístrese

El **botón Regístrese**, servirá para establecer una contraseña única que nos permitirá acceder a la **pantalla de ajustes**. Al pulsarlo, abrirá un diálogo

emergente, que podemos observar en la **figura 4.4**, formado por dos campos. En estos campos se debe escribir la contraseña deseada y al darle al **botón de aceptar**, se comprueba si la contraseña es la misma en ambos campos y si es así, esta se establecerá como contraseña de la aplicación. Este elemento tiene una peculiaridad y es que desaparece una vez que se establece una contraseña, y no vuelve a aparecer a no ser que se borren los datos de la aplicación.

Figura 4.4: Diálogo emergente de registro

4.1.6. Botón Loguearse

Este botón servirá, como su propio nombre indica, para loguearse en la aplicación y poder acceder a la pantalla de ajustes. Su funcionamiento es bastante simple, aunque previamente se debe haber establecido una contraseña mediante el **botón Regístrese** como se explica en la **sección 4.1.5**. Cuando este es pulsado, comprueba el texto que esté escrito en el **campo contraseña** visto en la **sección 4.1.4** con la contraseña establecida y si estas coinciden se abrirá la pantalla de ajustes. De lo contrario, si estas no coinciden, se muestra un mensaje informando al usuario de que la contraseña no es correcta.

4.2. Pantalla Ajustes

En la **figura 4.5** podemos apreciar el aspecto de la **pantalla de Ajustes** y los elementos que la componen.

4.2.1. Campos Nombre y Número de contacto

Estos campos son bastante intuitivos, sirven para introducir contactos manualmente en la base de datos. En el campo de arriba se introduce el **nombre del contacto** y en el de abajo el **número del contacto**. Posteriormente, para añadir este contacto a la base de datos se debe pulsar el **botón añadir**, explicado en el **punto 4.2.2**.

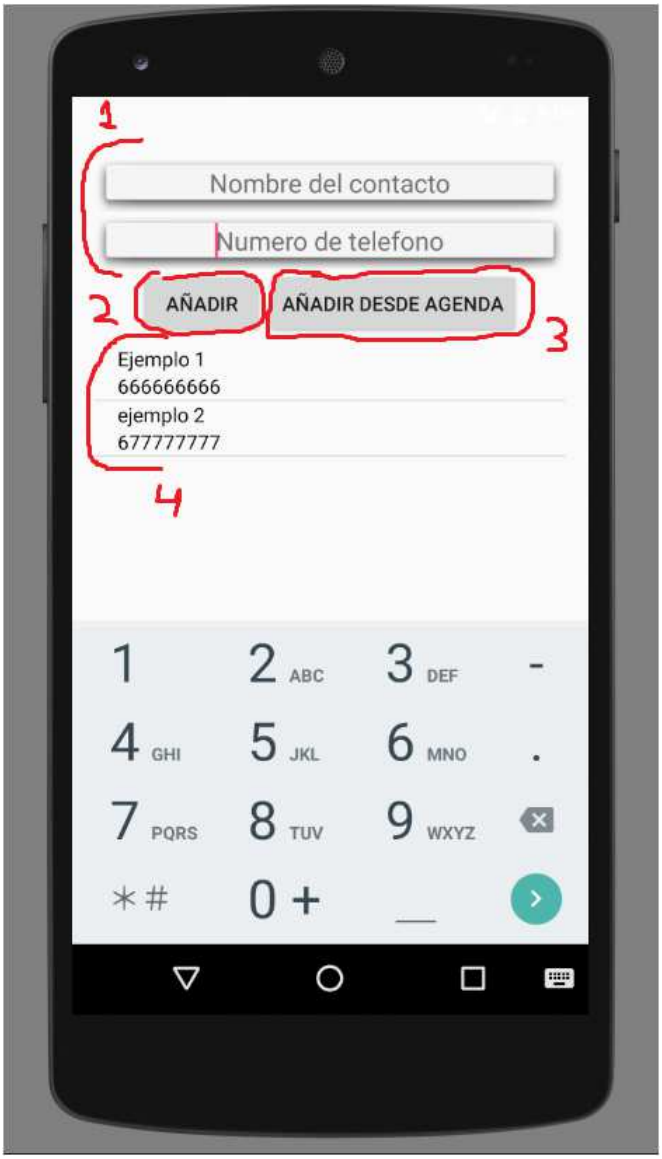

Figura 4.5: Pantalla de ajustes de BulsTapp

4.2.2. Botón Añadir

Como se ha comentado en la **sección anterior**, la función de este botón es añadir contactos manualmente a la base de datos de contactos ya comentada en la **sección 2.5.2**. Para llevarlo a cabo, en primer lugar comprobamos si ya existe el número escrito en dicho campo en la base de datos, ya que este es la **clave primaria**. Si este ya existe, se muestra un **mensaje informativo** al usuario y no se añade el número. Si por el contrario no existe, entonces se añade en la base de datos con normalidad.

Recordemos que los **atributos** de la base de datos de contactos son los mismos que los **campos** de los que trata esta sección, por lo que la inserción es bastante intuitiva.

4.2.3. Botón Añadir desde agenda

Este último botón, nos permite añadir un contacto desde la **agenda de contactos** que nos provee nuestro dispositivo móvil. Al pulsarlo, se abre la agenda de contactos, tal y como podemos apreciar en la **figura 4.6**, en la que podremos buscar el contacto que deseamos añadir. Una vez encontrado, basta con pulsar sobre este y se añadirá automáticamente a la base de datos de contactos, vista en la **sección 2.5.2**. No obstante, debemos recordar que el atributo número de la base de datos es **clave primaria**, por lo que como en el **apartado anterior** solo se añadirá el contacto a nuestra base de datos si no existe ya otro contacto con el mismo número.

Figura 4.6: Agenda de contactos

4.2.4. Lista de contactos

En esta **lista**, se mostrarán todos los contactos existentes en la base de datos de contactos. Además, se actualizará automáticamente cada vez que se añada algún contacto, por cualquiera de los métodos descritos anteriormente en los **apartados 4.2.2** y **4.2.3**.

Para actualizar esta lista, cada vez que se elimina o se inserta un elemento en la base de datos, se llama a la función que observamos en la figura 4.7. Antes de explicar el código que aparece en esta, hay que saber que cada elemento de la lista está compuesto por dos cuadros de textos situados uno encima de otro (el de arriba para los nombres y el de abajo para los números). Para almacenar de texto de todos los elementos, utilizaremos una lista para cada cuadro de texto; por lo que tenemos dos listas, llamadas "numeros" y "nombres".

El algoritmo de la función que aparece en la figura 4.7, funciona de la siguiente manera. Primero, se elimina el contenido de las listas "numeros" y "nombres", mencionadas anteriormente, para que no existan duplicidades en la lista de contactos. Luego, se realiza una consulta en la base de datos para extraer todos los datos que contenga y cada fila de la base de datos se almacenará en la variable fila. A continuación, se comprueba si la variable fila no está vacía, o dicho de otro modo, se comprueba que haya al menos un elemento almacenado en la de datos. Si el resultado es positivo se inicia un bucle que utiliza cada columna de una fila para añadir el dato de esta a la lista que corresponda. Finalmente se cierra la base de datos y se utiliza la última linea para actualizar la vista de la lista de contactos.

```
private void cargarBD(){

    numeros.clear();
    nombres.clear();

    tablaNumeros = new NumDB(Ajustes.this, "administracion", null, 1);
    SQLiteDatabase bd = tablaNumeros.getWritableDatabase();
    Cursor fila = bd.rawQuery(  //es una consulta devuelve filas.
            "select *  from numeros", null);

    if (fila.moveToFirst()) {  //si ha devuelto 1 fila, vamos al primero (que es el unico)
        for (int i=0;i<fila.getCount();i++) {
            nombres.add(fila.getString(0));
            numeros.add(fila.getString(1));

            fila.moveToNext();
        }

    } else
        makeToast("No existen numeros en nuestra base de datos",0);

    fila.close();
    bd.close();
    tablaNumeros.close();

    adapter.notifyDataSetChanged();

}
```

Figura 4.7: Función que carga la lista

Capítulo 5

Implementación

En este capítulo se nombran algunos aspectos que aparecieron en la implementación a lo largo de este trabajo.

5.1. Principales problemas

- **Extraer el texto de las notificaciones:** El principal problema con la notificaciones ha sido averiguar como se extrae el texto de cada notificación de distintas aplicaciones, ya que cada notificación crea la suya de forma diferente. Los problemas más graves se han dado a la hora de extraer el texto de las notificaciones, pues algunos no se extraen completos, sino un numero determinados de caracteres, seguidos de puntos suspensivos indicando que el mensaje continua. Otro problema ocurre cuando se reciben varias notificaciones de la misma aplicación, pues en vez de mostrar los mensajes muestran cuantos mensajes existen sin leer y de cuantas conversaciones. Esto hace que nuestra aplicación pierda algo de eficacia en mensajes más largos o en casos de muchos mensajes de acoso en poco lapso de tiempo mientras el usuario no los haya leído.
- **Algunas funciones de extracción de mensaje podrían quedar obsoletas:** Si se llegase a dar el caso de que Android o alguna de las aplicaciones utilizadas cambiasen la estructura de sus notificaciones la función encargada de extraer esa notificación quedaría obsoleta.
- **Cuando salgan nuevas aplicaciones de mensajería instantánea BulStapp no podrá actuar sobre ellas:** Dado que la extracción del mensaje se ha implementado en función de la aplicación de la que se tratase, si saliera al mercado otra aplicación no dispondríamos de ningún método para extraer el mensaje de sus notificaciones.
- **Comunicación entre móviles:** Este fue un problema bastante importante, dado que la comunicación sirve para avisar a los padres de la amenaza de acaso producida y el aviso es una de las principales características

de esta aplicación. Los métodos para realizarla eran bastante escasos y siempre se encontraba algún pero.

- **Factores que matan procesos:** Este problema retrasó mucho el desarrollo de la aplicación, puesto que se dedicó bastante tiempo a solucionar un problema que no existía, simplemente era un problema de Android. La base de este problema es que hay ciertos factores como el modo ahorro de batería o aplicaciones para mejorar el rendimiento, que matan procesos según su criterio. Por esa razón, a veces nuestro servicio misteriosamente dejaba de funcionar y se pensaba que era un fallo interno, cuando realmente no dependía de nuestra aplicación.

- **Recibir notificaciones:** El problema de trabajar con una aplicación que requiera recibir notificaciones de una serie de notificaciones es que no siempre estamos recibiéndolas. En nuestro caso, como ya se comentó en un punto anterior, cada aplicación trata de una forma distinta sus notificaciones, por lo que la manera extraer el texto de ellas será diferente para cada aplicación. Esto supone que para comprobar si estamos realizando bien la extracción debemos recibir varias notificaciones de distintas aplicaciones, pues a lo mejor una de las aplicaciones no genera problemas pero el resto sí. el verdadero problema es que en aplicaciones de mensajería directa se depende de terceras personas para recibir notificaciones.

- **La aplicación no funciona si los mensajes llegan mientras el usuario tiene la ventana del chat abierta:** Esto quiere decir que si el usuario se encuentra inmerso en una conversación en tiempo real, nuestra aplicación no puede ejercer su funcionamiento habitual. El motivo de esto, es que nuestra aplicación trabaja con las notificaciones y al usuario tener la ventana de chat abierta, la aplicación no envía notificación, por lo que nuestra aplicación no tiene acceso al mensaje.

- **Vulnerabilidad de datos en Firebase:** En la **figura 3.4** podemos observar la estructura del árbol en la API de Firebase, en esta podemos observar que los números están al descubierto y prácticamente cualquiera podría escribir en ellos, esto supondría generar avisos falsos a los padres. Esto es difícil de que ocurra porque se tendría que encontrar el enlace a nuestra API de Firebase.

5.2. Soluciones

- **Extraer el texto de las notificaciones:** Este problema se ha conseguido solucionar a medias, pues se ha conseguido extraer el texto, la aplicación y el emisor de todas las aplicaciones más usadas como WhatsApp,

Telegram, Gmail, SMS, Messenger. Sin embargo, en algunas aplicaciones como Telegram no se ha conseguido extraer el mensaje completo. En otras aplicaciones como WhatsApp, no se han conseguido desglosar las notificaciones cuando existen varios mensajes y nos sigue devolviendo lo mismo que se comentó anteriormente, este es un tema complicado dado que WhatsApp utiliza su propio método de notificaciones.

- **Algunas funciones de extracción de mensaje podrían quedar obsoletas:** Para solucionar este problema, la única solución posible es modificar los métodos necesarios manualmente o contratar a otro informático que lo haga.
- **Cuando salgan nuevas aplicaciones de mensajería instantánea BulStapp no podrá actuar sobre ellas:** Al igual que en el punto anterior solo existe una solución para solucionar este problema y esta es implementar los métodos necesarios manualmente o contratar a otro informático que lo haga.
- **Comunicación entre móviles:** Este Problema, fue solucionado creando una aplicación para padres llamada BulStapp Parent y se comunican mediante la API de Firebase. Todo este proceso está explicado en el **capítulo 3**.
- **Factores que matan procesos:** Para solucionar este problema se desactivaron todas las aplicaciones que optimizaran el rendimiento del dispositivo móvil y las funciones de ahorro de batería. En el caso de los usuarios deberían hacer lo mismo, la única solución viable será recomendarlo en las instrucciones. Aunque con las nuevas APIs de Android se podrá aplicar o quitar el ahorro de batería a aplicaciones específicas, esto podría favorecer a nuestra aplicación.
- **Recibir notificaciones:** Este problema se ha solucionado con paciencia, esperando a las horas en que más se habla por aplicaciones de mensajería directa y manteniendo conversaciones mediante estas. Estos periodos de tiempo de actividad fueron aprovechados al máximo, mientras que por las noches se trabajó en otros aspectos de la aplicación.
- **La aplicación no funciona si los mensajes llegan mientras el usuario tiene la ventana del chat abierta:** Este problema, lamentablemente no tiene solución por nuestra parte, dado que las aplicaciones sobre las que se trabaja no permiten el acceso a sus mensajes.
- **Vulnerabilidad de datos en Firebase:** Esto se ha solucionado aplicando a los datos escritos en la API de Firebase el cifrado explicado en la **sección 3.3** de este documento.

Capítulo 6

Presupuesto

En este capítulo se estimará un presupuesto total para el desarrollo de este trabajo, estos presupuestos podrían variar notablemente dependiendo de la situación y del valor del mercado. Por otra parte, es preciso aclarar que desarrollar este trabajo no ha supuesto coste alguno, dado que no ha hecho falta contratar personal ni comprar ninguna tecnología o material. Se ha trabajado con un móvil y un ordenador personal y con un móvil del departamento de Ingeniería Informática y Sistemas de la Universidad de La Laguna.

6.1. Personal

Para desarrollar este trabajo necesitaremos de forma profesional se necesita contratar personal cualificado, en este caso se precisará un informático que desarrolle las aplicaciones y todas sus funcionalidades, un psicólogo familiarizado con el campo del ciberacoso que diseñe teóricamente el filtro mediante patrones de conducta y un diseñador gráfico que se encargue del aspecto de las aplicaciones. A continuación se añade una sección por cada profesional que se necesite y en esta se desarrollará el presupuesto individual aproximado que requeriría cada uno.

6.1.1. Informático

Teniendo en cuenta que en España el sueldo de un informático es de media 12 euros la hora y sumando todas las horas que aparecen en la **tabla 6.1**, cuyo resultado es 254, haciendo una simple multiplicación obtenemos un coste aproximado de **3.048 euros** en total.

6.1.2. Psicólogo

Este presupuesto es el más complicado de estimar debido a que es un trabajo particular para un profesional de la psicología. Para calcularlo, se procederá a

Tarea	**Horas**
Estudio sobre el campo de trabajo (ciberacoso)	8
Estudio del arte	8
Instalación de programas, plugins y emuladores necesarios	8
Estudio de las herramientas y lenguajes de programación necesarios para implementar todas las funcionalidades	30
Desarrollo del código fuente de BulStapp	120
Desarrollo del código fuente de BulStapp Parent	30
Estudiar los métodos mediante los que se pueden conectar las dos aplicaciones desarrolladas	15
Implementar el método elegido para la conexión entre las dos aplicaciones	10
Buscar información sobre los tipos de cifrado válidos para el proyecto y selección del algoritmo adecuado	10
Implementar el algoritmo de seguridad	15
Total	254

Cuadro 6.1: Tareas a realizar por el informático contratado

dividir las tareas a realizar y estimar un tiempo para cada una.

Una vez realizada la estimación, la cual puede observarse en la **tabla 6.2**, la suma de las horas asciende a 82, lo que equivale a casi un mes de trabajo cumpliendo con el estatuto de los trabajadores. Por tanto, teniendo en cuenta el salario medio de un psicólogo que es aproximadamente 1.500 euros mensuales [19] y que las horas que se han calculado equivalen a medio mes, el presupuesto para el psicólogo se aproxima a **750 euros**.

6.1.3. Diseñador gráfico

Tras una exhaustiva búsqueda sobre el coste que supondría contratar a un diseñador gráfico, podría asegurar que al ser un trabajo rápido que no requiere de gente excesivamente experimentada, el presupuesto rondaría entre los 150 y los 300 euros, dependiendo del nivel de detalles y el tiempo invertido.

Por otra parte, en la **tabla 6.3** se puede observar que tareas realiza este profesional y cuanto tiempo se estima que dure cada una. Teniendo en cuenta los datos que figuran en la misma, el trabajo realizado se elevaría a unas 47 horas y al tratarse de un trabajo no muy complicado, se estima que el presupuesto para este profesional sería de **200 euros** aproximadamente.

Tarea	Horas
Acordar con el cliente las bases del filtro	5
Realizar una búsqueda de información e introducción al tema en cuestión	8
Analizar y estudiar casos recientes de ciberacoso infantil	8
Realizar un borrador de palabras y frases más comunes en los casos de ciberbullying	15
Familiarizarse con la jerga juvenil para evitar confundir casos de ciberacoso con expresiones populares o propias del argot infantil de la nueva generación	16
Reunir toda la información obtenida y crear el filtro en base a ella	30
Total	82

Cuadro 6.2: Tareas a realizar por el psicólogo contratado

Tarea	Horas
Conocer el tipo de aplicación y qué clase de imagen corporativa se quiere dar a conocer	4
Llegar a un acuerdo con el cliente sobre el tipo de interfaces a diseñar	4
Realizar un estudio previo, y diseñar varias propuestas para presentar posteriormente al cliente	7
Suponiendo que no se estuviese de acuerdo con ninguna de las propuestas se realizarían nuevas propuestas	7
Diseño de logotipo, imagotipo e isotipo	10
Diseño de interfaces internas de la aplicación	10
Presentación de los diseños al cliente	5
Total	47

Cuadro 6.3: Tareas a realizar por el diseñador gráfico contratado

6.2. Material

A continuación se listará una serie de posibles materiales, suponiendo que podrían hacer falta en caso de no contar con el ellos. Se calculan precios en la gama media que es la estándar en relación calidad/precio y no quedarán obsoletos muy pronto, sirviendo así para futuros trabajos.

Los materiales necesarios pueden observarse en la **tabla 6.4**. En esta aparecen reflejados dos móviles, esto es porque se necesita comprobar que funciona bien la parte de la comunicación entre las dos aplicaciones desarrolladas en este

trabajo y un ordenador con el que se desarrollar todo el código fuente donde se pueda además buscar la información necesaria para la parte más teórica del desarrollo.

Tipo	**Cantidad**	**€/u**	**Precio total(€)**
Teléfono móvil de gama media	2	200	400
Ordenador de gama media	1	400	400
Total			800

Cuadro 6.4: Presupuesto estimado para el material

6.3. Presupuesto total estimado

Para concluir este capítulo, se unifica el contenido visto en las secciones anteriores, en las que se hacen estimaciones de presupuesto individual, para hallar el presupuesto total aproximado. Para ello elaboramos la **tabla 6.5**, en la que se puede apreciar como se trabaja con las cifras obtenidas en las secciones anteriores del capítulo.

Tipo	**nombre**	**Precio(€)**
Personal	Informático	3.048
	Psicólogo	750
	Diseñador gráfico	200
Material	Móviles	400
	Ordenador	400
Total		5.398

Cuadro 6.5: Presupuesto total estimado

Finalmente obtenemos el resultado final que como se aprecia en la tabla se eleva a 5.398 euros. Cabe destacar que esta cantidad es muy variable, dado que todos los precios y horas calculados en los apartados anteriores han sido solo una estimación.

Capítulo 7

Conclusiones y líneas futuras

En este trabajo se ha llevado a cabo la implementación de una aplicación para combatir un problema real que afecta a bastantes personas, no solo en nuestro país sino en prácticamente todo el mundo, y que preocupa mucho a una gran parte de la población, el ciberacoso.

No hace falta mencionar que la tecnología nos invade cada día más, llegando a los más pequeños cada vez en edades más tempranas. Si bien es cierto que la tecnología nos aporta utilidades muy positivas como la mensajería instantánea, redes sociales con las que mantenernos en contacto con nuestros conocidos, facilidad y rapidez para la búsqueda de información, programas para realizar videollamadas, entre una larga listas de cosas, también es cierto que estas utilidades pueden suponer una amenaza si se utilizan incorrectamente y sin pensar en las consecuencias que las acciones realizadas puedan tener.

Por ello, es un problema que un teléfono móvil llegue a las manos de un niño sin enseñarle previamente los peligros que puede conllevar, pues puede hacer un mal uso del mismo, bien acosando a otros niños o siendo víctima. Además, con este tipo de tecnología los menores están expuestos al denominado "grooming", también conocido çhild grooming.º Ïnternet grooming", definido como una serie de conductas y acciones deliberadamente emprendidas por un adulto con el objetivo de ganarse la amistad de un menor de edad, creando una conexión emocional con el mismo, con el fin de disminuir las inhibiciones del niño y poder abusar sexualmente de él. Además en algunos casos, incluso se puede pretender la introducción del menor en el mundo de la prostitución infantil o de la producción de material pornográfico [20].

Teniendo en cuenta todos estos datos, la motivación para realizar esta aplicación está clara pues su objetivo es intentar detectar y evitar casos de ciberacoso infantil mediante el control de mensajes recibidos en el teléfono móvil, extrayendo y analizando el contenido de los mismos mediante las notificaciones.

Es inevitable que la tecnología tarde o temprano acabe llegando a los más pequeños, pero esto no tiene porque ser malo. Uno de los objetivos de esta aplicación, visto en la sección 1.2, es proteger a los jóvenes que puedan ser víctimas de algún tipo de ciberacoso, para que así puedan disfrutar de la parte más didáctica y recreativa de los grandes avances del mundo tecnológico. Además, otro de los objetivos mencionados anteriormente, y bastante importante, es la seguridad y confianza de los padres hacia sus hijos, evitando la sobreprotección o las prohibiciones excesivas que provocan ese factor que hace que los niños deseen lo prohibido.

Por otra parte, como líneas futuras quedan las siguientes tareas por desarrollar:

- Realizar un filtro muy detallado, con la ayuda de un/a profesional de la psicología, para reducir la posibilidad de crear falsas alarmas e intentar que no se pase por alto ninguna expresión o palabra clave.
- Arreglar el problema existente con la lectura de notificaciones que se ha comentado en la sección 5.1.
- Incrementar la seguridad, en caso de que sea posible, mediante por ejemplo la implementación del esquema de Diffie Hellman con curvas elípticas.
- Optimizar el código, ya que este es uno de los puntos más importantes en el desarrollo de cualquier aplicación y lamentablemente no ha habido tiempo de llevarlo a cabo en este proyecto.
- Mejorar las interfaces gráficas, pues las aplicaciones necesitan una buena imagen que haga que la primera impresión sea positiva. Para llevarlo a cabo, lo mejor sería contratar a un diseñador gráfico, como se contempla en el capítulo 6.
- Diseñar para las dos aplicaciones desarrolladas un icono propio, personal y que transmita y represente los objetivos.
- Implementar unas reglas en la API de Firebase para que solo puedan leer y escribir las personas adecuadas. Esto podría implicar desarrollo adicional de las aplicaciones o de los datos utilizados en Firebase.

Capítulo 8

Summary and conclusions

This work contains the implementation of an application to combat a real problem that affects many people, not only in our country but virtually everywhere, and that involves a great concern to a large part of the population, cyberbullying.

Needless to mention that the technology invades us every day, reaching the children each time at younger ages. While it is true that technology gives us very positive utilities such as instant messaging, social networks that keep us in contact with our acquaintances, easy and quick information search, programs for video calls, among a long list of useful tools, it is also true that these utilities can pose a threat if used incorrectly without considering the consequences that actions may have.

Therefore, it is a problem that a mobile phone reaches the hands of a child without first having taught him/her the dangers that it may entail, such as he/she can misuse it, either harassing other children or being a victim. In addition, with this technology children are exposed to the called "grooming", also known çhild grooming.º Ïnternet grooming", defined as the establishment of an emotional connection with a child, to lower his/her inhibitions, with the intention of child sexual abuse. Besides, in some cases, it lures minors into trafficking of children, illicit businesses such as child prostitution, or the production of child pornography [21].

Considering these data, the motivation to perform this application has been to try to detect and prevent cases of child cyberbullying, by monitoring messages received on the mobile phone, to extract and analyze their content.

Technology is inevitable that sooner or later ends up coming to the youngest, but this need not be bad. One of the purposes of this application, seen in Section 1.2, is to protect the younger ones who may be victims of some form of cyberbullying and that they can enjoy the part more educational and recreational of the great advances of the technological world. In addition, another of

the abovementioned objectives and quite important to consider is the safety and confidence of parents towards their children, avoiding overprotection or excessive prohibitions that cause the factor that makes kids wish the forbidden.

Moreover, as future lines the following tasks can be developed:

- Perform a very detailed filter with a professional psychologist, to reduce the possibility of creating false alarms and by contrast, do not ignore some expressions or keyword.
- Fix the existing problem with the reading of notifications that has been discussed in Section 5.1.
- Introduce more security if it were possible, one of the options had in mind it was to implement elliptic curves Diffie-Hellman key exchange.
- Optimize the code, this is one of the most important issues in the development of an application,but in this case there was no time to realize in this project.
- Improved graphical interfaces. Applications need a good image, because the first impression is very important. To carry it out, it would be best to hire a graphic designer, as you can see in the Chapter 6.
- For the two developed applications, design an own icon, which is personal and that this represents the objectives of these applications.
- Implementing rules in Firebase API so that only can be read and write to this, the appropriate person and not just anyone. This could involve further development of the applications or data used in Firebase and should also be implemented.

Bibliografía

[1] Microsoft. Un 37 % de los jóvenes españoles sufre ciberacoso.

[2] Olga R. Sanmartín. El 'ciberacoso' comienza a los 10 años. *EL MUNDO*, 2015.

[3] María Ruiz Nievas. España figura entre los países donde más ciberacoso sufren los menores. *Agencia EFE*, 2016.

[4] STOPit page. http://store.samhsa.gov/apps/knowbullying/index.html.

[5] KnowBullying page. http://stopitcyberbully.com/.

[6] Bully Button page. https://itunes.apple.com/es/app/bully-button/id510884692?mt=8.

[7] My Mobile Watchdog page. https://www.mymobilewatchdog.com/.

[8] Net Nanny page. https://www.netnanny.com/.

[9] Bully Tag page. http://www.bullytag.com/.

[10] Anonymous Alerts page. http://www.anonymousalerts.com/webcorp/.

[11] Lempa, P., Ptaszynski, M. & Masui, F. Cyberbullying blocker application for android.

[12] PocketGuardian page. https://gopocketguardian.com/.

[13] ReThink page. http://es.4androidapps.net/tag/applications/nearparent-download-40025.html.

[14] NearParent page. http://www.rethinkwords.com/.

[15] Jose Angel Zamora. ¿qué es firebase? la mejorada plataforma de desarrollo de google, 2016. http://logistica.fime.uanl.mx/miguel/docs/BibTeX.pdf.

[16] Qode. ¿qué son las notificaciones push? *QodeBlog*, 2015. http://qode.pro/blog/que-son-las-notificaciones-push/.

[17] Wipedia AES page. https://es.wikipedia.org/wiki/Advance_Encryption_Standard.

[18] Wikipedia. Modos de operacion de una unidad de cifrado por bloques.

[19] Cuánto gana un psicólogo. http://www.cuantoganaa.com/cuanto-gana-un-psicologo/.

[20] Wikipedia. Definición de grooming en español. https://es.wikipedia.org/wiki/Grooming.

[21] Wikipedia. Definición de grooming en inglés. https://en.wikipedia.org/wiki/Child_grooming.

Apéndice A

Apéndice

A.1. Algoritmo del servicio de BulStapp

```
*********************************************************************************
*
* NLServcice.java
*
*********************************************************************************
*
* Anael Eneas Melián Baute
*
*
* 5 de julio de 2016
*
*
* Trabajo realizado por el servicio de BulStapp cada vez que llega una notificación.
* Primero se llama a onNotificationPosted y dentro de esta se llama al resto de funciones
* que aparecen.
*
*
*********************************************************************************/

void enviarAviso(){

        Firebase.setAndroidContext(this);
        NumDB admin = new NumDB(this, "administracion", null, 1);
        SQLiteDatabase bd = admin.getWritableDatabase(); //Create and/or open a database
         that will be used for reading and writing.


        Cursor fila = bd.rawQuery(  //es una consulta
                "select numero from numeros", null);


        if (fila.moveToFirst()) {  //si ha devuelto alguna fila, señalamos a la primera.
            for (int i=0;i<fila.getCount();i++) {
                String num=fila.getString(0);
                Log.d("NLService::enviarAviso",fila.getString(0) );
                myFirebaseRef = new Firebase(FIREBASE_URL).child(FIREBASE_CHILD).child(num);

                Map<String, String> agresor = new HashMap<>();

                String em = AESUtil.encrypt(emisor, num);
```

```
            String ap = AESUtil.encrypt(aplicacion, num);

            agresor.put("nombre del agresor", em);
            agresor.put("aplicacion", ap);

            myFirebaseRef.setValue(agresor);

            fila.moveToNext();
        }

    }

    fila.close();
    bd.close();
    admin.close();

}


void checkLimit(String ap, String em){

    MyDB admin = new MyDB(this, "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase(); //Create and/or open a database
     that will be used for reading and writing.

    Cursor fila = bd.rawQuery(  //devuelve 0 o 1 fila //es una consulta
            "select *  from notificaciones where aplicacion= \'" + ap + "\' and
             emisor= \'" + em + "\'", null);

if (fila.getCount()%limite==0)
    enviarAviso();

    fila.close();
    bd.close();

}

boolean filtrado(String msg){

    boolean devuelve=false;

    if (msg!=null)
    for (int j = 0; j < filtro.size(); j++)
        if (msg.contains(filtro.get(j))) {
            devuelve=true;
            break;
        }
    return devuelve;

}


@Override
public void onNotificationPosted(StatusBarNotification sbn) {

    if (sbn.getNotification().tickerText!=null) {

        Bundle extras;
        extras = sbn.getNotification().extras;
```

```
String notificacion = String.valueOf(sbn.getNotification().tickerText);
switch (sbn.getPackageName()) {

    case "org.telegram.messenger":
        aplicacion = "telegram";
        id = sbn.getId();
        emisor = notificacion.substring(0, notificacion.indexOf(":"));
        mensaje = notificacion.substring(notificacion.indexOf(":") + 2);
        break;

    case "com.whatsapp":
        aplicacion = "WhatsApp";
        id = sbn.getId();
        emisor = sbn.getNotification().tickerText.toString().substring(11);
        mensaje = (String) sbn.getNotification().extras.get("android.text");
        break;

    case "com.facebook.orca":
        aplicacion = "Facebook Messenger";
        id = sbn.getId();
        emisor = notificacion.substring(0, notificacion.indexOf(":"));
        mensaje = String.valueOf(extras.getCharSequence("android.text"));
        break;

    case "com.android.messaging":
        aplicacion = "SMS";
        id = sbn.getId();
        emisor = extras.getString("android.title");
        mensaje = String.valueOf(extras.getCharSequence("android.text"));
        break;


    default:
        id = sbn.getId();
        aplicacion = sbn.getPackageName();
        emisor = "Desconocido";
        mensaje = String.valueOf(sbn.getNotification().tickerText);
}




if (filtrado(mensaje)) {

    MyDB admin = new MyDB(this, "administracion", null, 1);
    SQLiteDatabase bd = admin.getWritableDatabase(); //Create and/or open a
     database that will be used for reading and writing.
    //String dni = et1.getText().toString();


    Cursor fila = bd.rawQuery(  //devuelve 0 o 1 fila //es una consulta
            "select *  from notificaciones", null);

    if (!fila.moveToFirst())
        id=0;
    else
        id=fila.getCount();


    //String dni = et1.getText().toString();
```

```
                fila = bd.rawQuery(  //devuelve 0 o 1 fila //es una consulta
                        "select *  from notificaciones where id=" + id + " and aplicacion=
                         \'" + aplicacion + "\' and mensaje= \'" + mensaje + "\'", null);


                if (!fila.moveToFirst()) {  //si ha devuelto 1 fila, vamos al primero.
                    ContentValues registro = new ContentValues();  //Clase para guardar datos
                    registro.put("id", id);
                    registro.put("aplicacion", aplicacion);
                    registro.put("mensaje", mensaje);
                    registro.put("emisor", emisor);

                    bd.insert("notificaciones", null, registro);

                    checkLimit(aplicacion,emisor);


                    fila.close();
                    bd.close();

                } else {
                    Log.d(TAG, "No se ha podido añadir a la tabla"
                }
            }
        }
    }
```

A.2. Algoritmo del servicio de BulStapp Parent

```
/*******************************************************************************
 *
 * ServiceListener.java
 *
 *******************************************************************************
 *
 * Anael Eneas Melián Baute
 *
 * 5 de julio de 2016
 *
 * Comportamiento del servicio de BulStapp Parent. Cada vez que se llama al servicio se
 * crea un listener en Firebase y cuando este escucha algo se crea una notificación.
 *
 *
 *******************************************************************************/

static int id=0;

    private final String FIREBASE_URL = "https://simple-firebase.firebaseio.com/";
    private final String FIREBASE_CHILD = "contactos";
    private final String TAG = "BulStappParentService";
    private Firebase myFirebaseRef;

    private SharedPreferences settings;
```

```
ValueEventListener listenerValue=null;

void generarNotificacion(String key,String value){

    value=value.substring(1,value.length()-1);

    String[] val= value.split("[,]");
    String text="";

    Log.d(TAG+" 1", val[1]);
    Log.d(TAG+" 1", key);


    for (int i=0;i<val.length;i++){

        String parte= val[i];


        int index=parte.indexOf("=")+1;

        Log.d(TAG +" 2", parte.substring(index));
        Log.d(TAG +" 2", AESUtil.decrypt(parte.substring(index),key));


        if(i!=0)
            text= text + ",\n"+ parte.substring(0,index).trim()+ " "+
             AESUtil.decrypt(parte.substring(index),key);
        else
            text= parte.substring(0,index).trim()+ " "+
             AESUtil.decrypt(parte.substring(index),key);

    }
    Log.d(TAG +" 3", text);


    NotificationManager nManager =
     (NotificationManager)getSystemService(NOTIFICATION_SERVICE);
    Builder builder = new Builder(this)
            .setSmallIcon(android.R.drawable.ic_dialog_info)
            .setLargeIcon(BitmapFactory.decodeResource(getResources(),
            R.drawable.ic_cloud_white_48dp))
            .setContentTitle("ALERTA DE BULLYING")
            .setContentText(text)
            .setStyle(new NotificationCompat.BigTextStyle().bigText(text))
            .setWhen(System.currentTimeMillis());


    nManager.notify(id, builder.build());
    id++;

}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Firebase.setAndroidContext(this);

    if (listenerValue!=null)
        myFirebaseRef.removeEventListener(listenerValue);
```

```
    if (!settings.getString("numero","").equals("")) {

        myFirebaseRef = new Firebase(FIREBASE_URL).child(FIREBASE_CHILD).child(
                settings.getString("numero", ""));

        listenerValue = new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot snapshot) {
                if (snapshot.getValue()!=null) {
                    generarNotificacion(snapshot.getKey(), snapshot.getValue().toString());
                    System.out.println(snapshot.getValue());
                }
            }

            @Override
            public void onCancelled(FirebaseError error) {
            }
        };
        myFirebaseRef.addValueEventListener(listenerValue);

    }
    return super.onStartCommand(intent, flags, startId);
}
```