En caso de emergencia: descargue una app

Parte II

Asociación por los Derechos Civiles

Diciembre 2020
adc.org.ar

Investigación y redacción: Leandro Ucciferri y Celeste Gauna
Diseño y diagramación: El Maizal

En caso de emergencia: descargue una app – Parte II

En esta investigación profundizamos el análisis técnico de las principales apps móviles utilizadas en Argentina frente a la pandemia por COVID-19: Cuidar, a nivel nacional; las aplicaciones implementadas en Salta, Santa Fe, La Rioja, Jujuy, Tierra del Fuego, en el orden provincial, y en el ámbito municipal, la empleada en la Ciudad de Córdoba.

Introducción | 5

Metodología de análisis técnico | 8
- Conclusiones | **10**

Cuidar | 12
- Preguntas planteadas a la Secretaría de Innovación Pública | **14**
- Antecedentes y controversias | **18**
- Manejo del repositorio en GitHub | **22**
- Análisis de los dominios | **24**
- Método de autenticación y Registro de Usuario | **25**
- Esquema de firma utilizado | **27**
- Trackers de analiticas implementados | **29**

¿Cuál es la situación en las provincias? | 30

Provincia de Salta | 31
- Permisos solicitados | **31**
- Clase BuildConfig.java | **32**
- Trackers de analìticas implementados | **36**
- Registro de ubicación geográfica | **35**
- Método de autenticación y Registro de Usuario | **37**

Provincia de Santa Fe | 40
- Registro de ubicación geográfica | **40**
- Método de autenticaciòn y Registro de Usuario |**41**

Ciudad de Córdoba | 42
- Método de autenticación y Registro de Usuario | **42**
- Registro de ubicación geográfica | **43**

Provincia de La Rioja | 45
- Registro de ubicación geográfica | **45**
- Credenciales escritas en el código fuente | **45**
- Método de autenticación y Registro de Usuario | **46**

Provincia de Jujuy | 47
- Base de datos | **47**
- Configuración del tráfico de red | **49**
- Análisis de los dominios | **50**
- Método de autenticación y Registro de Usuario | **53**
- Solicitud de Permisos | **53**

Provincia de Tierra del Fuego | 55
- Credenciales escritas directo en el código fuente | **55**
- Método de autenticación y Registro de Usuario | **55**

Glosario | 58

Anexo
- Comparación de las aplicaciones analizadas | **62**

Introducción

Desde los primeros meses de 2020, un elemento constante en las discusiones sobre la contención de la pandemia por COVID-19, es el rol de las tecnologías digitales en las estrategias sanitarias de cada país. A mediados de marzo, comenzamos a monitorear los diversos desarrollos tecnológicos que iban surgiendo a lo largo del país, para entender los matices de las múltiples iniciativas, las limitaciones técnicas de su efectividad y los posibles impactos en los derechos humanos. Esto dio lugar a una **publicación sobre consideraciones mínimas** que se deben tener en cuenta para implementar tecnología como respuesta a la pandemia.

A finales de mayo publicamos **"En caso de emergencia: descargue una app"**, un análisis técnico preliminar sobre cinco aplicaciones, implementadas por la Ciudad de Córdoba y las provincias de Santa Fe, La Rioja, Tierra del Fuego y Jujuy. Analizando las funciones, los datos recolectados y los Términos y Condiciones de cada una de ellas, vimos que las finalidades perseguidas por estas aplicaciones se concentraban en una combinación de diversos propósitos. Brindar información sobre higiene y síntomas; validar permisos de circulación; controlar la circulación de personas; asistir a la persona si reporta síntomas; y realizar denuncias a personas (por incumplimiento de cuarentena) o comercios (por infracciones como vender a un precio superior al fijado por el Estado).

Esta investigación nos permitió arribar a una serie de conclusiones iniciales:

- El rol que ocupan esas apps en la estrategia sanitaria provincial no queda claro. Como tampoco si, antes o durante su desarrollo, se involucró a profesionales de la salud especializados.
- Salvo el caso de la app utilizada en Tierra del Fuego, las otras cuatro no establecen en sus Términos y Condiciones si los datos que se recolectan serán eliminados una vez que finalice la emergencia sanitaria.

- Los Términos y Condiciones de las cinco apps establecen cláusulas amplias para la cesión de datos recolectados entre organismos estatales, sin determinar específicamente quiénes tendrán acceso, de qué manera y por cuánto tiempo.
- Las cinco apps recolectan la ubicación geográfica, algunas de modo persistente (por ejemplo, en el caso de Tierra del Fuego, cada vez que se abre la app) y otras solo al momento de enviar el formulario de autodiagnóstico (como en el caso de Jujuy).
- Las cinco apps solicitan permisos excesivos del *smartphone*, que no son necesarios para cumplir con la finalidad perseguida. Por ejemplo: el uso de la ubicación geográfica en segundo plano, el acceso a archivos multimedia, o ver qué otras aplicaciones fueron abiertas.

Habiendo transcurrido más de cinco meses desde esta primera investigación exploratoria, nos planteamos profundizar el análisis técnico y solicitar información a los organismos estatales pertinentes. El objetivo fue dilucidar qué tan significativo fue el rol de las aplicaciones en la estrategia sanitaria, además de evidenciar posibles problemas técnicos en el desarrollo de las apps y el nivel de recolección de datos personales. Además de volver a revisar las cinco aplicaciones del anterior informe, trabajamos también sobre la app nacional Cuidar y la app de la provincia de Salta[1].

El análisis técnico sobre cada una de las apps comenzó con las primeras descargas de las mismas en septiembre, finalizando en noviembre. La última descarga de cada app disponible oficialmente fue realizada en octubre.

Tanto el Estado Nacional, como las provincias, no han empezado a usar las apps para realizar rastreo de contactos (*contact tracing*). Debido a que las apps no se utilizan con este fin, ninguna de ellas implementa el protocolo desarrollado por Apple y Google,

1 Un resumen comparativo de las apps analizadas se puede ver en el Anexo de este informe

denominado **"Exposure Notification"**, pensado bajo un modelo descentralizado, enfocado en preservar la privacidad.

El 22 de septiembre enviamos una serie de preguntas a los administradores de cada una de las apps correspondientes a las provincias de Santa Fe, Jujuy, Salta, Tierra del Fuego, La Rioja y la Ciudad de Córdoba. Para contactarnos utilizamos principalmente las direcciones de correo electrónico declaradas públicamente, ya sea en Google Play Store, en los sitios web institucionales o en los Términos y Condiciones de cada app:

- Provincia de Santa Fe:
  covidapp@santafe.gov.ar
- Provincia de Jujuy:
  coe@jujuy.gob.ar
- Provincia de Salta:
  agenciadeinformaciondepoliticaspublicas@salta.gob.ar
- Provincia de Tierra del Fuego:
  deis@tierradelfuego.gov.ar
  y tierradelfuego.gobierno@gmail.com
- Provincia de La Rioja:
  ministeriodesalud@larioja.gob.ar
- Ciudad de Córdoba:
  aplicaciones@cordoba.gov.ar

Solo recibimos respuesta inmediata en el caso de la Ciudad de Córdoba, cuyo Director de Sistemas nos puso en contacto directo con el Director de Modernización de la Municipalidad. Sin embargo, al igual que en el resto de los organismos públicos contactados, no recibimos respuesta.

Ante la falta de respuesta generalizada en las seis jurisdicciones, volvimos a enviar un email, a las mismas direcciones, el 2 de octubre.

Este último email, del 2/10, solo fue respondido por la Agencia de Información de Políticas Públicas de Salta, para confirmar que la solicitud estaba en proceso. Finalmente, el 13 de octubre recibimos la

respuesta formal a las preguntas enviadas, las cuales encontrarán en la sección correspondiente al análisis específico de dicha provincia.

El 19 de octubre reenviamos los emails originales a otras direcciones disponibles en los sitios web institucionales, correspondientes a Jujuy, Tierra del Fuego y La Rioja. Sin embargo, en el caso de Jujuy (info@gajujuy.gob.ar correspondiente a la Dirección Provincial de Transparencia y Gobierno Abierto) y Tierra del Fuego (jdangelo@tierradelfuego.gov.ar correspondiente a la Secretaría General, Legal y Técnica) las direcciones de email arrojaron errores para recibir el correo.

Al mismo tiempo, para los casos de Santa Fe, Jujuy y la Ciudad de Córdoba, también presentamos solicitudes formales de información pública mediante los formularios institucionales disponibles en sus sitios web. Al momento de la publicación de este informe, ninguna de las solicitudes fue contestada.

Respecto a la aplicación Cuidar, el 19 de octubre enviamos un email a la Secretaría de Innovación Pública para realizar una serie de preguntas en igual sentido que a las provincias, y proponer recomendaciones en base a los aspectos identificados como resultado del análisis. Tres semanas después, el 9 de noviembre, la Secretaría envió las respuestas a nuestras preguntas, las cuales encontrarán en la sección específica de la app.

Metodología de análisis técnico

La metodología utilizada para el análisis de las diferentes aplicaciones se basó principalmente en la realización de un ***análisis estático,*** es decir en la revisión del código fuente descompilado de la aplicación. Se optó por la distribución Kali Linux, montado en un entorno virtual de pruebas, junto con herramientas de ***pentesting*** *y análisis de seguridad*.

Se comenzó el análisis con una fase de reconocimiento, con

recolección de información y prueba de las funciones externas de la aplicación. Luego, se continuó con el análisis del binario descompilado utilizando técnicas de ***ingeniería inversa***, a partir del cual se pudo obtener información sobre permisos, actividades utilizadas, recursos, puntos de entrada, datos e información sensible inserta directo en el código fuente. Para la misma, se realizaron los siguientes pasos:

- Valiéndonos de la utilidad ***gplaydl***, se realizó la descarga de los paquetes .APK de cada una de las aplicaciones directamente de la tienda oficial en Google Play Store, para obtener la versión más actualizada.
- Cada uno de los archivos .APK descargado fue sometido a la realización de un análisis estático a través de la herramienta **MobSF (Mobile Security Framework)**, un entorno de análisis automatizado de *malware* y seguridad para aplicaciones móviles. Esta aplicación divide el resultado de la auditoría en diferentes categorías: información básica de la app, certificado y firmas, permisos, clases que implementan la API de Android, actividades, análisis de seguridad, análisis de malware, URLs, base de datos tipo Firebase, *trackers*, claves/contraseñas insertadas en el código fuente, servicios, librerías, etc.
- Sobre cada punto que identificamos como en posible riesgo, se realizó una inspección más detallada dentro del código, sobre todo de aquellos puntos relacionados a la implementación de permisos de geolocalización, claves/contraseñas insertadas directo en el código, y reglas de seguridad de base de datos.
- Para ello se utilizaron algunas herramientas de descompilado e ingenieria inversa, como **JADX** o **APKTool**, además de algunas solicitudes que se realizaron por línea de comandos sobre la terminal de Kali Linux.

Previo a la publicación del informe, se informó a las provincias de La Rioja, Jujuy y Tierra del Fuego, sobre los problemas de seguridad encontrados en sus respectivas aplicaciones. El envío de esta información fue realizado vía email, utilizando las mismas direcciones

que comentamos anteriormente. Al momento de la publicación de este informe no recibimos respuesta a los emails enviados.

Conclusiones

Previo a exponer las conclusiones que surgen como resultado de nuestro análisis, compartimos una serie de consideraciones que destacamos a raíz de nuestra experiencia en el proceso de investigación:

- A pesar de que en la mayoría de los casos se declaran emails de contacto, por ejemplo en Google Play Store o en los Términos y Condiciones, durante el proceso de investigación para este informe, salvo en dos casos puntuales, esos canales no resultaron útiles para entablar una conversación con los administradores de las apps y solicitar mayor información, ya que no brindaron ningún tipo de respuesta.
- Al momento de tener que reportar la existencia de errores o fallas que pueden comprometer la seguridad de los datos recolectados, no solo no existen procesos claros para brindar esa información, sino que además los Términos y Condiciones desalientan el trabajo de investigación en seguridad informática. En igual sentido al punto anterior, a través de los canales de comunicación disponibles públicamente tampoco se obtuvo una respuesta luego de informar sobre dichos problemas.
- La única aplicación de código abierto es Cuidar, con los reparos mencionados en su sección. Ninguna de las apps provinciales que se analizaron tienen su código publicado y, lo que es aún más grave, siguen manteniendo cláusulas que prohíben la ingeniería inversa en sus Términos y Condiciones.

El análisis técnico efectuado y la investigación sobre el uso de las apps por cada provincia, nos permite llegar a las siguientes conclusiones:

1. Todas las apps adoptaron un sistema centralizado para la recolección y procesamiento de datos, esto implica que los datos recolectados son procesados y almacenados en un mismo

servidor, desde el cual luego se accede a la información.

2. Si bien es cierto que, siendo un país federal, las provincias toman decisiones propias para enfrentar la pandemia de acuerdo con sus contextos, esto no debería ser sinónimo de descoordinación. Mediante esta investigación no se pudo determinar que exista una estrategia unificada, a nivel nacional, para el uso de la tecnología como respuesta a la pandemia. Cada provincia plantea sus propias finalidades, "estándares" y obligatoriedad. Situación que queda aún más en evidencia con la apertura de los vuelos de cabotaje y el inicio de la temporada de vacaciones, ante la cual cada provincia estableció sus propios requisitos de ingreso.

3. Esta investigación evidencia la rapidez con la que se mira a la tecnología sin una mirada crítica sobre sus limitaciones, a la vez que no se priorizan análisis previos con profesionales expertos en la materia. Desde la llegada de la pandemia al país, se han creado múltiples aplicaciones que, con el paso de solo un par de meses, se dejaron de utilizar y mantener. En este proceso se recolectaron datos personales sensibles (en especial por su vínculo a la salud) con un potencial impacto en la privacidad de miles de personas, que no responden a una lógica o estrategia clara, precisa y transparente.

4. El diseño de la mayoría de las aplicaciones analizadas supone que los dispositivos corresponden siempre a un único dueño, cuando en realidad existen múltiples situaciones en donde un mismo celular es utilizado por todo el grupo familiar o comunidad. Si una app no permite cerrar la sesión, o utilizarse sin necesidad de un usuario, se introduce una nueva barrera para su potencial efectividad.

Antes de implementar soluciones tecnológicas en la lucha contra la pandemia, es imprescindible que el Estado realice un análisis serio dentro del marco de la estrategia sanitaria, basado en criterios legítimos y equitativos, con políticas claras y transparentes que resulten en una mayor eficiencia de la protección de los datos. De otra manera,

se estarían introduciendo mecanismos de vigilancia innecesarios y desproporcionados, bajo el argumento de asistir a la salud pública, a riesgo de resultar en una vulneración de derechos fundamentales.

Cuidar

A finales de abril de 2020, la Secretaría de Innovación Pública, dependiente de la Jefatura de Gabinete de Ministros, relanzó su **aplicación móvil "COVID-19 Ministerio de Salud" bajo el nombre Cuidar**. Según sus Términos y Condiciones, "La Aplicación es una herramienta tecnológica que brinda información sobre diversos temas referentes a los síntomas y/o prevención del virus COVID-19 con la finalidad de ayudar a prevenir la propagación del virus, como así también erigirse en una fuente de información fidedigna para los Usuarios".

En su versión actual, Cuidar permite realizar un autodiagnóstico, con carácter de declaración jurada, que le informa a la persona si los síntomas que declaró son compatibles con **COVID-19**. De acuerdo a sus desarrolladores, las preguntas del autodiagnóstico se basan en la **definición de caso provista por el Ministerio de Salud**. Además, la app permite vincular el **Certificado Único Habilitante de Circulación** (CUHC), el cual le otorga a la persona la posibilidad de movilizarse por su ciudad, siempre y cuando el autodiagnóstico no arroje como resultado que tiene síntomas posibles por coronavirus. En caso de detectar sintomatología compatible con COVID-19, se envía la información al Comité Operativo de Emergencia Provincial de su jurisdicción, para que se contacten con la persona y le brinden atención médica.

El desarrollo de la aplicación **surgió como una solicitud del Estado a diversas empresas privadas**. La Secretaría de Innovación Pública ha aclarado que la app fue desarrollada en conjunto con el Ministerio de Ciencia y Tecnología de la Nación, la Fundación Sadosky, el Consejo Nacional de Investigaciones Científicas y

Técnicas (CONICET) y la Cámara de la Industria Argentina del Software (CESSI), que nuclea a las empresas Hexacta, Globant, G&L Group, C&S, QServices, GestiónIT, Intive, Finnegans y Faraday. Se complementó con el trabajo de Arsat, servicios brindados por Amazon Web Services, RedHat Argentina, Thinkly y Biodyn SAS.

Uno de los aspectos más discutidos, durante los primeros meses de su lanzamiento, fue si el uso de la app era obligatorio o voluntario. Después de **múltiples declaraciones contradictorias** por parte de **funcionarios públicos** a los medios (incluyendo el Presidente y el Jefe de Gabinete), la Secretaría de Innovación Pública incluyó un aviso en su página web mencionando que serían las provincias las que determinen su obligatoriedad para cada jurisdicción. En septiembre, **la Secretaría aclaró en una nueva publicación** que "no es de instalación obligatoria. No existe reglamentación alguna que instale la obligatoriedad de la aplicación. El Estado nacional recomienda su utilización optativa con el objetivo de facilitar el acceso al diagnóstico temprano y expeditar el contacto con el sistema de salud por parte de la población en el marco de la pandemia."

La **Disposición 1771/2020, de la Dirección Nacional de Migraciones** (DNM), establecía que todas las personas que hayan ingresado al país desde el 26 de marzo debían adherir a utilizar la app por un plazo mínimo de 14 días. Sin embargo, esto fue revertido con la **Resolución Conjunta 11/20**, del Ministerio de Salud y la DNM, que determina en el punto cuatro de su anexo:

"Se recomienda también registrarse en la APP CUIDAR, o en la APP ESPECÍFICA a partir de su implementación, según se trate de argentinos o extranjeros residentes, o extranjeros no residentes, respectivamente, o vía web, dentro de las 48 hs del arribo al país autoreportando síntomas cada 48 hs, con único fin de:

- autofavorecer el conocimiento de la situación de riesgo.
- autodiagnostico.
- sensibilización sobre recomendaciones sanitarias de

prevención y autocuidado.

NO será exigible la geolocalización de estas aplicaciones y se resguardarán los datos personales.”

Según la información provista por la Secretaría, Cuidar es utilizada por más de 7.5 millones de personas. Asimismo, a través de la app se informaron más de 37 mil autodiagnósticos con síntomas compatibles con COVID-19. Hoy, según los datos disponibles en Google Play, la versión para Android cuenta con más de 10 millones de descargas (encontrándose solo disponible para dispositivos con Android 5 o superior).

Preguntas planteadas a la Secretaría de Innovación Pública

A continuación, les compartimos las respuestas de la Secretaría a las preguntas planteadas por la ADC en el email del 19 de octubre, obtenidas el 9 de noviembre.

¿Cómo describen el éxito de la persecución de los objetivos planteados para la app, a más de seis meses de su lanzamiento?

–La aplicación Cuidar (en adelante la Aplicación) tiene aproximadamente 7 millones y medio de usuarios y usuarias en todo el país. Es una herramienta de autodiagnóstico de síntomas que además brinda información recomendada y se conecta con todo un sistema de seguimiento para los posibles casos positivos. Creemos que, a seis meses de su lanzamiento, fue aceptada por gran cantidad de personas ya que además de

posibilitar su autoevaluación de síntomas, les permite portar del Certificado Único Habilitante para Circular.
Asimismo, la Aplicación significó la federalización de la tecnología puesta en valor y beneficio de todos los Gobiernos Provinciales.

¿Cuáles son los indicadores (cualitativos y/o cuantitativos) con los que se mide el éxito de cada objetivo planteado para la app?

–La Aplicación se vincula con un sistema más amplio que articula la información que la misma recolecta con las áreas sanitarias encargadas del cuidado ante la emergencia, tanto del gobierno nacional como de los gobiernos provinciales. De este modo, Cuidar complementa y asiste las políticas de prevención y cuidado de la población y, en particular, brinda elementos e insumos concretos para la intervención sanitaria.
En particular, al completarse un autodiagnóstico compatible con COVID-19 a través de la aplicación, el usuario es contactado con el Comité de Emergencia Provincial (en adelante, COEP) de la jurisdicción en la que se encuentra. Esto permite que sea contactado por las autoridades sanitarias a la brevedad, favoreciendo la intervención temprana del caso, descongestionando así las líneas telefónicas de asistencia y otros dispositivos puestos a disposición de la emergencia.

Por lo expuesto, los indicadores cualitativos y cuantitativos por los que se miden los objetivos para los cuales fue creada la app son:

- Cantidad de usuarios de la misma
- Cantidad de autoevaluaciones
- Cantidad de autoevaluaciones positivas y derivadas

¿Cuenta la Secretaría de Innovación y/o la autoridad competente (como puede ser el Ministerio de Salud) con estadísticas o reportes generados que visualicen el despliegue de la aplicación, desde su lanzamiento hasta hoy? De ser así, brindar una copia de la información con los resultados.

–Cabe señalar que la Secretaría se encuentra en proceso de generar los análisis completos respecto del despliegue, utilización y demás estadísticas respecto la Aplicación los cuales serán publicados a la brevedad.

¿Según sus registros, cuál es el porcentaje de la población que utiliza la app? De ser posible, desagregar la información por provincia y localidad.

–Como expresamos, son más de 7 millones y medio los usuarios y usuarias de la aplicación Cuidar. Aproximadamente el 70% se concentra en la Provincia de Buenos Aires y Ciudad Autónoma de Buenos Aires, seguidos por Santa Fe, Córdoba y Mendoza, correspondiéndose también con la densidad poblacional de cada provincia.

Brindar el número con la cantidad de usuarios activos mensuales. Desagregado por provincias y localidades.

–Nos remitimos a lo expresado en la respuesta anterior.

¿Cuántas derivaciones a hospitales y/o centros de salud se iniciaron a partir del uso de la aplicación para reportar síntomas?

–Desde la Secretaría de Innovación Pública no contamos con este número debido a que, una vez que una persona informa síntomas compatibles con COVID-19 es atendido en cada jurisdicción y sigue el circuito sanitario decidido por esa autoridad.
Ahora bien, cabe señalar que la aplicación Cuidar realizó aproximadamente 37 mil autodiagnósticos compatibles con COVID-19 que fueron derivados a los organismos de salud pertinentes.

¿Con la información obtenida desde la aplicación, se elaboraron mapas de zonas de riesgo?

–Respecto de su consulta, señalamos que todo lo concerniente a la información sobre cantidad de casos, multiplicación de los mismos y demás información epidemiológica se encuentra exclusivamente bajo las funciones del Ministerio de Salud.

En octubre de 2020, **el Consejo de Europa publicó un informe** sobre protección de datos en el contexto de soluciones digitales al COVID-19. En este informe figura información brindada oficialmente por la Argentina respecto a las finalidades de Cuidar, donde se detalla (tabla 3, página 30): "autodiagnóstico; hacer cumplir medidas de aislamiento sanitario; control de multitudes; mapear patrones de viajes de la población; funcionar como pasaporte de inmunidad".

Al respecto, consultamos a la Secretaría: **¿Cuál fue el organismo que brindó las respuestas al cuestionario del Consejo de Europa enviado en mayo 2020? Debido a que algunas de las finalidades declaradas en el cuestionario no se encuentran detalladas en los T&C ¿Se plantea extender el uso de la aplicación para cubrir estos propósitos en un futuro inmediato?**

–Respecto de su consulta, señalamos que la referida respuesta no fue brindada por la Secretaría de Innovación Pública y, tal como fuera reflejada en la referida tabla, pareciera que es información que es interpretada por la información pública.
Asimismo, cabe subrayar que la aplicación Cuidar no funciona como pasaporte de inmunidad, ni mapea patrones de viajes, ni realiza control de multitudes. La función principal es el autodiagnóstico de síntomas y brindar información para hacer cumplir las medidas.
En virtud de lo requerido aclaramos que todas las funcionalidades actuales de la aplicación son las que se encuentran cubiertas por los Términos y Condiciones. No

se pretende por el momento extender las finalidades de la aplicación “más allá del trabajo de cara a la temporada de vacaciones”. En el caso de suceder, será debidamente comunicado, como también en la aplicación a cada usuario/a.

¿En el diseño de la aplicación se han contemplado estándares de accesibilidad digital para que la misma pueda ser utilizada por personas con discapacidad? Asimismo, ¿se han efectuado tests de la app para evaluar si diversas personas con discapacidad pueden utilizarla sin inconvenientes? En caso afirmativo, brindar una explicación detallada de dicho proceso y los resultados obtenidos.

–La aplicación Cuidar cuenta con un botón de derivación al servicio de videollamadas para personas sordas e hipoacúsicos. Además, desde la Secretaría de Innovación Pública articulamos permanentemente con la Agencia Nacional de Discapacidad de cara a la implementación de mejoras de todas las aplicaciones y sistemas para hacerlos más accesibles.

El botón referido dirige a una página web estatal sobre la **noticia del servicio**.

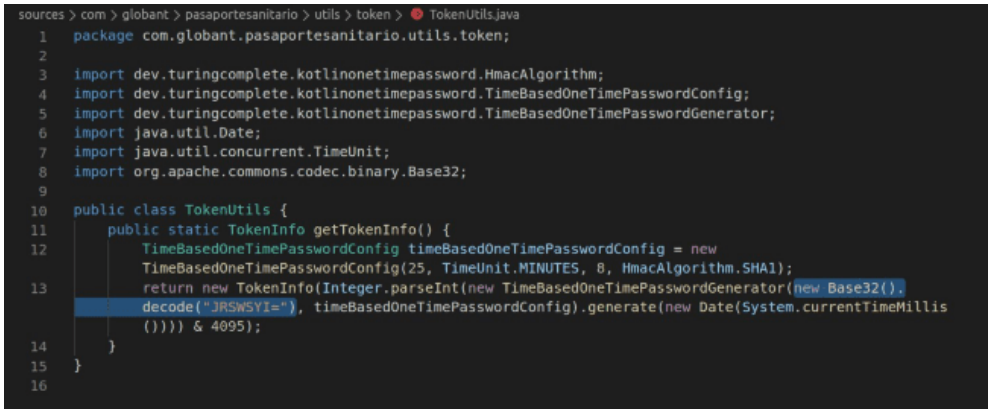
Antecedentes y controversias

En los meses posteriores al relanzamiento de la aplicación tuvieron lugar múltiples discusiones, tanto por su desarrollo y problemas técnicos, como también por los datos recolectados y las finalidades perseguidas.

La primera versión de la app tras el relanzamiento solicitaba múltiples permisos del teléfono, como el acceso a la geolocalización (aproximada y precisa), calendario, contactos, micrófono, cámara, acceso completo de red, configuración de audio, iniciarse cuando se prende el dispositivo y prevenir que el teléfono se duerma.

Asimismo, las principales discusiones públicas giraron en torno a la recolección de datos de ubicación geográfica, en primer y segundo plano, en este último caso con una frecuencia de quince minutos; la prohibición de realizar ingeniería inversa dispuesta en la primera versión de los términos y condiciones; y la necesidad de que el código fuente sea transparente. Al mismo tiempo, principalmente durante el mes de mayo, se denunciaron diversas vulnerabilidades y problemas de seguridad originados en el desarrollo de la app.

Expertos de la comunidad técnica compartieron, **principalmente por redes sociales**, que la app tendría una vulnerabilidad en



la generación del token de validación de un solo uso asociado al dispositivo, convirtiendo a la autenticación de dos factores absolutamente predecible.

La autenticación por contraseña de un solo uso (OTP), es un mecanismo de autenticación que utiliza una contraseña que sólo es válida para una única sesión de autenticación, es decir, no existe posibilidad de reutilizar una misma contraseña. En el caso Cuidar, para implementar este mecanismo en ese momento se valieron de un patrón especial denominado “Time based One Time Password” (TOTP), implementandolo con una l**ibrería de código abierto escrita en lenguaje Kotlin**.

Este tipo de algoritmo requiere de dos valores: un *token* o

clave secreta de inicialización, la cuál es de tipo simétrico y es conocida tanto por el servidor como por el usuario; y una variable, relacionada con el tiempo o la hora actual del sistema. Ambos valores pasan por una función de tipo *hash* conocida como "HMAC" y generan el *token* o patrón final (TOTP). Esta operación es realizada al mismo tiempo tanto en el servidor, como en el dispositivo móvil del usuario. Si ambos patrones resultantes son idénticos, la autenticación se valida.

Time-based One-time Password (TOTP)

The TOTP generator is available through the class `TimeBasedOneTimePasswordGenerator`. The constructor takes the shared secret and a configuration instance of the class `TimeBasedOneTimePasswordConfig` as arguments:

```
val secret = "Leia"
val config = TimeBasedOneTimePasswordConfig(codeDigits = 8,
                                            hmacAlgorithm = HmacAlgorithm.SHA1,
                                            timeStep = 30,
                                            timeStepUnit = TimeUnit.SECONDS)
val timeBasedOneTimePasswordGenerator = TimeBasedOneTimePasswordGenerator(secret.toByteArray(), config)
```

La vulnerabilidad descubierta se relaciona con la clave secreta de inicialización, la cuál, además de estar completamente expuesta

en el código, se pudo constatar que era idéntica a la que figura en repositorio oficial de la librería utilizada (*val secret=Leia)*

Decodificando la clave que aparece en el código (*'JRSWSYI='*) a Base32, podemos corroborar el resultado:

La clave secreta compartida ("Leia") parecía estar codificada (no cifrada) y fija (no aleatoria) del lado del usuario, de forma de generar siempre el mismo *token TOTP* para una hora determinada en el dispositivo, y lo mismo para todos los teléfonos que tenían instalada la aplicación. Es probable que esto haya sido utilizado para realizar la corroboración entre los teléfonos del personal de seguridad y las personas que transitaban con el permiso. Así, se validaba la credencial si se obtenía en ambos teléfonos el mismo

token TOTP para un momento determinado, asegurando que no se tratara de una posible captura de pantalla o imagen falsificada.

Esta librería se dejó de utilizar con las subsiguientes actualizaciones de la app, por lo que no se encontraba en la versión que fue descargada para el análisis de este informe.

También, comenzaron a crecer los **reclamos** en redes sociales por **fallos y problemas técnicos** vinculados con la tramitación del permiso para aquellos trabajadores esenciales. En ese momento, muchos usuarios manifestaron que la app no mostraba la autorización con el código QR para poder circular, siendo esenciales y con permisos en vigencia. Tampoco lograban acceder a la opción alternativa de tramitación mediante la página web, ya que la misma se encontraba caída.

Finalmente, una de las cuestiones que aún no ha sido resuelta, es que los Términos y Condiciones aparecen tildados (“aceptados”) por defecto, contrario a toda buena práctica sobre la prestación de consentimiento de las personas usuarias.

Con las siguientes actualizaciones de la app, algunos de los puntos mencionados en esta sección se fueron abordando. La cantidad y tipo de permisos se redujo, la recolección de la geolocalización se limitó

solo al momento de completar el autodiagnóstico si la persona habilitó el permiso, se quitó de los términos y condiciones la limitación para la realización de ingeniería inversa y revisión del código, y la Secretaría confirmó que publicaría el código fuente en Github.

Manejo del repositorio en GitHub

Desde la confirmación inicial, de parte de la Secretaría de Innovación, sobre la publicación del código fuente, se tardaron más de dos meses en actualizar el repositorio en Github con el código. A finales de julio se subió el código fuente del lado del cliente, tanto para la versión de **Android** como de **iOS**. Sin embargo, no se publicó el código del backend, es decir, del lado del servidor.

Conocer cómo funciona el backend permitiría analizar y transparentar efectivamente todo el circuito de recolección y tratamiento de datos personales. En otros países, como el caso de **España** y **Canadá**, han publicado los códigos fuente completos de sus apps, las cuales además se enfocan en una función más sensible que las ofrecidas por Cuidar, el rastreo de contactos digital.

Las mejores prácticas en materia de seguridad de la información indican que no se puede depender del secreto para proteger un sistema o sus componentes, tal como lo afirma el Instituto Nacional de Estándares y Tecnología de Estados Unidos en su ***Guía sobre la seguridad general del servidor***. Una buena parte de los riesgos inherentes al software abierto resultan mitigados por la disponibilidad del mismo, ya que existe mayor cantidad de programadores revisando el código, aumentando la posibilidad de encontrar y arreglar errores. Además, la experiencia demuestra que en muchos casos es el buen diseño de una pieza de software, centrado en la privacidad de las personas, lo que hace que un sistema adquiera mayor robustez en términos de seguridad.

Analizando el proyecto en Github, notamos que no está detallado

su propósito en ningún apartado. Al mismo tiempo, se identificaron múltiples reclamos y una desatención general respecto a cómo se lleva adelante la actualización de la información publicada.

Entre las recomendaciones compartidas con la Secretaría de Innovación vía email, sugerimos mejorar la claridad en el manejo del repositorio, precisando los fundamentos, objetivos y la metodología que se pretende aplicar con el mismo (criterios, roles, integración del código, etc).

Además, se sugirió considerar la posibilidad de poner a disposición del público una API de prueba (mock), para posibilitar la realización de análisis efectivos al código del frontend subido. Al mismo tiempo, la Secretaría también debería publicar el código del backend, donde se pueda auditar cómo se guardan los datos personales, si se cruzan con alguna otra información y bajo qué estándares de seguridad, etc.

También, para evitar una duplicación fraudulenta de la app y fortalecer la cadena de confianza, les sugerimos implementar ***Reproducible Builds***, mediante herramientas que realicen compilaciones deterministas como ***Bazel*** o ***Gitian***. Este proceso podría brindar los mecanismos necesarios para garantizar que los archivos binarios de compilación generados sean idénticos y puedan coincidir con el código fuente utilizado para compilarlos. Actualmente, existen varios proyectos de software libre que trabajan sobre esta estructura de desarrollo, y se pueden consultar aquí: ***"Who is involved?"***.

De la revisión del código en Github no surgió que exista documentación sobre el análisis, diseño o implementación del proceso de desarrollo de la aplicación Cuidar. En todo proyecto de desarrollo que pretenda ser transparente y abierto, es esencial

el rol de la documentación, ya que posibilita a los colaboradores aportar mejoras de forma más eficiente en caso de que fuera necesario, así como también ayudar a interpretar correctamente la estructura del código y su arquitectura. Estas recomendaciones también fueron extendidas a la Secretaría.

Análisis de los dominios

Realizando un escaneo de los diferentes dominios insertados en el código de la aplicación, nos encontramos con la siguiente URL, ubicada dentro de algunas clases ofuscadas: ***https://api.app.covid.ar***

Utilizando el comando host, se puede determinar que son dos las IPs asociadas a ese dominio: *75.2.14.245 y 99.83.205.212*, luego con el comando whois se puede verificar que las mismas estan registradas a nombre de *Amazon Web Services* (AWS). Luego, utilizando el comando *nslookup* sobre ambas IPs, podemos ver que apuntan al DNS: *a8852a2a614eba584.awsglobalaccelerator.com.*

AWS Global Accelerator es un servicio que permite realizar un enrutamiento de tráfico desde los usuarios hacia la red interna de AWS, para que los paquetes enviados ingresen a la red global de AWS lo más cerca posible del usuario, para mejorar el rendimiento y reducir latencias.

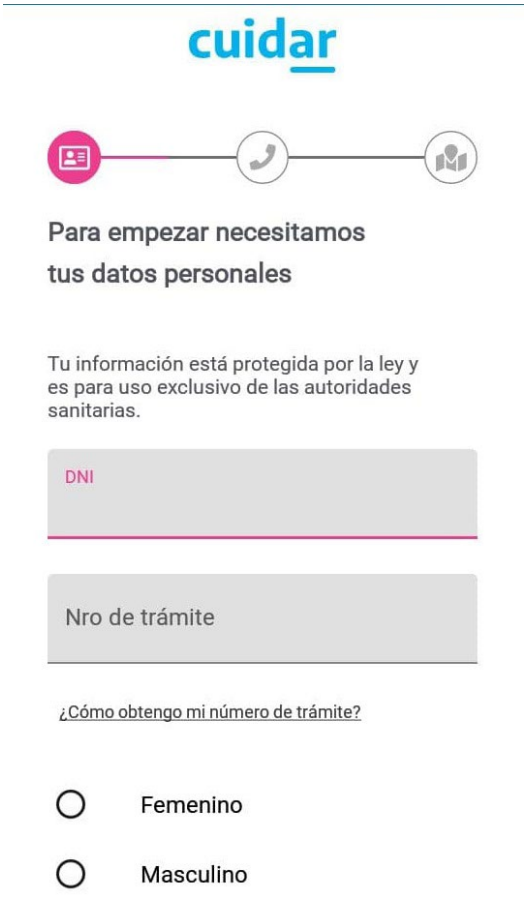
El uso de *Amazon Web Services* ya había sido transparentado por parte de la Secretaría de Innovación Pública.

Es importante mencionar que AWS se encuentra radicada en Estados Unidos, y que la Secretaría confirmó que allí se ubican los servidores donde se almacenan los datos de Cuidar, ya que según la regla general en la ley argentina de protección de datos personales, la transferencia internacional de datos a países considerados como “no adecuados” –como Estados Unidos– se encuentra prohibida (**artículo 12, Ley 25.326**). Aún así, existen distintas excepciones a esta regla.

El decreto reglamentario de la Ley 25.326 estipula que la prohibición del art. 12 no rige cuando el titular del dato haya prestado su consentimiento. Además, añade que el nivel de protección se considera adecuado “cuando dicha tutela se deriva (...) del amparo que establezcan las cláusulas contractuales que prevean la protección de datos personales”.

La **Disposición 60 - E/2016** establece una serie de cláusulas contractuales tipo para asegurar que la cesión internacional de datos se realice preservando la Ley 25.326, además de aclarar qué países son considerados con legislación adecuada.

Si bien la Secretaría de Innovación Pública ha confirmado que se firmó un contrato con AWS para la transferencia internacional de datos, aunque no ha publicado su contenido, es importante resaltar que la última versión de los **Términos y Condiciones (v5)** no incluye ninguna cláusula sobre la cesión o transferencia de los datos a Estados Unidos, o menciones a “Amazon”, “Amazon Web Services” o “AWS”.



Esta situación también fue remarcada por la Agencia de Acceso a la Información Pública en su informe sobre Cuidar publicado en julio, sugiriendo a la Secretaría la inclusión de esa información para cumplir con la obligación de informar establecida por la Ley 25.326.

Método de autenticación y Registro de Usuario

Al abrir la aplicación por primera vez, se abre una pantalla para registrarse, la cúal solicita tres

campos: Número de Documento Nacional de Identidad (DNI), Nro de Trámite y Sexo (“Femenino”, “Masculino”):

En el repositorio del código en Github se publicó una aclaración al respecto:

“Esta aplicación utiliza para su autenticación la combinación de DNI y número de trámite. Ese mecanismo es imperfecto, pero no es viable reemplazarlo sin incurrir en “alternativas” que resultan más invasivas, más “pesadas” o inviables en un contexto de pandemia donde es necesario desplegar rápidamente una aplicación que utilicen millones de ciudadanos y ciudadanas con diversa experiencia en tecnología”.

Consideramos que utilizar este mecanismo de autenticación, por combinación de DNI y Nro de trámite es altamente inseguro, ya que se están utilizando como clave/contraseña datos que se encuentran impresos en las tarjetas de los DNI. Esto también fue advertido por la Agencia de Acceso a la Información Pública, que recomendó utilizar un método independiente de ambos datos.

Al ser datos que se encuentran en el documento físico, el principal riesgo es que esa información puede llegar a ser obtenida por un tercero. La presentación del DNI para múltiples trámites se ha naturalizado en el día a día de la sociedad argentina. Algunos organismos y empresas incluso solicitan una fotocopia o una fotografía de ambos lados, e incluso se pueden **obtener directamente del código de barras del DNI**, lo que incrementa el riesgo sobre quién puede llegar a tener acceso a esos datos. A esto se le suman diversos incidentes de seguridad que han sufrido varios organismos estatales, donde estos mismos datos fueron expuestos. Por ejemplo, el caso de la **página web del Ministerio del Interior para el programa Procrear**.

El uso del DNI/Nro de trámite podría facilitar que se realice una suplantación de identidad, derivando en graves inconvenientes para el titular, ya que el autodiagnóstico es tomado con grado

de “declaración jurada”. Esta situación **ya ha empezado a ser reportada**, donde se utiliza a Cuidar como una vía de hostigamiento hacia la víctima, reportando síntomas falsos. Otro caso que fue difundido **involucró el DNI del Papa Francisco**, Jorge Bergoglio, debido a que circula una imagen de su documento con el código de barras, lo que le permitió a una persona generar un certificado y realizar el autodiagnóstico en su nombre.

Dentro de las sugerencias compartidas con la Secretaría de Innovación, incluímos actualizar el método de autenticación por uno más fuerte y confiable, como es el uso de la “Autenticación por múltiple factor”, añadiendo un mecanismo de recuperación y detección de acceso indebido (permitiendo la posibilidad de modificación).

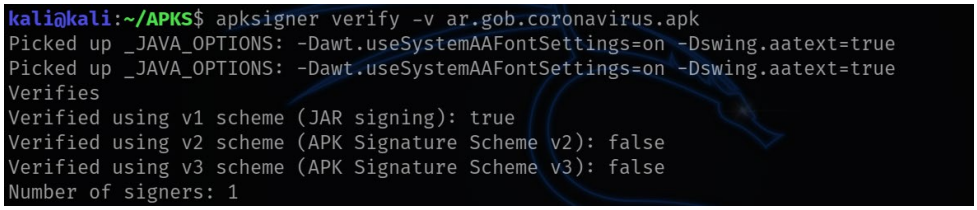
La respuesta oficial de la Secretaría fue la siguiente:

“Asimismo y con respecto al método de autenticación por combinación de DNI y Nro de Trámite, elegimos esta forma debido a que fue necesario el desarrollo de esta aplicación en tiempo record para brindar una herramienta en el marco de la pandemia que permitiera que personas de la más diversa experiencia tecnológica, con o sin correo electrónico u otros mecanismos alternativos de recuperación de contraseña, con o sin un teléfono de alta gama, pudieran usar la aplicación de manera simple, en un contexto donde se comenzó una gestión sin un mecanismo público de identidad digital pensado para la totalidad de la población. Estamos abocados desde todo el Estado nacional a poder brindar soluciones en este contexto. Por este motivo y debido a que esta necesidad surgió pocos meses después de asumir en la gestión, decidimos que se trataba de un método que posibilitaba la escalabilidad y rapidez de la herramienta”.

Esquema de firma utilizado

Antes de instalar cualquier aplicación, Android verifica que el

paquete APK se encuentra firmado con un certificado que identifica al autor de la app, cuya clave privada debe estar en manos del



desarrollador.

Android admite tres tipos de esquemas o formato de firmas para cada certificado: v1 (JAR), v2, introducida con Android 7, y v3, Introducida con Android 9.

Utilizando la herramienta *apksigner*, se detectó que la aplicación sólo implementa un esquema de firma tipo v1:

En 2017, un equipo de investigadores de la empresa *Guard Square*, descubrió una **vulnerabilidad denominada "Janus"** (CVE-2017-13156) sobre el esquema de tipo v1. Mediante la misma, es posible inyectar malware a las aplicaciones saltándose la verificación, es decir, sin afectar la firma, especialmente para sistemas Android que están por debajo de la versión 7.

El parche para esta falla de seguridad fue publicado e informado en diciembre del mismo año en el **boletín oficial de Android**. Sin embargo, en 2019, surgió un malware denominado "Agente Smith", que explotaba esta vulnerabilidad. Este malware afectó alrededor de 25 millones de dispositivos, dejando en evidencia la gran cantidad de sistemas que no cuentan con las respectivas actualizaciones de seguridad.

Esta situación se agrava debido a la fragmentación dentro del ecosistema de Android, por los tiempos que cada fabricante maneja para distribuir y habilitar las actualizaciones en sus dispositivos (sumado a que los usuarios pueden ignorarlas) y la práctica de descargar APKs desde páginas web que no informan sobre la autenticidad o el origen de esas aplicaciones.

Es por esto que una buena parte de la solución a esta problemática depende de implementar un mecanismo de firma más robusto, durante el desarrollo de una aplicación, como es el caso de v2 y v3.

```
//Otras
implementation "com.github.bumptech.glide:glide:$glideVersion"
implementation "com.jakewharton.timber:timber:$timberVersion"
implementation "net.zetetic:android-database-sqlcipher:$sqlCipherVersion"
implementation "commons-codec:commons-codec:$commonsCodecVersion"
implementation "com.newrelic.agent.android:android-agent:$newRelicVersion"
implementation "com.google.zxing:core:$zxingVersion"
implementation("com.journeyapps:zxing-android-embedded:$zxingEmbeddedVersion") { transitive = false }
```

En las recomendaciones extendidas a la Secretaría de Innovación incluimos la revisión de las firmas y la implementación de los tres esquemas juntos (v1, v2 y v3), como también se recomienda en la referencia de seguridad del Android Open Source Project (AOSP).

Trackers de analiticas implementados

Dentro de las dependencias del código de la app se encontró el empleo de un agente de monitorización externo perteneciente a la empresa *New Relic*.

New Relic es una compañía radicada en San Francisco, California, fundada en 2008, dedicada a brindar servicios y soluciones de software orientado a la supervisión y monitorización de sitios web y aplicaciones móviles.

El uso de este tracker de analíticas no se ha aclarado en el repositorio en *Github*, no se ha justificado o mencionado la necesidad de su implementación, o por qué se eligió por encima de herramientas similares.

De un análisis del tráfico enviado por la app al servidor, pudimos revisar cuáles son los datos que se envían en el primer momento en que se abre la aplicación, entre los que se encuentran la versión del sistema operativo, el modelo y fabricante del dispositivo; luego, al cierre de la app, vuelven a enviarse esos datos junto con el tiempo que estuvo abierta.

En el email a la Secretaría incluímos recomendaciones para transparentar cuál es el objetivo, el fundamento y el código referidos a la utilización de este *tracker* externo, además de aclarar cuál es la información que efectivamente se recolecta.

¿Cuál es la situación en las provincias?

En nuestro informe anterior ya habíamos resumido los aspectos centrales de las apps de las provincias de Santa Fe, La Rioja, Jujuy, Tierra del Fuego y la Ciudad de Córdoba, por lo que en esta sección vamos a enfocarnos en profundizar los aspectos que identificamos como más graves o importantes respecto a dichas apps. Asimismo,

en esta oportunidad sumamos el análisis de la app de Salta.

Según se **informó a fines de octubre**, con la flexibilización de ciertas medidas para el tránsito de personas entre provincias, las diversas jurisdicciones del país empezaron a establecer sus propios criterios para el ingreso a sus territorios.

De acuerdo con la información disponible más actualizada (al 6 de noviembre), las provincias de Córdoba, Jujuy, Neuquén, San Juan (dando la opción de presentar el CUHC solo en papel) y Tucumán mencionan específicamente el uso de la aplicación Cuidar. Mientras que las provincias de Santa Fe, Salta y Santiago del Estero mencionan sus apps propias.

Provincia de Salta

A principios de junio, el gobierno provincial presentó la aplicación "**SaltaCovid**", administrada por el Comité Operativo de Emergencia. El **anuncio oficial** destaca que la app "tiene como función principal indagar sobre síntomas de los ciudadanos. Además el objetivo es realizar un control y seguimiento a través de la geolocalización, de todas aquellas personas que ingresen a la provincia, ya que en todos estos casos, el uso de la app será obligatorio".

La provincia también dispuso una app orientada a los comercios, denominada "**SCAN Commerce**", para que registren los ingresos de sus clientes escaneando los datos de su DNI. Para el análisis técnico de este informe nos enfocamos solo en la app principal orientada al uso por cada habitante de la provincia.

Al momento de publicar este informe, "SaltaCovid" cuenta con más de 50 mil descargas en Google Play, disponible para smartphones con Android 5.0 en adelante.

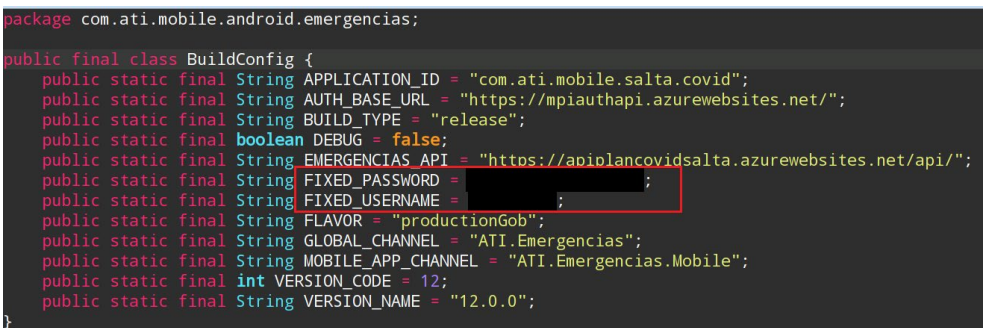
Permisos solicitados

La descripción de los permisos que brinda Google Play es la siguiente:

- Ubicación
  - acceder a tu ubicación aproximada (basada en red)
  - acceder a tu ubicación precisa (basada en red y GPS)
- Información de la conexión Wi-Fi
  - Ver redes Wi-Fi
- Otro motivo
- Recibir datos de internet
  - Ver conexiones de red
  - Tener acceso completo a la red
  - Ejecutarse al inicio
  - Impedir que el teléfono entre en modo de suspensión

Esta descripción se condice con el listado de permisos declarados en AndroidManifest.xml:

- android.permission.ACCESS_FINE_LOCATION
- android.permission.ACCESS_COARSE_LOCATION
- android.permission.ACCESS_NETWORK_STATE
- android.permission.ACCESS_WIFI_STATE
- android.permission.FOREGROUND_SERVICE
- android.permission.INTERNET
- android.permission.WAKE_LOCK
- android.permission.RECEIVE_BOOT_COMPLETED
- com.google.android.c2dm.permission.RECEIVE
- com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE



```
package com.ati.mobile.android.emergencias;

public final class BuildConfig {
    public static final String APPLICATION_ID = "com.ati.mobile.salta.covid";
    public static final String AUTH_BASE_URL = "https://mpiauthapi.azurewebsites.net/";
    public static final String BUILD_TYPE = "release";
    public static final boolean DEBUG = false;
    public static final String EMERGENCIAS_API = "https://apiplancovidsalta.azurewebsites.net/api/";
    public static final String FIXED_PASSWORD = [redacted];
    public static final String FIXED_USERNAME = [redacted];
    public static final String FLAVOR = "productionGob";
    public static final String GLOBAL_CHANNEL = "ATI.Emergencias";
    public static final String MOBILE_APP_CHANNEL = "ATI.Emergencias.Mobile";
    public static final int VERSION_CODE = 12;
    public static final String VERSION_NAME = "12.0.0";
}
```

Clase BuildConfig.java

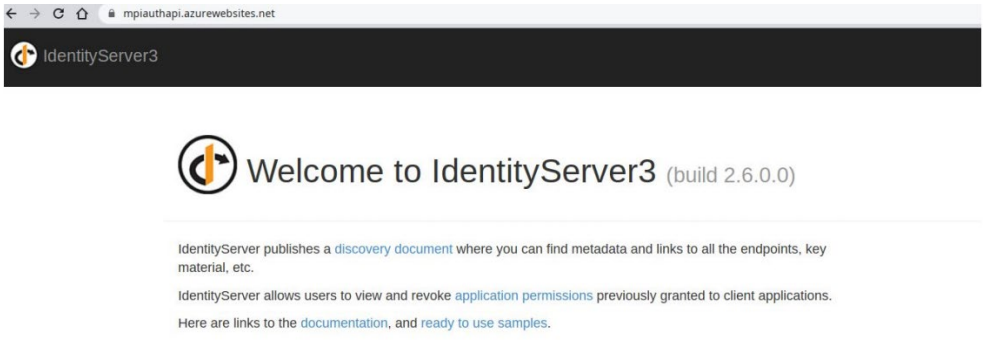
La clase 'BuildConfig' es una clase generada automáticamente en tiempo de compilación, cuyo contenido suele estar relacionado a diversas constantes de configuración específicas de una aplicación, como por ejemplo: *Id de la app*, versión, *Debug*, *flavor*, etc.

```java
package com.ati.mobile.android.emergencias;

public final class BuildConfig {
    public static final String APPLICATION_ID = "com.ati.mobile.salta.covid";
    public static final String AU = "https://mpiauthapi.azurewebsites.net/";
    public static final String BUILD_TYPE = "release";
    public static final boolean DEBUG = false;
    public static final String EA = "https://apiplancovidsalta.azurewebsites.net/api/";
    public static final String FLAVOR = "productionGob";
    public static final String GLOBAL_CHANNEL = "ATI.Emergencias";
    public static final String MOBILE_APP_CHANNEL = "ATI.Emergencias.Mobile";
    public static final int VERSION_CODE = 13;
    public static final String VERSION_NAME = "13.0.0";
}
```

Puede ocurrir que en el proceso de programación se dejen algunos campos estáticos que son utilizados en tiempo de ejecución, como pueden ser las credenciales de acceso para almacenar o leer información de un servidor.

En el mes de septiembre, se visibilizó en **redes sociales** una vulnerabilidad encontrada dentro del código de esta clase, al dejar el usuario y contraseña de acceso a la API directo en el código



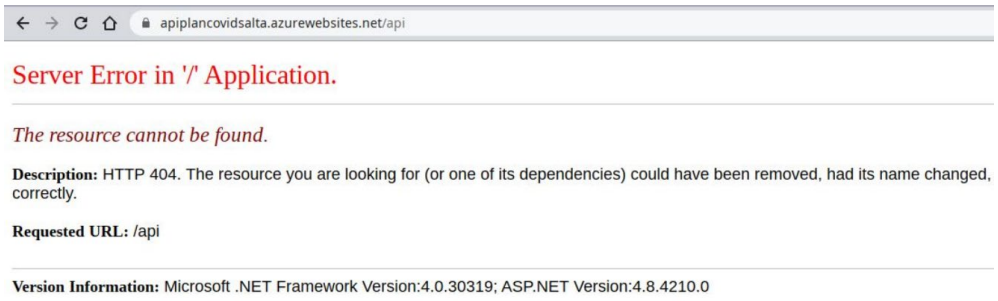
fuente de la app.

Al momento del análisis técnico que realizamos con una versión

```
public void login(String str, String str2, final LoginCallback loginCallback) {
    AndroidNetworking.post("https://mpiauthapi.azurewebsites.net/[redacted]") addUrlEncodeFormBodyParameter("gran
    addUrlEncodeFormBodyParameter("userName", str).addUrlEncodeFormBodyParameter("password", str2).
```

actualizada de la aplicación, se pudo observar que las claves escritas directamente en el código de esta clase, y en todo el código decompilado, fueron removidas.

Las direcciones: ***https://mpiauthapi.azurewebsites.net/ y https://apiplancovidsalta.azurewebsites.net/api***, que están dentro de esta clase, también fueron inspeccionadas.



La primera URL nos derivó a una página relacionada a “Identity Server”, versión 3; un servidor de autenticación de código abierto, utilizado para realizar un inicio de sesión único para diferentes tipos de aplicaciones.

Según lo que se puede constatar en el código, se realiza por única vez un request de autenticación tipo POST a esta dirección con una ruta alternativa:

Es muy probable que el sistema utilice primero un mecanismo de autenticación contra este servidor para obtener un token de validación, que luego será utilizado para obtener acceso a la API (***https://apiplancovidsalta.azurewebsites.net/api***) para realizar las diferentes consultas.

Si intentamos ingresar a la segunda url nos devuelve un código de error de tipo 404:

Es posible que este error se deba a que la url se encuentra incompleta, y se requiera especificar una ruta diferente.

Trackers de analìticas implementados

En la solicitud de información presentada ante los administradores de la app preguntamos si la misma utiliza alguna herramienta de analíticas y, en caso afirmativo, cuál, con qué finalidad y dónde se almacena la información recolectada.

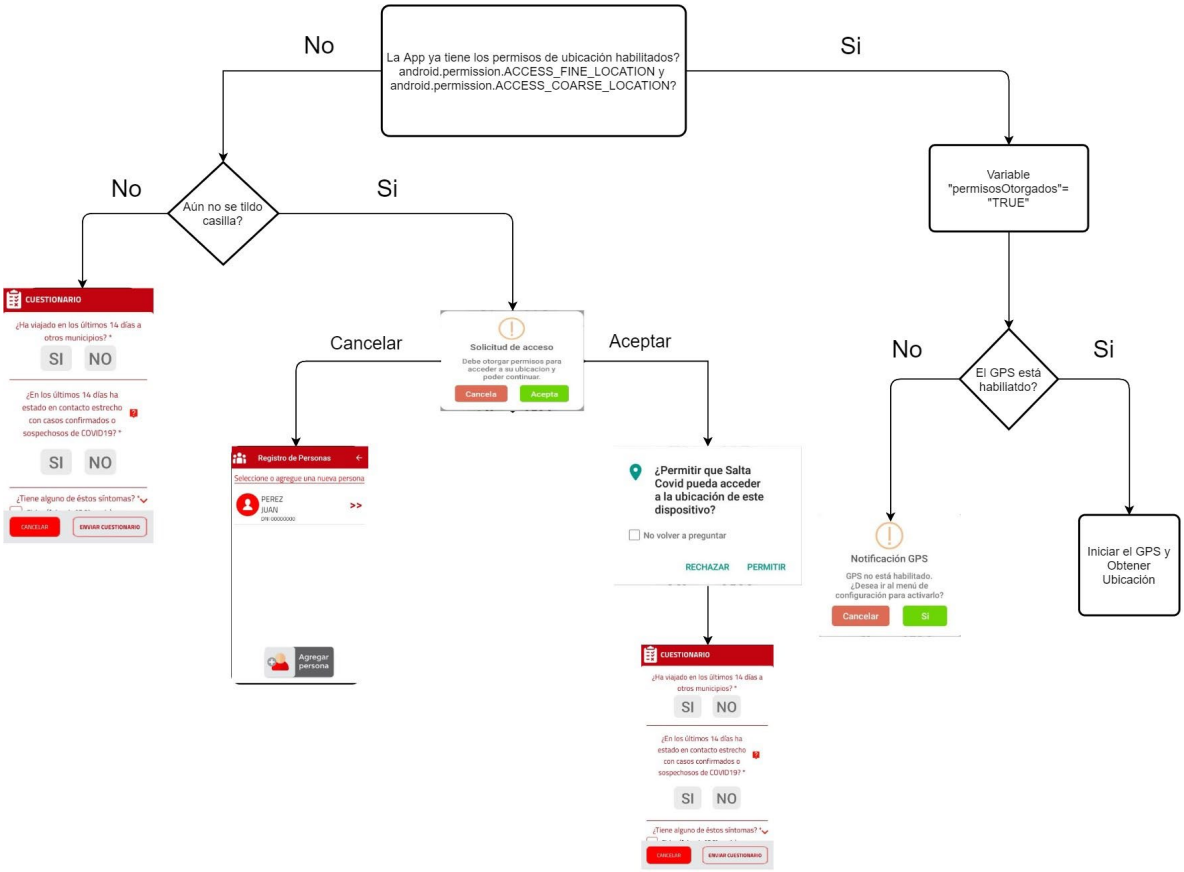
La respuesta brindada informa que “La aplicación no utiliza herramientas de analíticas.”, sin embargo, en el análisis técnico se encontraron dos herramientas que provee Google:

- CrashLytics
- Firebase Analytics

El uso de estos trackers no se encuentra declarado en los Términos y Condiciones.

Registro de ubicación geográfica

El permiso de geolocalización se solicita por primera vez para rellenar el campo del domicilio o residencia actual. Allí se informa que se puede completar a mano, en caso de no querer habilitar el permiso, lo que implica brindar una ubicación manualmente en el



mapa (aunque puede ser no coincidente con la ubicación que se detalla en el formulario).

```
private void checkGpsPermission() {
    if (ContextCompat.checkSelfPermission(this, "android.permission.ACCESS_FINE_LOCATION") == 0 && ContextCompat.checkSelfPermission(this, "android.permission.ACCESS_COARSE_LOCATION") == 0) {
        this.permisosOtorgados = true;
        Log.d(TAG, "GPS Permission has already been granted");
        if (this.haveAddress && this.lat == null && this.lng == null) {
            this.lat = this.address.getLatitude();
            this.lng = this.address.getLongitude();
            return;
        }
        return;
    }
    Log.d(TAG, "No tiene permisos para usar el GPS... Solicitando...");
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, "android.permission.ACCESS_FINE_LOCATION")) {
        new SweetAlertDialog(this, 3).setTitleText("Solicitud de acceso").setContentText("El acceso a la ubicación del dispositivo permitirá una localización mas precisa para la aplicación. De todas maneras
            /* class com.ati.mobile.android.emergencias.views.addresses.AddressActivity.AnonymousClass2 */

            @Override // cn.pedant.SweetAlert.SweetAlertDialog.OnSweetClickListener
            public void onClick(SweetAlertDialog sweetAlertDialog) {
                sweetAlertDialog.dismissWithAnimation();
                ActivityCompat.requestPermissions(AddressActivity.this, new String[]{"android.permission.ACCESS_FINE_LOCATION", "android.permission.ACCESS_COARSE_LOCATION"}, 100);
            }
        }).setCancelText("Aceptar").showCancelButton(true).setCancelClickListener(new SweetAlertDialog.OnSweetClickListener() {
            /* class com.ati.mobile.android.emergencias.views.addresses.AddressActivity.AnonymousClass1 */

            @Override // cn.pedant.SweetAlert.SweetAlertDialog.OnSweetClickListener
            public void onClick(SweetAlertDialog sweetAlertDialog) {
                sweetAlertDialog.dismissWithAnimation();
            }
        }).show();
    } else {
        ActivityCompat.requestPermissions(this, new String[]{"android.permission.ACCESS_FINE_LOCATION", "android.permission.ACCESS_COARSE_LOCATION"}, 100);
    }
}
```

Luego, existe una segunda solicitud antes de iniciar el autodiagnóstico y es menos clara. Se le informa a la persona que se deben otorgar permisos para poder continuar, y solo aparecen dos botones: Aceptar y Cancelar. Al hacer clic en “Cancelar”, se retorna a la pantalla principal y no se permite acceder al cuestionario. Al hacer clic en “Aceptar”, aparece una ventana “ActivityCompact”, preguntando si le otorga permisos a la app para acceder al dispositivo.

En caso de que se hiciera clic en “rechazar” aun se podría continuar realizando el autodiagnóstico. En el siguiente diagrama de flujo se puede visualizar este recorrido:

Dentro del paquete *‘com.ati.mobile.android.emergencias.views.test’* encontramos su implementación:

En la solicitud de información a la provincia, informaron que “ninguno de los permisos son obligatorios”. Respecto al uso de geolocalización, explicaron que “sólo lo pide una sola vez al registrar donde pasará los días en caso de no quedar internado o dónde buscarlo en caso de requerir una ambulancia. Utiliza la localización para dar una referencia al Mapa, luego el usuario puede colocar cualquier dirección. Es oportuno aclarar, que conforme las características de nuestro territorio, muchas zonas de la Provincia, sean urbanas o rurales, carecen de los parámetros urbanísticos de calle y numeración que permita la fácil ubicación de una vivienda, resultando indispensable el uso de la ubicación para el caso de requerir asistencia en su domicilio, villa, asentamiento o campamento donde se encuentre”.

La respuesta oficial brindada

no se condice con el análisis técnico que realizamos sobre la implementación del permiso de geolocalización. Al respecto cabe notar dos cuestiones. En primer lugar, se solicita nuevamente un permiso de ubicación, cuando ya ha sido otorgado previamente al realizar el registro inicial. En segundo lugar, no se informa claramente al usuario sobre la obligatoriedad de otorgar estos permisos para realizar el autodiagnóstico.

Método de autenticación y Registro de Usuario

```
public void onClick(View view) {
    switch (view.getId()) {
        case R.id.btn_ap_back /*{ENCODED_INT: 2131296336}*/:
            finish();
            return;
        case R.id.btn_ap_save /*{ENCODED_INT: 2131296337}*/:
            if (validateForm()) {
                if (this.objPerson == null) {
                    this.objPerson = new Person();
                }
                this.objPerson.setName(this.txtFirtName.getText().toString());
                this.objPerson.setSurname(this.txtLastName.getText().toString());
                this.objPerson.setDocumentTypeId(((SpinnerOption) this.ddlDocumentType.getSelectedItem()).getCodigo());
                this.objPerson.setDocumentNumber(this.txtDocumentNumber.getText().toString());
                this.objPerson.setAge(this.txtAge.getText().toString());
                this.objPerson.setGenderId(this.codSex);
                this.objPerson.setNationalityId(this.codNationality);
                this.objPerson.setOriginCountryId(((SpinnerOption) this.ddlCountryOrigin.getSelectedItem()).getCodigo());
                this.objPerson.setHealthCoverageId(this.codHealthCoverage);
                PersonRepository.getInstance().addOrEdit(this.objPerson, true);
                Intent intent = new Intent();
                intent.putExtra(Person.PERSON_ID_EXTRA, this.objPerson.getId());
                setResult(-1, intent);
                finish();
                return;
            }
            new SweetAlertDialog(this.myContext, 1).setTitleText("Notificación").setContentText("Debe completar los campos obligatorios.").show();
            return;
        case R.id.tipo_documento_text /*{ENCODED_INT: 2131296688}*/:
            this.ddlDocumentType.performClick();
            return;
        default:
            return;
    }
}
```

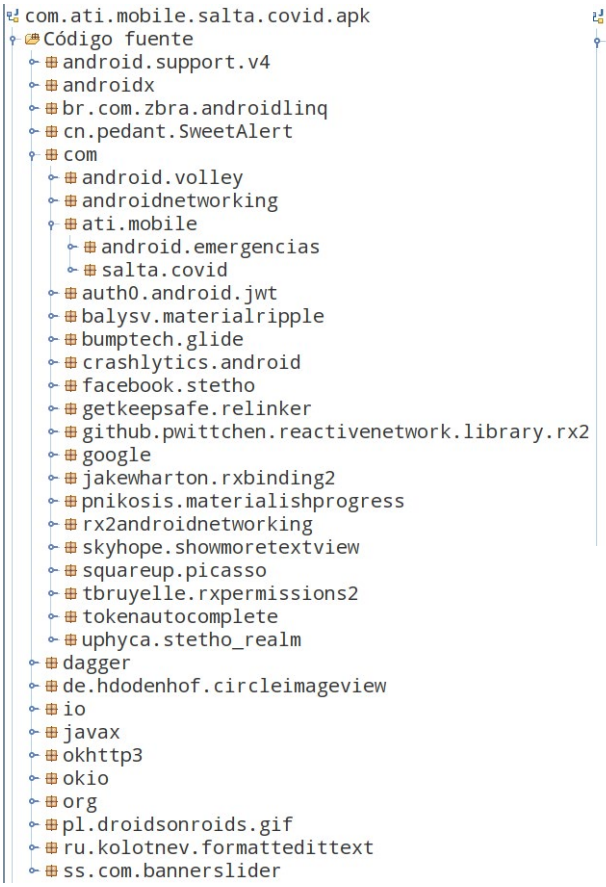
```
private boolean validateForm() {
    if (((SpinnerOption) this.ddlDocumentType.getSelectedItem()).getCodigo().isEmpty()) {
        return false;
    }
    if (findViewById(R.id.pnl_avp_document_number).getVisibility() != 0 || !this.txtDocumentNumber.getText().toString().trim().isEmpty()) {
        return true;
    }
    return false;
}
```

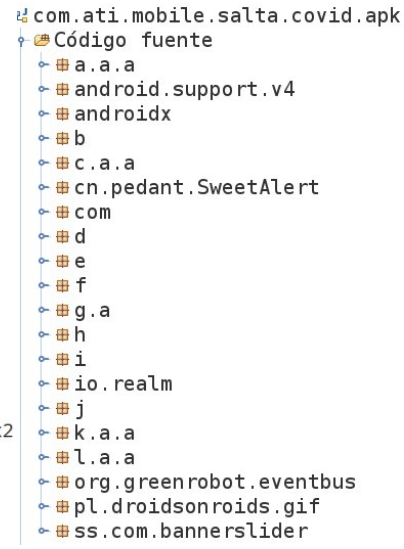
```
private void validatePersonLocal() {
    PersonRepository.getInstance().getByDocumentNumber(this.txtDocumentNumber.getText().toString(), new BaseCallback<Person>() {
        /* class com.ati.mobile.android.emergencias.views.validate_person.ValidatePerson.AnonymousClass2 */

        public void onResult(Person person) {
            super.onResult((Object) person);
            if (person != null) {
                ValidatePerson.this.txtDocumentNumber.setText("");
                new SweetAlertDialog(ValidatePerson.this, 3).setTitleText("Notificación").setContentText("Ya existe una persona registrada con este Número de Documento.").show();
            } else if (((SpinnerOption) ValidatePerson.this.ddlDocumentType.getSelectedItem()).getCodigo().equals("ID_INDOCUMENTADO")) {
                String codigo = ((SpinnerOption) ValidatePerson.this.ddlDocumentType.getSelectedItem()).getCodigo();
                Intent intent = new Intent();
                intent.putExtra(Person.PERSON_ID_EXTRA, "-1");
                intent.putExtra(DocumentType.DOCUMENT_TYPE_ID_EXTRA, codigo);
                intent.putExtra("DocumentNumber", "");
                ValidatePerson.this.setResult(-1, intent);
                ValidatePerson.this.finish();
            } else {
                ValidatePerson.this.validatePerson();
            }
        }
    });
}
```

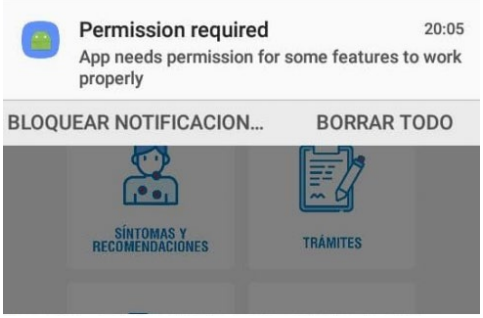
En la primera pantalla (“Validación de Identidad”), la app realiza el siguiente proceso sobre el campo del DNI:

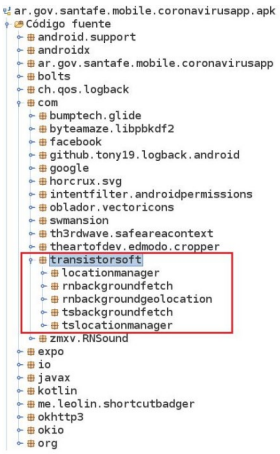
Si se selecciona desde la lista desplegable como tipo de documento





“DNI”, se permite ingresar cualquier número de DNI entre 1 y 8 dígitos (“00000000”, “11111111”, “1234”, etc) antes de pasar a la siguiente pantalla, “Registro de Personas”. De esta manera, se podría realizar una falsificación o suplantación de identidad. Este y otros datos, son tomados para registrar el autodiagnóstico con carácter de DDJJ.





Cuando el usuario llama al evento que se activa haciendo clic en el botón *Validar,* se realiza sólo la validación del campo relacionado con el número de documento verificando que este no sea un campo vacío (“*validateForm()*”) y luego, en otro método (“*validatePersonLocal()*”), se verifica que el número de DNI no haya sido registrado previamente.

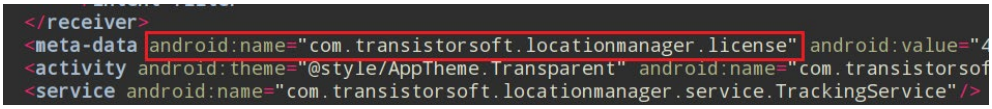
En la descarga más reciente del archivo .apk, notamos que la gran mayoría del código se encuentra ofuscado, es decir, ha pasado por un proceso que modifica su lectura y dificulta su auditoría.

Provincia de Santa Fe

Registro de ubicación geográfica

Al abrir la aplicación se le solicita al usuario que acepte los

permisos de ubicación en el dispositivo. Si el usuario se niega, puede seguir navegando por la app, sin embargo, al momento de dejar la aplicación en segundo plano, se muestra una notificación



periódica e insistente avisando que se requieren permisos para trabajar correctamente, aún a pesar de tildar sobre la opción “no volver a preguntar”.

Dentro del archivo *AndroidManifest.xml*, y en la lista de paquetes y dependencias utilizadas, nos encontramos con la referencia a un *plugin* (complemento) desarrollado por *React Native,* llamado ***React Native Background Geolocation***. Esta librería permite acceder

a la geolocalización del dispositivo en segundo plano, es decir, mientras la aplicación no está siendo activamente usada por la persona usuaria, incluso, se puede configurar para que se active automáticamente el rastreo cuando se reinicia el dispositivo o se sale de la aplicación. También utiliza la detección de movimiento para activar la geolocalización sólo, cuando el dispositivo se encuentre en movimiento y no en reposo ,con la idea de hacer un uso más eficiente de la batería.

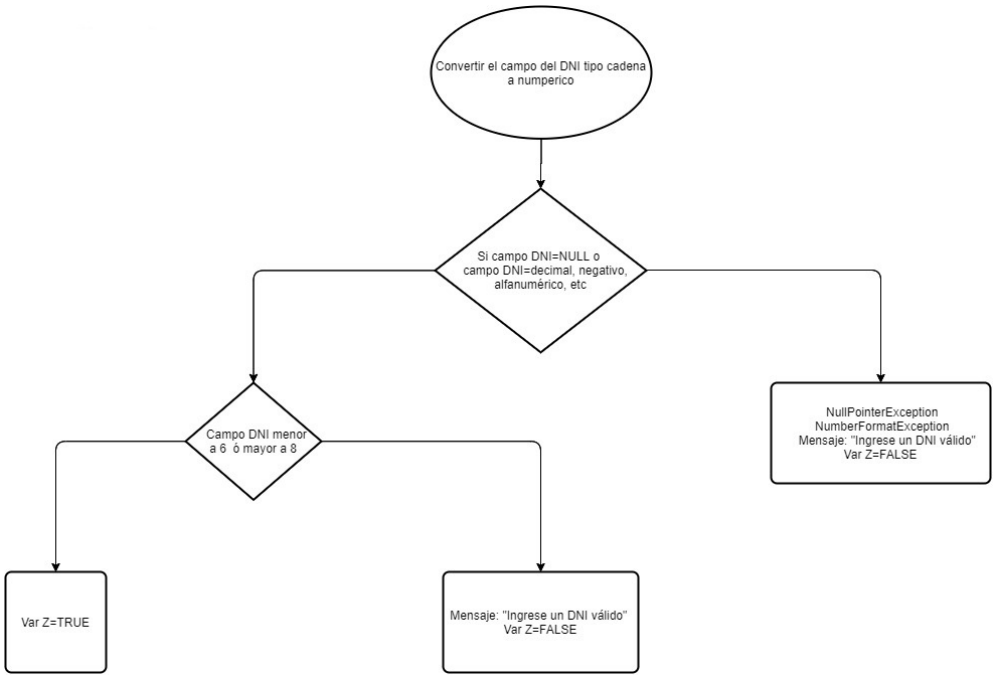
Dada la gran extensión de código decompilado y su nivel de ofuscación, no fue posible ahondar mucho más en el corto plazo sobre la implementación de esta herramienta dentro de la aplicación.

Método de autenticación y Registro de Usuario

La aplicación solo permite ingresar y registrarse mediante una cuenta de Google, por ejemplo la que ya esté configurada para Android. Asimismo, tampoco se permite cerrar sesión, ni visualizar

```
private boolean todosLosCamposSonValidos() {
    boolean z;
    boolean z2;
    boolean z3;
    boolean z4;
    boolean z5;
    boolean z6;
    if (this.et_dni.getText().toString().length() < 6 || this.et_dni.getText().toString().length() > 8) {
        this.et_dni.setError("Ingrese DNI válido");
        z = false;
    } else {
        z = true;
    }
    try {
        Long.parseLong(this.et_dni.getText().toString());
    } catch (NullPointerException | NumberFormatException unused) {
        this.et_dni.setError("Ingrese DNI válido.");
        z = false;
```

correctamente la información sobre la sesión iniciada. Esto es una característica común en la mayoría de las aplicaciones analizadas.



Ciudad de Córdoba

Método de autenticación y Registro de Usuario

Cuando se ingresa a la opción de "Autochequeo", aparece un formulario para rellenar con algunos datos personales como DNI, nombre, apellido, etc. El número de DNI no es validado, sino que solo se corroboran ciertos parámetros: su longitud (mayor a 6 y menor a 8 dígitos) y que la información rellenada no sea decimal ni alfanumérica. Al no validar el DNI, se puede dar la situación de que

cualquier usuario introduzca datos falsos, haciéndose pasar por otra persona y enviar el formulario de autochequeo en su nombre,

```
private boolean checkLocationPermission() {
    if (Build.VERSION.SDK_INT < 23 || getActivity() == null || getActivity().checkSelfPermission("android.permission.ACCESS_FINE_LOCATION") == 0) {
        return true;
    }
    requestPermissions(new String[]{"android.permission.ACCESS_FINE_LOCATION"}, REQUEST_LOCATION_ACTIVE);
    return false;
}

private boolean checkActiveLocation() {
    LocationUtils locationUtils = new LocationUtils(this);
    if (locationUtils.isLocationEnabled()) {
        return true;
    }
    locationUtils.displayLocationSettingsRequest(REQUEST_LOCATION_ACTIVE);
    return false;
}

@Override // androidx.fragment.app.Fragment
public void onResume() {
    super.onResume();
    MapView mapView2 = this.mapView;
    if (mapView2 != null) {
        mapView2.onResume();
    }
    if (checkLocationPermission() && checkActiveLocation()) {
        requireContext().startService(new Intent(getContext(), LocationService.class));
    }
}

@Override // androidx.fragment.app.Fragment
public void onRequestPermissionsResult(int i, String[] strArr, int[] iArr) {
    super.onRequestPermissionsResult(i, strArr, iArr);
    if (iArr[0] != 0) {
        SnackbarUtils.with(this.rootView).setMessage("Debes activar los permisos de ubicación").show();
    }
}
```

el cual tiene grado de declaración jurada. La aplicación tampoco permite que se cierre la sesión del usuario.

Dentro del paquete *com.example.cba_covid.reportarmyself* encontramos parte de su implementación:

El recorrido de este proceso se puede visualizar en el siguiente diagrama de flujos:

Registro de ubicación geográfica

Una vez abierta la aplicación, al seleccionar Autochequeo y luego de completar el formulario con los datos personales, aparece una ventana solicitando información sobre el estado de salud de la persona, a continuación aparece una pantalla para el ingreso del domicilio, donde la aplicación obligatoriamente requiere que se otorguen permisos de geolocalización para brindar esa información, debido a que solo permite completarlo mediante Google Maps, sin existir la posibilidad de saltarse la habilitación del permiso y completar el proceso igual.

El código se puede ver en la clase *'Step5LocateHome.java'*.

En esta clase se validan los permisos de una porción de la interfaz (*fragment*) a través de la implementación de dos métodos: *'checkLocationPermission'* y '*checkActiveLocation*'. Ambos métodos devuelven un valor de tipo booleano (Verdadero o Falso),

```
private static final long UPDATE_INTERVAL = 600000;
private static final String credentialsPassword = "[redacted]";
private static long lastReportedTime;
private static Context mContext;
private static FusedLocationProviderClient mFusedLocationClient;
```

dependiendo si el usuario ya ha otorgado permisos de ubicación a la app y si el GPS está habilitado. En el código se puede visualizar que no existe un camino alternativo en caso de que el usuario

```
okHttpClient.newCall(new Request.Builder().url("https://[redacted]").
addHeader("Authorization", Credentials.basic([redacted] + applicationCredentials.getString("username"), credentialsPassword))
.addHeader("Content-Type", "application/json").post(create).build()).enqueue(new Callback() {
```

rechace la habilitación del permiso.

Provincia de La Rioja

Registro de ubicación geográfica

Luego de realizar la instalación, lo primero que solicita la aplicación, antes de mostrar los Términos y Condiciones, es un permiso para acceder a la ubicación del dispositivo. Es decir, no se infor ma previamente al usuario sobre el tratamiento que se realizará de sus datos personales antes de comprobar los permisos del dispositivo que se otorgan a la app.

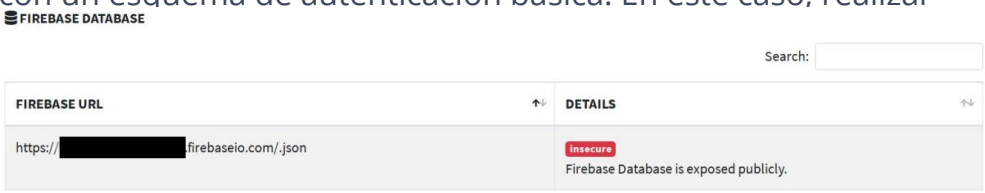
Credenciales escritas en el código fuente

Dentro del paquete *'com.mavimovil.rnsos.location.LocationManager'* se pudo visualizar la existencia de un parámetro crítico con valor

asignado dentro del código, denominado *“credentialsPassword”*.

Esta credencial es utilizada en el método *‘reportLocation’*, y se puede visualizar dentro de la instancia cliente HTTP de la librería ***OkHttp***, utilizada para realizar peticiones HTTP a cualquier API Rest:

Según se pudo analizar en estas líneas de código, esta contraseña es utilizada para realizar las peticiones (*requests*) hacia el servidor, con un esquema de autenticación básica. En este caso, realizar



una petición de envío de datos (POST) hacia el servidor. Por lo que se puede inferir de su vínculo con el método mencionado anteriormente, estos datos tienen relación con la ubicación geográfica del dispositivo.

Método de autenticación y Registro de Usuario

Luego de Aceptar los Términos y Condiciones, se muestra una pantalla para realizar el registro de usuario. En el caso del número de DNI, la app permite ingresar cualquier número siempre y cuando tenga una longitud que no supere los ocho dígitos. Esto permite que se realice una suplantación o

falsificación de identidad. Después del registro, se utiliza esta información para hacer el envío del resultado del autodiagnóstico que tiene carácter de declaración jurada. La aplicación tampoco

admite la posibilidad de cierre de sesión.

Provincia de Jujuy

Base de datos

Realizando una inspección sobre el archivo .apk con la herramienta

MobSF, se identificó un fallo de seguridad grave sobre la base de datos ***Firebase***:

Los datos en Firebase son almacenados en formato .json, y los mismos pueden ser consultados o modificados a través de una API tipo REST. Toda la seguridad de la base de datos se especifica con una serie de ***reglas de autorización***, almacenadas en el archivo *rules.json*, que se basan en tres permisos: lectura, escritura y validación de datos.

Se realizaron una serie de pasos para verificar la existencia de una mala configuración en las reglas de esta base de datos.

Primero, se corroboró la URL de la base, ubicada dentro del archivo *strings.xml*:

Luego se verificó la existencia de permisos públicos a través de una consulta a la API REST, utilizando el comando: *curl '**https://[id-proyecto].firebaseio.com/.json?shallow=true**'*

Dado que la consulta nos retornó el valor *"null"*, se infirió la existencia de una vulnerabilidad, desde la raíz de la jerarquía, ya que de lo contrario, el mensaje retornado sería: *{"error":"Permission*

denied"}

Posteriormente, se chequearon los permisos de escritura:

Lo que devolvió un nuevo JSON con el ID del nodo agregado. Luego, con ese ID devuelto, se logró remover el nodo agregado:

De esta manera se logró corroborar que los permisos de escritura y lectura de nuevos nodos (con un ID determinado) se encuentran abiertos. Inferimos que por el valor "null" devuelto, la base de datos podría contar con tan solo un nombre de nodo (nodo raíz), y este estaría vacío. Por ello, se deduce que dentro de la base de datos no se encuentra información almacenada que guarde relación previa con los datos personales de los usuarios.

El mayor riesgo de una base de datos expuesta es que se puedan leer los datos almacenados en ella, más aún en el contexto de tecnologías utilizadas en medio de una crisis sanitaria, cuando se necesita brindar la mayor confianza posible para su uso.
La empresa **ESET realizó un análisis** de varias aplicaciones en América Latina para identificar posibles vulnerabilidades en las configuraciones de Firebase.

Si bien en la base de datos Firebase de Jujuy no se encontraron datos almacenados, el permitir la escritura y lectura de nuevos datos trae consigo otros problemas.

Una persona que pretenda explotar esta vulnerabilidad puede almacenar información con fines espurios, sobre todo, valiéndose del anonimato que puede obtener al utilizar una base de datos de terceros; lo que terminaría perjudicando el trabajo del personal sanitario que responde a la pandemia, neutralizando la posible efectividad de la aplicación. Asimismo, debido a que Firebase permite almacenar hipervínculos en la base de datos, una persona

```
android:usesCleartextTraffic="true"
```

podría utilizar esta vulnerabilidad para que el administrador de la base haga click en un link e infectar sus dispositivos, mediante técnicas de *phishing*.

Configuración del tráfico de red

Al momento de establecer la configuración de la app para el envío de información a través de internet, es importante prestar atención al uso de protocolos seguros que cifren los paquetes de tráfico (los cuales contienen los datos que se recolectan desde la app), como el caso de HTTPS. Una aplicación que envíe datos, sin cifrar, mediante

```
<string name="coe_official_phone_number">0800 888 4747</string>
<string name="coe_official_url">http://coe.jujuy.gob.ar/</string>
<string name="com.crashlytics.android.build_id">00000000000000000000000000000000</string>
```

```
<string name="provider_api_url">http://coe.jujuy.gob.ar/</string>
<string name="refuse_condition">Rechazar</string>
```

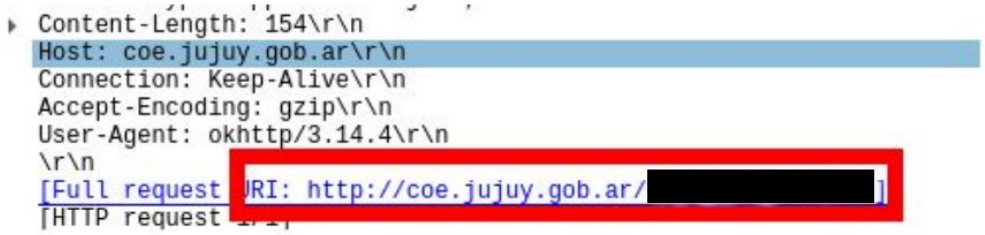
HTTP, podría incrementar el riesgo de que sus comunicaciones sean interceptadas o manipuladas antes de llegar al servidor (por

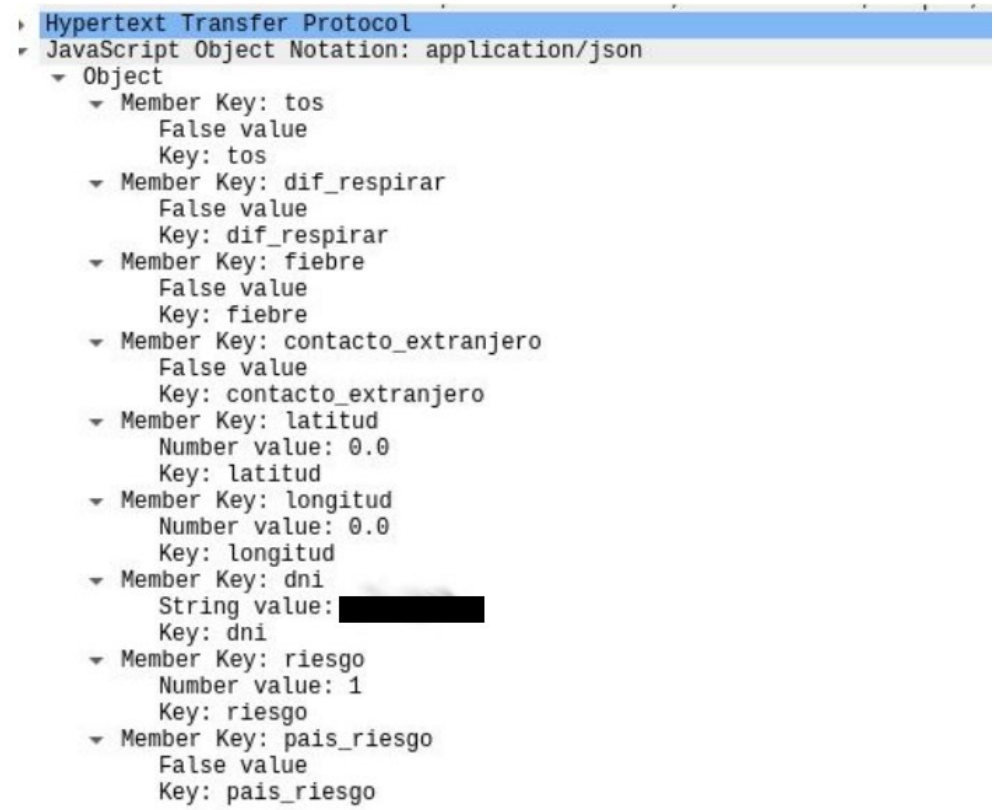
```
private final ApiService getApiService() {
        return (ApiService) ServiceBuilder.INSTANCE.buildService(ApiService.class,
         MapsKt.hashMapOf(TuplesKt.to(ImagesContract.URL, this.context.getString(R.string.provider_api_url)),
          TuplesKt.to("authorization", "")), this.context);
    }
```

ejemplo, algún atacante podría valerse de una técnica de ***sniffing*** al momento de conectar el smartphone a una red Wi-Fi pública). Para evitar que la aplicación se conecte por accidente a servidores o sitios que podrían utilizar protocolos inseguros, se recomienda realizar una **configuración** desde el archivo *AndroidManifest.xml,*

```
Accept-Encoding: gzip\r\n
User-Agent: okhttp/3.14.4\r\n
\r\n
[Full request URI: http://coe.jujuy.gob.ar/[redacted]
[HTTP request [redacted]]
[Response in frame: 1327]
File Data: 372 bytes
```


Transmission Control Protocol, Src Port: 49816, Dst Port: 80, Seq: 1, Ack: 1, Len: 590
Hypertext Transfer Protocol
JavaScript Object Notation: application/json
Object
Member Key: localidad_nombre
String value:
Key: localidad_nombre
Member Key: localidad
Number value:
Key: localidad
Member Key: direccion_numero
String value:
Key: direccion_numero
Member Key: direccion_calle
String value:
Key: direccion_calle
Member Key: apellido
String value:
Key: apellido
Member Key: nombre
String value:
Key: nombre
Member Key: dni
String value:
Key: dni
Member Key: telefono
Number value:
Key: telefono
Member Key: push_notification_id


Content-Length: 154\r\n
Host: coe.jujuy.gob.ar\r\n
Connection: Keep-Alive\r\n
Accept-Encoding: gzip\r\n
User-Agent: okhttp/3.14.4\r\n
\r\n
[Full request URI: http://coe.jujuy.gob.ar/
[HTTP request


Hypertext Transfer Protocol
JavaScript Object Notation: application/json
Object
Member Key: tos
False value
Key: tos
Member Key: dif_respirar
False value
Key: dif_respirar
Member Key: fiebre
False value
Key: fiebre
Member Key: contacto_extranjero
False value
Key: contacto_extranjero
Member Key: latitud
Number value: 0.0
Key: latitud
Member Key: longitud
Number value: 0.0
Key: longitud
Member Key: dni
String value:
Key: dni
Member Key: riesgo
Number value: 1
Key: riesgo
Member Key: pais_riesgo
False value
Key: pais_riesgo

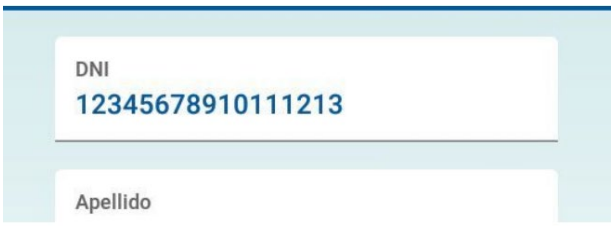
mediante el atributo 'android:usesCleartextTraffic'.

Este atributo fue introducido en Android 6.0 (Marshmallow) y, a partir de la versión Android 7.0 (Nougat), se le dio mayor amplitud introduciendo la función de configuración de seguridad de red de Android; permitiendo a los desarrolladores dar más especificidad a sus comunicaciones a través de un archivo .xml. Si bien su nivel de criticidad también depende en gran medida de otros factores, es destacable mencionar que **este valor se encontró configurado como "true" en la mayoría de las aplicaciones analizadas**.

En el caso particular de la provincia de Jujuy, y por la investigación de protocolos utilizados por la app, notamos la activación incidental de este parámetro, con la implicancia de un riesgo severo como se verá a continuación.

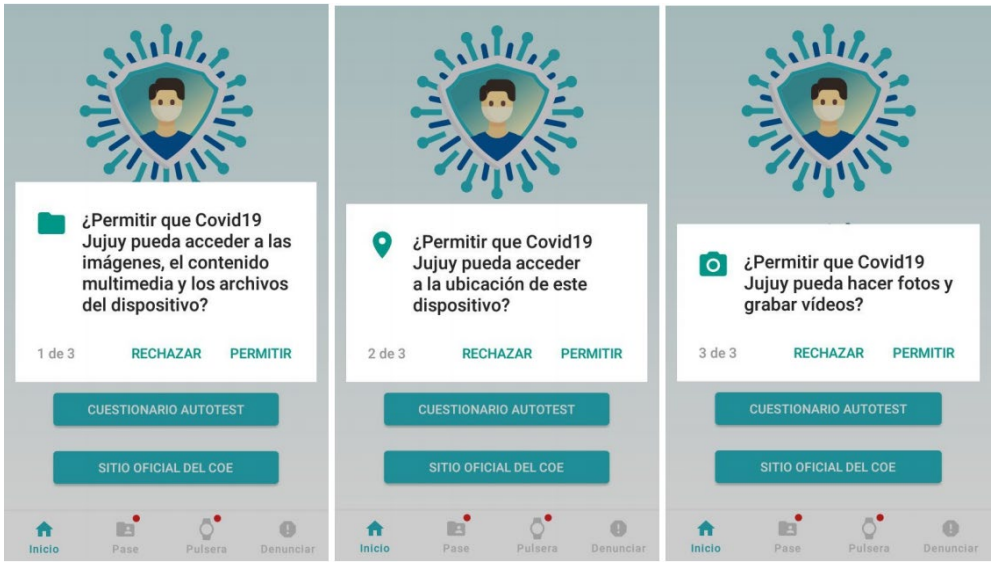
Análisis de los dominios

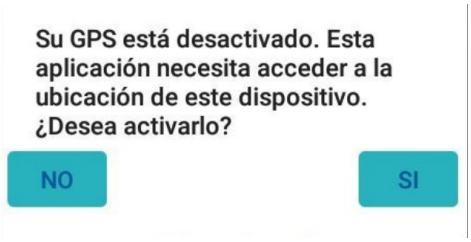
Mediante una inspección en el código, notamos que existen un par de variables asignadas a una url con protocolo HTTP (comunicación no cifrada): *coe_official_url* y *provider_api_url*. De esta última inferimos que podría tratarse de la ruta hacia la API:

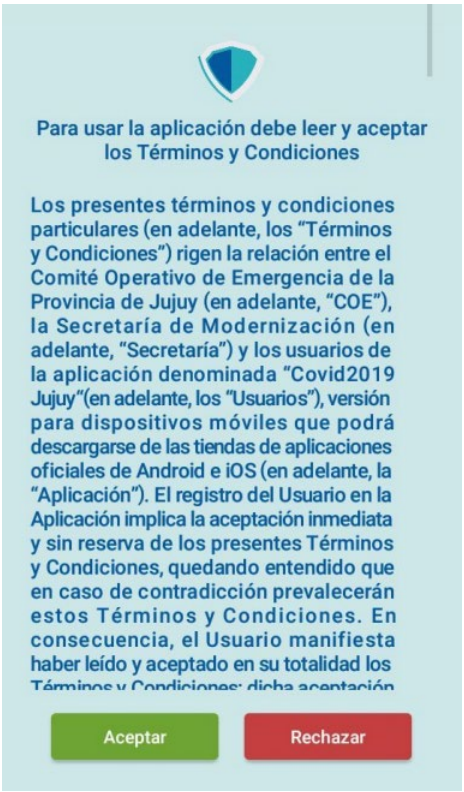


Realizando otra búsqueda, encontramos la utilización de la variable en el método *getApiService()*:

Acercándonos a pensar que se trata de la ruta hacia donde se envía la información.







Para corroborarlo dinámicamente, nos valimos de la herramienta de software libre para análisis de protocolos y tráfico de red, *Wireshark*:

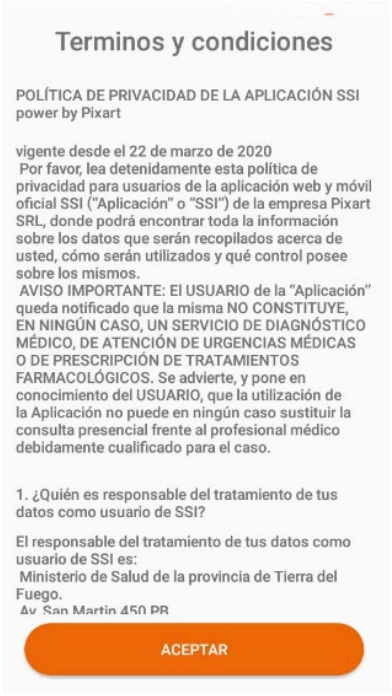
De este análisis se infiere que, efectivamente, se envían todos los datos e información de la persona (nombre, apellido, dni, dirección, localidad, teléfono, ubicación actual, sintomatología) a través de

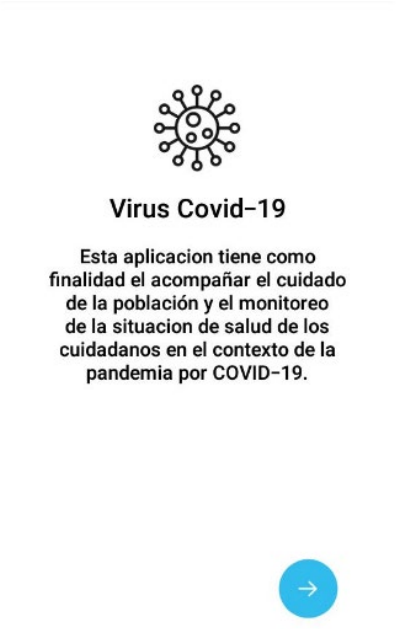
la red sin ningún tipo de cifrado, valiéndose de un protocolo de comunicación sumamente inseguro, hacia dicho host: *coe.jujuy.gob.ar*, coincidente con el sitio oficial del COE (“Comité Operativo de Emergencia”) de la provincia de Jujuy. Cabe destacar que la app fue configurada de manera deliberada para permitir dicha conexión sin cifrar hacia el servidor.

```
public static final String is_register_user_perfil = "is_register_user_perfil";
public static final String is_secret_key = [redacted];
public static final String is_show_presentation = "is_show_presentaion";
```

También es posible acceder a este sitio desde la pantalla principal de la aplicación.

Resultando preocupante, además, que cuente con un área o sección de *login*.





Método de autenticación y Registro de Usuario

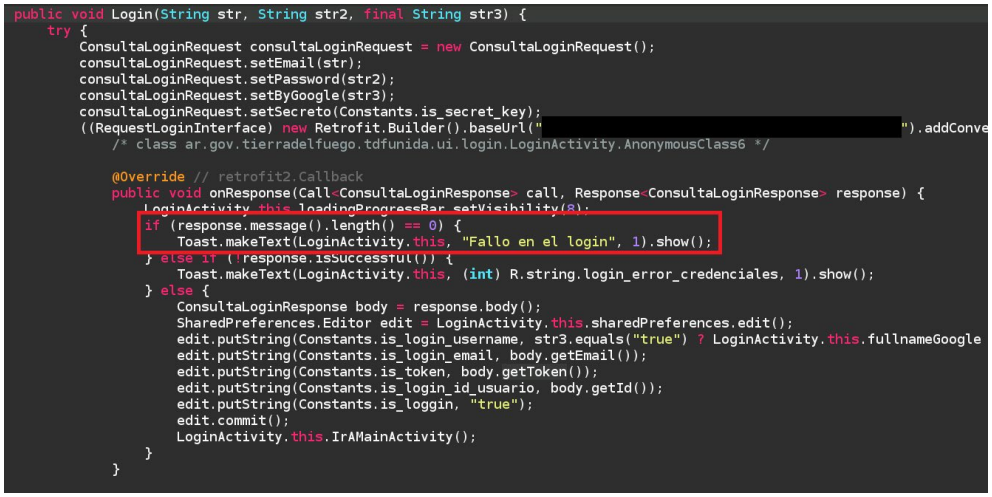
Al iniciar la aplicación por primera vez, se solicita el registro del usuario y sus datos personales. Para el ingreso del DNI, no existe

validación en cuanto al número de dígitos ingresados, ni proceso de verificación de identidad. La aplicación tampoco facilita el cierre de sesión de usuario.

Solicitud de Permisos

La primera vez que se abre la aplicación y al finalizar el registro con los datos personales, se muestran las solicitudes para acceder al contenido multimedia, ubicación del dispositivo y dar acceso a la cámara. Las tres solicitudes se realizan por duplicado. En caso de que se

rechacen dichas solicitudes, la aplicación deja de responder.



```
public void Login(String str, String str2, final String str3) {
    try {
        ConsultaLoginRequest consultaLoginRequest = new ConsultaLoginRequest();
        consultaLoginRequest.setEmail(str);
        consultaLoginRequest.setPassword(str2);
        consultaLoginRequest.setByGoogle(str3);
        consultaLoginRequest.setSecreto(Constants.is_secret_key);
        ((RequestLoginInterface) new Retrofit.Builder().baseUrl("[redacted]").addConve
            /* class ar.gov.tierradelfuego.tdfunida.ui.login.LoginActivity.AnonymousClass6 */

            @Override // retrofit2.Callback
            public void onResponse(Call<ConsultaLoginResponse> call, Response<ConsultaLoginResponse> response) {
                LoginActivity.this.loadingProgressBar.setVisibility(8);
                if (response.message().length() == 0) {
                    Toast.makeText(LoginActivity.this, "Fallo en el login", 1).show();
                } else if (!response.isSuccessful()) {
                    Toast.makeText(LoginActivity.this, (int) R.string.login_error_credenciales, 1).show();
                } else {
                    ConsultaLoginResponse body = response.body();
                    SharedPreferences.Editor edit = LoginActivity.this.sharedPreferences.edit();
                    edit.putString(Constants.is_login_username, str3.equals("true") ? LoginActivity.this.fullnameGoogle
                    edit.putString(Constants.is_login_email, body.getEmail());
                    edit.putString(Constants.is_token, body.getToken());
                    edit.putString(Constants.is_login_id_usuario, body.getId());
                    edit.putString(Constants.is_loggin, "true");
                    edit.commit();
                    LoginActivity.this.IrAMainActivity();
                }
            }
```

Si se aceptan las solicitudes y se tiene el GPS desactivado, se le notifica al usuario que ese permiso es necesario.

En caso de que se niegue, retorna a la pantalla de Términos y Condiciones. El usuario debe aceptar para seguir utilizando la aplicación, donde nuevamente se le solicitará la activación del GPS.

Glosario

- **Variable:** espacio de memoria reservado para almacenar datos. Se identifica con una etiqueta o un nombre establecido por el programador, el cual, además, le asigna un valor que puede cambiar durante la ejecución del programa.

- **Función:** una función es un grupo de instrucciones con un objetivo en particular y que se ejecuta al ser llamada desde otra función o procedimiento. Una función puede llamarse múltiples veces e incluso llamarse a sí misma (función recursiva).
  Las funciones pueden recibir datos desde afuera al ser llamadas a través de los parámetros y deben entregar un resultado.Se diferencian de los procedimientos porque estos no devuelven un resultado. En general las funciones deben tener un nombre único en el ámbito para poder ser llamadas, un tipo de dato de resultado, una lista de parámetros de entrada y su código.

- **Biblioteca o librería:** colección o conjunto de funciones usadas para desarrollar software y resolver necesidades específicas. En general, las bibliotecas no son ejecutables, pero sí pueden ser usadas por ejecutables que las necesitan para poder funcionar correctamente. La mayoría de los sistemas operativos proveen bibliotecas que implementan la mayoría de los servicios del sistema y son utilizadas por las aplicaciones que corren sobre dicho sistema operativo.

- **Hash** (función criptográfica)**:** es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija. Independientemente de la longitud de los datos de entrada, el valor hash de salida tendrá siempre la misma longitud. Estas funciones sólo funcionan en un solo sentido. Las más utilizadas en informática son: SHA-1, MD5 y SHA-2. Dos de los usos más

comunes son el de almacenamiento de contraseñas y el de corroboración de la integridad de un archivo (saber si ha sido modificado o no).

- **Codificación:** conjunto de reglas con las cuales convertir algo desde una representación a otra. Es el proceso por el cual la información de una fuente es convertida en símbolos para ser comunicada (de un formato a otro), con el propósito de estandarización, lograr mayor velocidad o capacidad de compresión.

- **Ingeniería inversa:** Reconstrucción de un producto ya existente, en este caso, software; para entender su configuración, estructura y funcionamiento. Suele emplearse con fines de aprendizaje, diagnóstico, análisis de seguridad y pirateo.

- **Ofuscación**: es el acto deliberado de crear código fuente o de máquina que es difícil de entender para los humanos. Los programadores suelen ofuscar deliberadamente el código para ocultar su propósito (seguridad a través de la oscuridad), su lógica o valores implícitos incrustados en él, principalmente, para evitar la manipulación o disuadir la ingeniería inversa. Se puede hacer manualmente o mediante herramientas automatizadas (más utilizada)..

- **Nombre de Dominio:** Nombre fácil de recordar asociado a una dirección IP lógica de Internet. Cada nombre de dominio es único. Para realizar la "traducción" de nombres de dominios a direcciones IP e identificar dónde está alojada la información a la cual se pretende acceder, existe el sistema denominado Domain Name System, DNS.

- **API:** abreviatura de Application Programming Interfaces, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar software, permitiendo la comunicación entre dos aplicaciones a través de un conjunto

de reglas. Sus respuestas generalmente se devuelven como datos JSON o XML.

- **API Rest:** Conjunto de funciones que permite a los desarrolladores realizar solicitudes y recibir respuestas a través del protocolo HTTP, como GET y POST

- **Plugin** (complemento)**:** aplicación que añade una funcionalidad adicional o una nueva característica a un software determinado.

- **Comando host:** Instrucción que se usa, principalmente, para encontrar la dirección IP de un dominio determinado y también para mostrar el nombre de dominio para una IP dada.

- **Comando whois:** Instrucción que busca en la base de datos pública “whois” (“quién es”) para obtener información sobre el propietario de un nombre de dominio en particular. La información proporcionada puede incluir el nombre, dirección, correo electrónico y número de teléfono. El comando whois también devolverá los servidores de nombres de dominio.

- **Comando nslookup:** es un programa utilizado para realizar diversos tipos de consultas sobre servidores de nombres, DNS.

Anexo

Comparación de las aplicaciones analizadas

	Fecha en que se subió a Google Play	Última actualización	Organismo y/o empresa encargada del desarrollo	Cantidad de permisos	Trackers	Permisos de ubicación	Cantidad de descargas en Google Play Store	Obligatoriedad (según información disponible públicamente)	Funciones	Datos solicitados	Método de autenticación
Cuidar	21/03/2020	28/11/2020	Secretaría de Innovación Pública (Jefatura de Gabinete), Fundación Sadosky, CONICET, Hexacta, Globant, G&L Group, C&S, QServices, GestiónIT, Intive, Finnegans y Faraday (nucleadas en la Cámara de la Industria Argentina del Software -CESSI-); ARSAT; Amazon Web Services	8 permisos	New Relic, Google Firebase Analytics	Ubicación Aproximada	**Más de 10 millones**	La **Resolución Conjunta 11/2020 del Ministerio de Salud y la Dirección Nacional de Migraciones** recomienda el uso de la app "dentro de las 48 hs del arribo al país autoreportando síntomas cada 48 hs". Asimismo, aclara que no es exigible la geolocalización.	(1) Realizar autodiagnóstico; (2) Portar el certificado de Circulación (supeditado al resultado del autodiagnóstico)	DNI, Número de trámite, sexo, Dirección actual, provincia, localidad, código postal, teléfono de contacto.	DNI + Número de trámite
SaltaCovid	23/04/2020	11/11/2020	Agencia de Información de Políticas Pública (Gobierno de la provincia de Salta)	10 permisos	Google CrashLytics, Google Firebase Analytics	Ubicación Exacta y Aproximada	**Más de 50 mil**	**Obligatoria para quienes ingresen a la provincia**	(1) Brindar Información; (2) Cuestionario	Tipo y número de documento, Apellido, Nombre, Edad, Sexo, Cobertura de Salud, Personas Convivientes, teléfono fijo o celular, domicilio actual.	Se accede a la funcionalidad de la app una vez completado el registro de usuario (se permite ingresar y registrar múltiples a usuarios).
Vecino Salud (Córdoba)	25/03/2020	5/10/2020	Secretaría de Planeamien-to, Modernización y Relaciones Internacionales (Municipalidad de Córdoba)	8 permisos	Google CrashLytics, Google Firebase Analytics	Ubicación Exacta	**Más de 10 mil**	No obligatoria para circular	(1) Información ("Consultar Síntomas", "Prevención"); (2) Preguntas Frecuentes; (3) Autochequeo	Dni, Nombre, Apellido, Fecha de Nacimiento, Teléfono, Género.	Se accede a las funciones de la app una vez completados todos los datos personales."
Santa Fe	12/04/2020	25/09/2020	Gobierno de la Provincia de Santa Fe	40 permisos	Google Firebase Analytics	Ubicación Exacta, aproximada, y en segundo plano (backgound location)	**Más de 100 mil**	No obligatoria para circular	(1) Síntomas y recomendaciones; (2) Trámites/ Denuncias; (3) Casos: Información oficial; (4) Notificaciones; (5) Test de autoevaluación; (6) DDJJ de salud	Apellido, Nombre, teléfono, DNI.	Solicita ingreso con cuenta de Google.
Jujuy*	1/4/2020	4/5/2020	Comité Operativo de Emergencia (COE) de Jujuy, Gobierno de la provincia de Jujuy, Usound, Nubimetric	14 permisos	Google CrashLytics, Google Firebase Analytics	Ubicación Exacta y Aproximada	**Más de 10 mil**	No obligatoria para circular	(1) Cuestionario Autotest; (2) Sitio oficial del COE	DNI, Apellido, Nombre, Domicilio, Localidad, Número telefónico.	DNI, Apellido, Nombre, Domicilio, Localidad, Número telefónico.
TDFUnida*	15/4/2020	3/5/2020	Gobierno de la provincia de TDF, Pixart SRL	12 permisos	Google Firebase Analytics	Ubicación Exacta, aproximada, y en segundo plano (backgound location)	**Más de 500**	No obligatoria para circular	(1) Farmacias y hospitales; (2) cuarentena; (3) Autocuidado; (4) autorizaciones; (5) Comunicaciones; (6) Prevención		Usuario, correo electrónico y contraseña. También permite ingresar con una cuenta de Google.
C19 Test La Rioja*	30/3/2020	22/4/2020	Ministerio de Salud (Gobierno de la Rioja), Comité Operativo de Emergencia (COE) de la Rioja, Subsecretaria de Innovación Pública de la Rioja	25 permisos	Google Firebase Analytics	Ubicación Exacta y Aproximada	**Más de 10 mil**	No obligatoria para circular	(1) Mapa; (2) Diagnóstico; (3) consejos; (4) alertas	DNI, Nombre y Apellido, teléfono (verificación código SMS), Año de nacimiento, género. dirección, localidad	

(*) aparentemente discontinuadas

ADC 25 AÑOS
CONSTRUYENDO PUENTES