ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Aplicaciones móviles para Contact Tracing usando Bluetooth

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Telemática

Presentado por:

Víctor José Espinoza Molina

Eduardo Javier Veintimilla Ullauri

GUAYAQUIL – ECUADOR

Año: 2020

DEDICATORIA

El presente proyecto lo dedicamos a nuestros padres y hermanos que siempre están apoyándonos en cada cosa que realizamos.

AGRADECIMIENTOS

Mis más sinceros agradecimientos a nuestra Alma mater ESPOL que asentaron todas las bases para llevar a cabo cualquier proyecto de índole tecnológico y social.

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, me(nos) corresponde conforme al reglamento de propiedad intelectual de la institución; *Víctor y Eduardo* doy(damos) mi(nuestro) consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

_____ _____

Autor 1 Autor 2

EVALUADORES

PhD. José Córdova García	**Msig. Adriana Collaguazo Jaramillo**
PROFESOR DE LA MATERIA	PROFESOR TUTOR

RESUMEN

Las aplicaciones móviles para contact tracing contribuyen a evitar la propagación del COVID-19 ya que usan la tecnología Bluetooth Low Energy para el rastreo de contactos la cual es un mecanismo de forma digital en comparación al rastreo de contactos realizado de manera presencial por las entidades de salud. Se plantea analizar la aplicación móvil de contact tracing llamada ASI Ecuador para obtener un marco de referencia respecto a los recursos de hardware e información consumida, debido a que supuestamente ocupan muchos recursos de hardware en un dispositivo móvil y la privacidad del usuario se ve comprometida. Este análisis permite obtener información que es fundamental cuando se pretende desplegar este tipo de aplicaciones móviles y en donde los usuarios deben sentirse conformes al momento de usarlas.

Se usaron laptops y dispositivos móviles Android, además de entornos de desarrollo, software y programas libres añadiendo dos técnicas para el análisis de la aplicación móvil las cuales corresponden a la técnica de caja negra y blanca.

Luego del análisis, se obtuvieron resultados como el promedio máximo de memoria RAM y CPU usados, además del consumo de batería y el promedio de tasa de red usados y, por otro lado, se rescató información entre la comunicación cliente-servidor y sobre la interacción entre los dispositivos móviles usando Bluetooth Low Energy.

Demostrando que hay seguridad para el usuario y poco consumo de recursos de hardware excepto por el consumo de batería por parte de Bluetooth en los dispositivos Android.

Palabras Clave: Contact tracing, Bluetooth Low Energy, recursos de hardware, comunicación cliente-servidor.

ABSTRACT

Mobile applications for contact tracing help prevent the spread of COVID-19 since they use Bluetooth Low Energy technology for contact tracing, which is a digital mechanism compared to contact tracing carried out in person by the entities of Health. It is proposed to analyze the contact tracing mobile application called ASI Ecuador to obtain a frame of reference regarding the hardware resources and information consumed, because they supposedly occupy many hardware resources on a mobile device and the user's privacy is compromised. This analysis allows obtaining information that is essential when it is intended to deploy this type of mobile applications and where users must feel satisfied when using them.

Android laptops and mobile devices were used, as well as development environments, software and free programs, adding two techniques for the analysis of the mobile application which correspond to the black and white box technique.

After the analysis, results were obtained such as the maximum average of RAM and CPU used, in addition to the battery consumption and the average network rate used and, on the other hand, information was recovered between the client-server communication and about the interaction between mobile devices using Bluetooth Low Energy.

Demonstrating that there is security for the user and little consumption of hardware resources except for the battery consumption by Bluetooth in Android devices.

Keywords: *Contact tracing, Bluetooth Low Energy, hardware resources, client-server communication.*

ÍNDICE GENERAL

ABREVITURAS

BLE Bluetooth Low Energy

RSSI Received Signal Strength Indicator

RAM Random Access Memory

CPU Central Processing Unit

iOS iPhone OS

GPS Global Positioning System

DP3T Decentralized Privacy-Preserving Proximity Tracing

CSV Comma-Separated Values

PNG Portable Network Graphics

GAEN Google/Apple Exposure Notifications

ADB Android Debug Bridge

HCI Host Controller Interface

USB Universal Serial Bus

API Application Programming Interface

UML Unified Modeling Language

HTTP Hypertext Transfer Protocol

MitM Man-in-the-Middle

AES Advanced Encryption Standard

JSON Javascript Object Notation

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

En todo el mundo se han tomado medidas de bioseguridad a causa del COVID-19 para disminuir la expansión de la enfermedad y se espera que a finales del año 2020 se tenga una vacuna para contrarrestar el virus debido a los avances realizados por empresas farmacéuticas los cuales se siguen con optimismo alrededor del mundo [8] y mientras tanto es preciso llevar a cabo estas medidas aplicando aislamientos, cuarentenas o distanciamiento social para reducir la cantidad de infectados registrados en una zona geográfica.

Ante la necesidad de ayudar a combatir la pandemia, gran parte de gobiernos nacionales [1] se han involucrado en la implementación de aplicaciones móviles de rastreo de contactos que ayudan a identificar a las personas que estuvieron en contacto con una persona infectada y así recolectar esta información para su conocimiento y notificación a las demás personas.

El desarrollo de aplicaciones móviles ha contribuido en varios sectores económicos, como es el caso de las empresas [9] en el campo del e-commerce, facilitando a sus usuarios en la adquisición de los productos. En la agricultura como complemento de varias tecnologías [10] tales como el internet de las cosas, inteligencia artificial y el manejo de grandes datos para aumentar la productividad agrícola y la calidad de los productos agrícolas. Hoy en día debido al crecimiento exponencial de los dispositivos móviles inalámbricos [11] se puede tener acceso a datos relacionados con la salud por medio de aplicaciones móviles implementadas por instituciones de salud las cuales hasta permiten desarrollar prácticas médicas a distancia (telemedicina) y llevar un control de las enfermedades que padecen los pacientes.

1.1 Descripción del problema

Las aplicaciones móviles de contact tracing han llamado la atención por la contribución que estas pueden realizar para evitar el contagio de COVID-19 a más personas, pero por otro lado el uso de estas aplicaciones ha generado interrogantes sobre el consumo de recursos de hardware del dispositivo móvil tales como memoria RAM, CPU, interfaz de red, interfaz Bluetooth y en la privacidad de los datos ya que es de mucha importancia saber si en este tipo de aplicaciones móviles existen violación o fuga de datos.

1.2 Justificación del problema

Las aplicaciones móviles de rastreo de contactos usan diferentes arquitecturas en las cuales son desplegadas para su funcionamiento donde algunas son ligeras y otras muy invasivas como es el caso del sistema de China la cual recoge los datos de identidad de las personas que usan la aplicación [12]. Además, surgen preguntas por parte de los usuarios de la aplicación sobre qué datos son recolectados y cómo es el proceso de notificación en el caso de estar cerca de un infectado originando dudas sobre la anonimidad de este.

En Ecuador se ha desplegado la aplicación móvil de rastreo de contactos ASI Ecuador la cual funciona tanto para dispositivos móviles con sistema operativo Android e IOS, pero de acuerdo a las opiniones vertidas en el perfil de descarga de Google Play Store de la aplicación, hacen énfasis en el alto consumo de recursos de hardware, entre las opiniones también mencionan que la aplicación usa GPS mientras que en la descripción de la aplicación niegan usar datos de geolocalización y, que además todo depende de la intención de la persona para registrar si fue un caso positivo de COVID-19, algunos de estos comentarios han ocasionado un interés en el estudio de este tipo de aplicaciones móviles.

El análisis que se realiza a este tipo de aplicaciones móviles permitirá conocer cuánto es lo máximo de recursos computacionales o de hardware que usan en los dispositivos móviles y si existe una privacidad en los datos del usuario ya que al momento de desplegar la aplicación móvil tendría una mejor aceptación entre la comunidad que la

usa debido a que la función principal de la aplicación si la lleva a cabo para satisfacer la necesidad de los usuarios.

1.3 Objetivos

1.3.1 Objetivo General

- Analizar aplicaciones móviles de rastreo de contactos que utilizan Bluetooth para la obtención de un marco de referencia respecto a los recursos de hardware e información consumida.

1.3.2 Objetivos Específicos

1. Realizar una simulación local de la aplicación móvil ASI Ecuador con su respectivo servidor para obtener resultados referentes al uso de Bluetooth y a la comunicación cliente-servidor.
2. Implementar un script propio para la complementación de una parte de las técnicas de pruebas a usar.
3. Usar herramientas open-source para visualizar el comportamiento del hardware usado por las aplicaciones móviles de rastreo de contactos y el registro de la comunicación cliente-servidor.

1.4 Marco teórico

El rastreo de contactos es una técnica utilizada por organismos de salud, generalmente públicos, para contribuir a detener la expansión de enfermedades infecciosas, como, por ejemplo, las ocasionadas por el coronavirus SARS-CoV-2 (mejor conocido con el nombre de COVID-19) en los actuales momentos, o el Ébola en épocas anteriores [7]. Su procedimiento comienza cuando se confirma un caso positivo, se investigan todas las actividades realizadas por el paciente desde el momento que presentó síntomas y asimismo las personas con las que tuvo contacto, después, se buscan dichas personas y se les realizan pruebas para asegurar o descartar que se hayan contagiado. De ser afirmativo, el individuo en cuestión entra en fase de seguimiento (atención médica) y se le obliga a permanecer en un período de aislamiento para impedir que el virus se siga reproduciendo. Igualmente, se repite el proceso antes

descrito (las veces que sean necesarias) y se indaga sobre sus acciones previas y sujetos implicados [6].

Todas las aplicaciones móviles de rastreo de contactos tiene un mismo objetivo, el cual es reducir el número de contagios, para llevar a cabo esto, usan sensores de los celulares acompañado de arquitecturas de despliegue [1] donde el 43% usan GPS y el 57% usa Bluetooth con arquitecturas tanto centralizadas y descentralizadas, esta última sin el uso de GPS presta privacidad de datos en comparación con las arquitecturas centralizadas por la cual se está haciendo tendencia a nivel mundial principalmente en Europa el uso de arquitecturas descentralizadas.

Como se mencionó, la mayoría de estas aplicaciones usan Bluetooth lo que ha llevado a realizar investigaciones en términos de interferencia para determinar la distancia social con precisión demostrando que aún hay desafíos por enfrentar con respecto a Bluetooth.

Hoy en día, hay un sin número de dispositivos portátiles o wearables como pulseras o relojes inteligentes que han salido al mercado para mitigar la propagación del virus. Cabe resaltar a los celulares como parte de este medio por su gran aporte en el día a día de las personas [2] razón por la cual se debe centrar en investigaciones adicionales como la periodicidad de las señales de transmisión y la relación con la velocidad de movimiento del usuario y consumo detallado de batería sobre estos dispositivos que son usados para el rastreo de contacto.

Las tecnologías inalámbricas consideradas para el rastreo de contactos son Bluetooth y GPS, la primera muestra una tasa de falsos positivos relativamente menor para rastrear contactos en comparación a la segunda que posee características como lo son su alto consumo de energía llevando al agotamiento de la batería del smartphone si es usado de manera prolongada y su baja precisión de ubicación en interiores y centros comerciales [3].

La interfaz Bluetooth tiene la cualidad de capturar la intensidad de la señal inalámbrica recibida (RSSI) para estimar la distancia de donde se transmite esta, es decir, el receptor estima que tan lejos está el transmisor por medio del RSSI, sin

embargo, la señal puede verse atenuada debido a obstáculos físicos e interferencia de otras que están a la misma frecuencia de Bluetooth afectando a la estimación de la distancia. Factores como el nivel de potencia y la orientación o ubicación del dispositivo móvil ocasionan también una variación de la estimación de la distancia [3].

Hay una diferencia entre el Bluetooth tradicional y otra variante de Bluetooth denominada Bluetooth Low Energy (BLE) [13] en la cual radica en el consumo de energía ya que esta última consume menos y es ideal para los wearables con propósitos de registro de contactos ya que usa balizas que son transmisores usados para propagar la señal bluetooth pero de baja energía y así detectar otros dispositivos, pero de la misma manera que Bluetooth tradicional, presenta problemas en distinguir la precisión dada entre los contactos, este factor es muy importante para las organizaciones de la salud para determinar la efectividad del rastreo de contactos automatizado.

Se han recolectado ciertos puntos de debate sobre el uso de las aplicaciones de contact tracing como lo son el tipo de arquitectura, el manejo de los datos recolectados, la privacidad y seguridad empleada para llevar el registro de contactos [4].

Entre las arquitecturas usadas por diferentes aplicaciones móviles de rastreo de contactos están:

Las **arquitecturas centralizadas** son aquellas donde la aplicación móvil se registra con el servidor central y este genera un identificador aleatorio cifrado para cada dispositivo que tiene la aplicación instalada, luego la aplicación intercambia los identificadores por medio de Bluetooth con otros dispositivos que también tienen esta aplicación y en el caso de que el usuario padezca de COVID-19 positivo, este enviará al servidor central, siempre y cuando un organismo de la salud lo confirme, todos los identificadores que ha recibido el smartphone del infectado en el intercambio. Luego el servidor central es el encargado de descifrar esos identificadores y realizar un análisis de los datos usando la potencia de transmisión y RSSI que son características de Bluetooth para luego enviar notificaciones a las personas que estuvieron cerca o en contacto con el caso positivo de COVID-19 [4]. En la figura 1.1 se muestra el diagrama de las arquitecturas centralizadas.

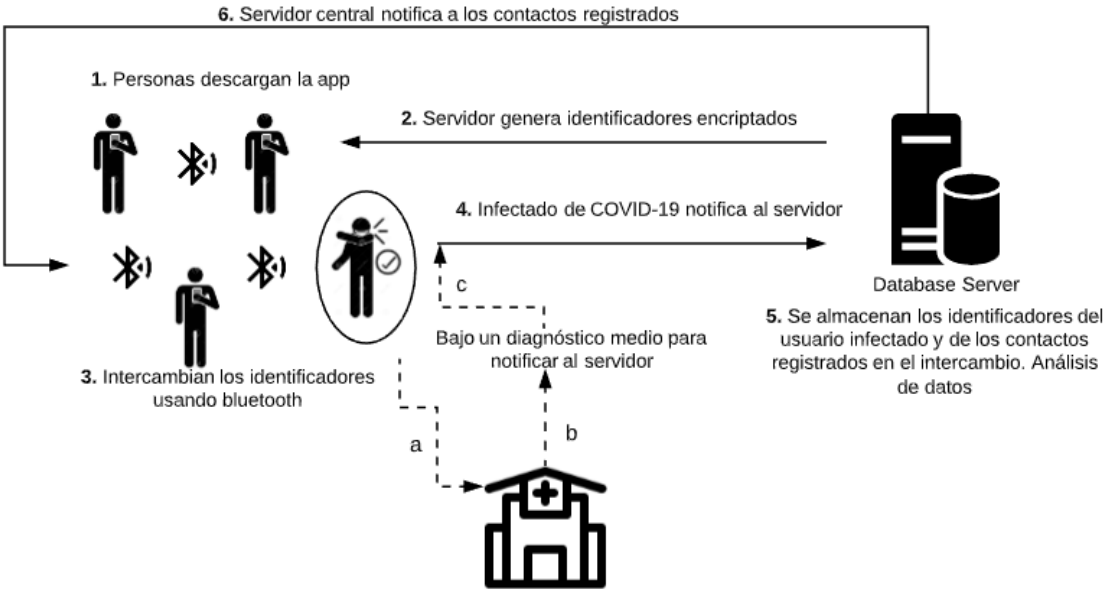


Figura 1.1. Arquitectura centralizada

Entre los protocolos usados para el intercambio de datos usando Bluetooth en una arquitectura centralizada están BlueTrace y ROBERT, algunas aplicaciones que usan este tipo de arquitectura se muestran en la Tabla 1.1.

Tabla 1.1. Aplicaciones móviles centralizadas para contact tracing

Nombre	**País**	**Protocolo**	**Características técnicas**
TraceTogether	Singapur	Bluetrace	Cámara, ubicación (GPS), almacenamiento, Bluetooth, Google Cloud, Android & iOS.
CovidSafe	Australia	Bluetrace	Ubicación (GPS), Bluetooth, AWS, Android & iOS

StopCovid	Francia	ROBERT	Cámara, ubicación (GPS), Bluetooth
Aarogya Setu	India	DAKS (Data Access and Knowledge Sharing)	Cámara, ubicación (GPS), Bluetooth, Android.
Corona-Warn-App	Alemania	Privado	Android & iOS, Cámara, Bluetooth
eRouška	República Checa	Privado	Bluetooth, Android & iOS.
NHS COVID-19	Reino Unido	NHS contact tracing protocol	Android & iOS, Bluetooth
Smittestop	Dinamarca	Privado	Bluetooth, Android & iOS
Health Code on Alipay and WeChat	China	Privado	Código QR, Android & iOS, Ubicación (GPS)
EHTERAZ	Qatar	Privado	Bluetooth, Ubicación (GPS), Android.

Las **arquitecturas descentralizadas** son aquellas en donde el servidor central tiene una participación mínima y solo interviene como un punto de encuentro para anunciar los usuarios infectados ya que el smartphone con la aplicación de contact tracing realiza todo el análisis de los intercambios de identificadores, además cada dispositivo genera aleatoriamente un identificador y si es caso positivo del virus pues notifica al servidor central siempre y cuando haya un informe médico de algún centro de salud verificado para que este lo registre en una lista negra que luego se notificará a los smartphones que estuvieron cerca del infectado, este tipo de arquitectura permite ayudar en la seguridad y privacidad del usuario [4]. En la figura 1.2 se muestra el diagrama de las arquitecturas descentralizadas.

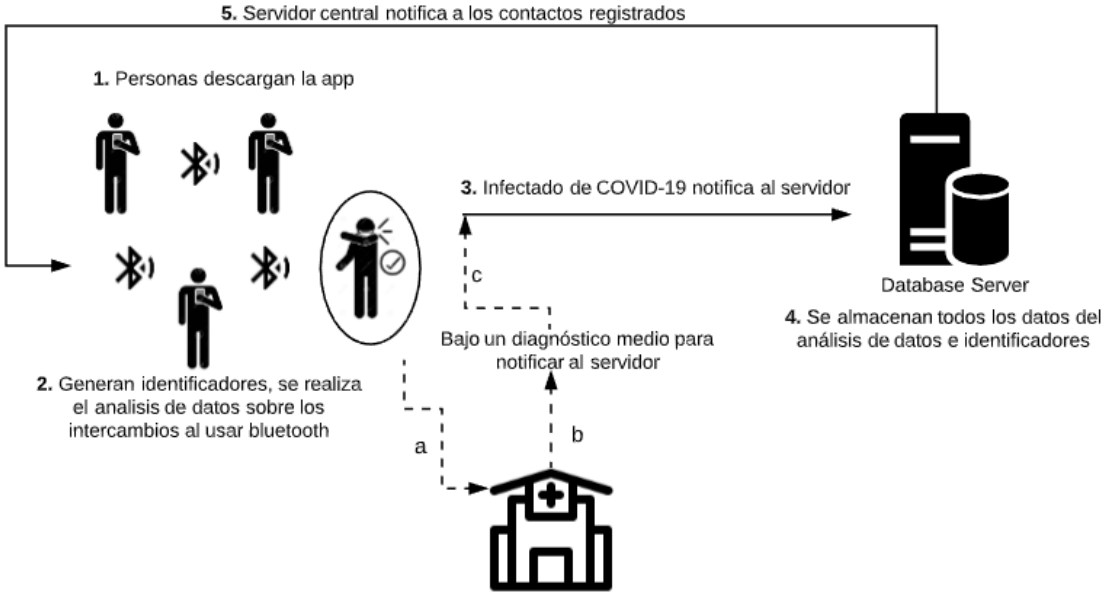


Figura 1.2. Arquitectura descentralizada

Las arquitecturas descentralizadas tienen algunos protocolos para el rastreo de contactos los cuales son: PACT West-coast (UoW), PACT East-coast (MIT), DP-3T, TCN, Pronto-C2 y GAEN (Google/Apple Exposure Notification). Algunas aplicaciones que usan este tipo de arquitectura se muestran en la Tabla 1.2:

Tabla 1. 2. Aplicaciones móviles descentralizadas para contact tracing

Nombre	**País**	**Protocolo**	**Características técnicas**
SwissCovid	Suiza	DP-3T	Bluetooth, Android & iOS
Covid Watch Arizona	USA	TCN	Bluetooth, Android & iOS
ASI	Ecuador	DP-3T	Bluetooth, GPS, Android & iOS
TraceCovid	Emiratos Árabes Unidos	Privado	Android & iOS, Bluetooth
COVID Alert	Canadá	GAEN	Android & iOS, Bluetooth
Immuni	Italia	GAEN	Android & iOS, Bluetooth Low Energy.

VirusRadar	Hungría	Privado	Android, Bluetooth, Ubicación (GPS)
BlueZone	Vietnam	Privado	Bluetooth, Android & iOS, Cámara
ProteGO Safe	Polonia	GAEN	Android & iOS, Bluetooth
Stopp Corona	Austria	GAEN	Bluetooth, Android & iOS

CAPÍTULO 2

2. METODOLOGÍA

Para las pruebas de las aplicaciones móviles de rastreo de contactos se ha tomado como caso específico ASI Ecuador, las razones por las cuales se escogió esta aplicación es por la cantidad de comentarios obtenidos respecto al consumo de recursos de hardware en un dispositivo móvil demostrando la falta de información hacia los usuarios, además sobre aspectos de uso y finalidad de la aplicación así como también la parte de seguridad de datos ya que muchas personas desean mantener su anonimidad en el caso de confirmar COVID positivo.

2.1 Técnicas y tipos de pruebas de software

Existen dos técnicas de prueba para las aplicaciones móviles o software en general, están son las **pruebas de caja blanca** que consisten en la realización de las pruebas teniendo el conocimiento tanto del código, tecnologías y arquitectura que constituyen la aplicación móvil o software y las **pruebas de caja negra** que se enfocan en el lado de la interfaz gráfica sin tener en consideración el funcionamiento de manera interna, es decir, la estructura del código y arquitecturas usadas [14].

Por otro lado, están los tipos de pruebas las cuales se clasifican en dos grandes grupos los cuales son las **pruebas funcionales** [15] que son aquellas que se realizan para confirmar el correcto funcionamiento de la aplicación móvil o software objetivo y las **pruebas no funcionales** [16] que son aquellas que verifican aspectos no funcionales tales como rendimiento, usabilidad, seguridad, entre otras.

2.2 Pruebas de caja negra

En este tipo de prueba, para la aplicación móvil en cuestión, se implementan herramientas necesarias para observar mediante gráficas estadísticas la utilización de los recursos de memoria RAM usada y porcentaje de CPU usado en el dispositivo móvil en tiempo real cuando la aplicación objetivo está siendo usada por el usuario. A

continuación, en la figura 2.1 se muestra un diagrama que abarca tanto el hardware necesario, programas, librerías y las etapas a realizar.

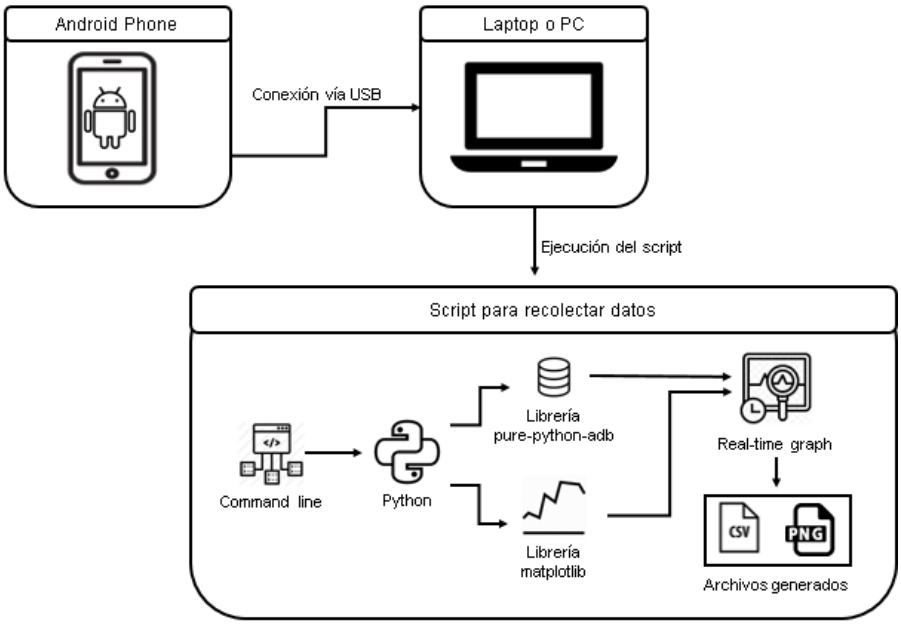


Figura 2.1. Esquema de obtención de datos

Con respecto a la figura 2.1 se observa un dispositivo móvil con sistema operativo Android en el cual se habilita las opciones de desarrollador (definición que se explicará más adelante), el cual se lo conecta mediante cable USB a una laptop en donde se instala la herramienta ADB y Python 3.7 para construir y ejecutar el script que permitirá recolectar los datos de memoria RAM y CPU usado por la aplicación móvil ASI Ecuador, luego realizará graficas de series de tiempo por medio de la librería matplotlib [17] y generará archivos con extensión .csv y .png que corresponde a la gráfica generada.

2.2.1 Hardware

Al llevar a cabo el análisis de las aplicaciones móviles de rastreo de contactos es necesario contar con un dispositivo móvil o crear un entorno virtual usando hipervisores dentro de una computadora para instalar el sistema operativo Android. Para ambos casos es posible realizar el registro de datos de consumo de la aplicación, pero es recomendable usar un dispositivo físico debido a la integración de la interfaz Bluetooth, en particular Bluetooth Low Energy (BLE) ya que la aplicación móvil de estudio utiliza el

sensor embebido en el dispositivo móvil. Es preciso tomar consideraciones en la versión del sistema operativo Android ya que BLE es compatible en versiones superiores a Android 4.3 (API nivel 18).

Una vez instalada la aplicación móvil en el celular se debe realizar ciertas configuraciones para habilitar las opciones de desarrollador y depuración por USB [18], para lo cual dirigirse a *Ajustes*, luego a *Acerca del teléfono* y tocar siete veces el campo de *Número de compilación*. Después de realizar lo antes mencionado dirigirse a Ajustes y aparecerá *Opciones de desarrollador*. Para habilitar la depuración por USB dirigirse a *Opciones de desarrollador* y activar *Depuración de USB*.

Luego en la computadora con sistema operativo Windows 10 se instala ADB (Android Debug Bridge) el cual es una herramienta para comunicarse con el dispositivo móvil Android mediante la línea de comandos del sistema operativo Windows 10.

Por último, se debe conectar el dispositivo móvil a la computadora usando el cable USB (Universal Serial Bus) y entrar a la línea de comandos para comprobar el funcionamiento de ADB.

2.2.2 Desarrollo de un script para medir el uso de RAM Y CPU

El script desarrollado usa el lenguaje interpretado Python, este lenguaje de programación posee muchas librerías para la visualización de datos estadísticos. En este caso la librería a usar para lograr la visualización de los recursos usados por la aplicación móvil es matplotlib [17]. Se muestran gráficas de memoria RAM y CPU versus tiempo en la figura 2.2 y figura 2.3 respectivamente, mostrando en tiempo real el comportamiento de la aplicación móvil sobre los recursos de hardware que posee el dispositivo móvil del cliente o usuario, por medio de esta visualización de los recursos de memoria RAM y CPU se tendrá una perspectiva gráfica del funcionamiento de la aplicación ASI Ecuador, en este caso cuando se navega por la aplicación móvil durante 13 segundos. Además de mostrar las gráficas, el script muestra resultados estadísticos como el promedio de recursos usado durante cierto tiempo y los valores máximos y mínimos que ocupa la aplicación móvil objetivo.

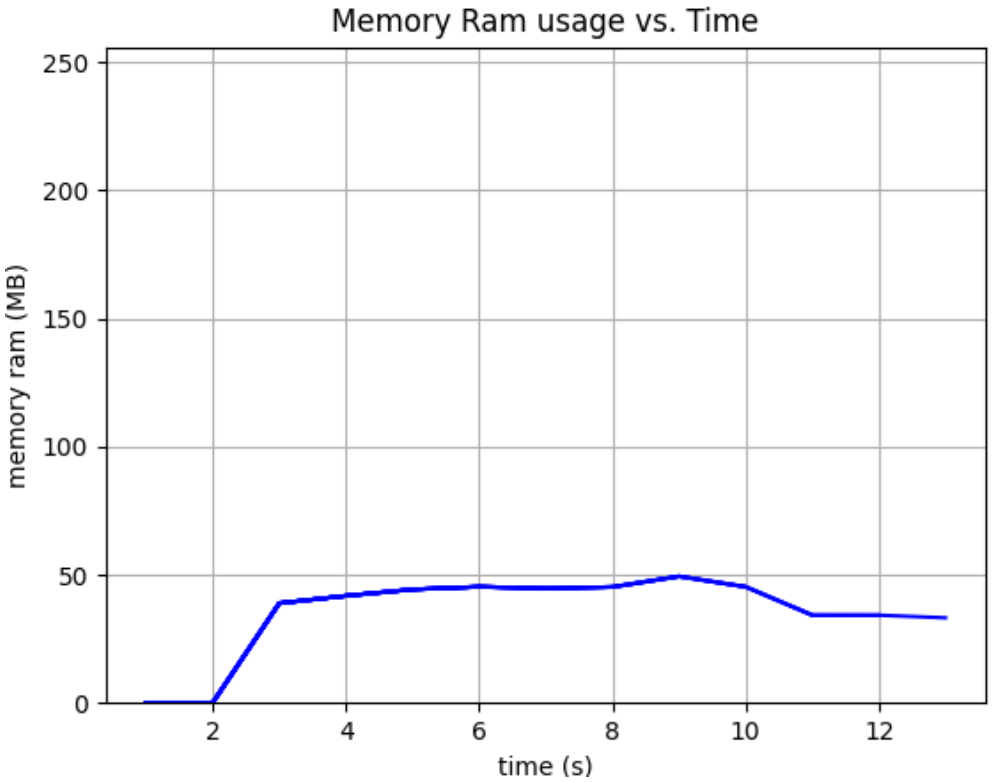


Figura 2.1. Consumo de memoria RAM de ASI Ecuador

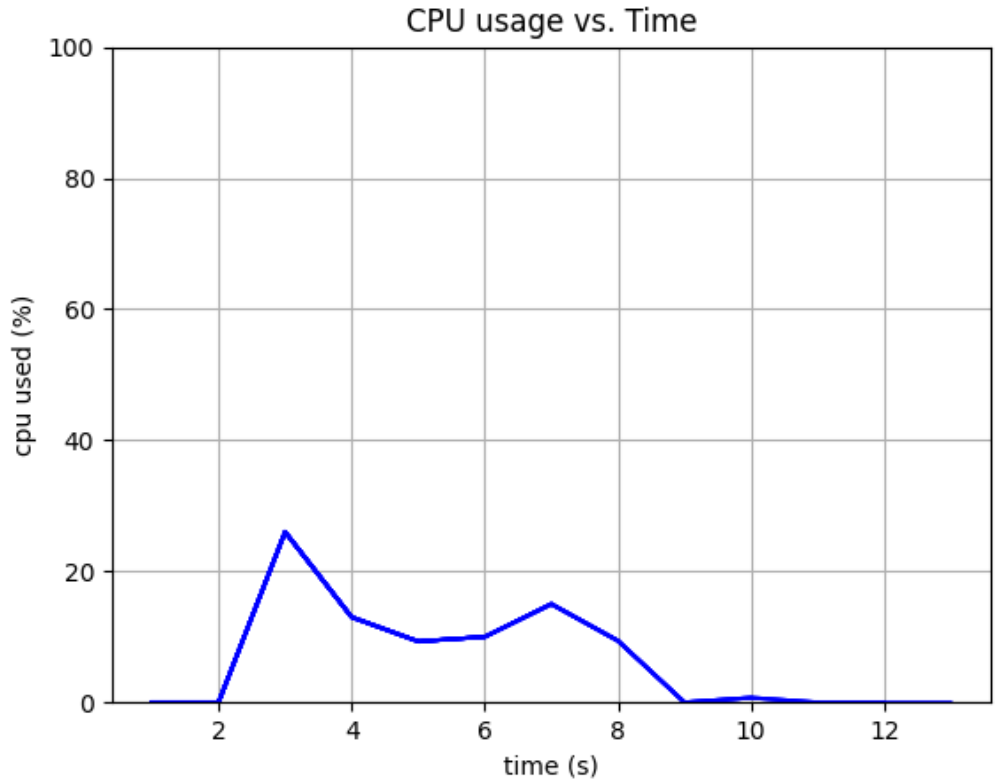


Figura 2.3. Consumo de CPU de ASI Ecuador

Existe una librería para Python que permite crear una instancia de la herramienta ADB y así usarla en el script, esta librería se denomina pure-python-adb [19] cuyo propósito es ejecutar los comandos ADB mediante codificación usando el lenguaje de programación Python.

Cabe resaltar que ADB posee tres componentes para su funcionamiento los cuales son cliente, daemon (adbd) y servidor. El cliente se ejecuta en la computadora enviando comandos, el daemon es el encargado de ejecutar los comandos en un dispositivo móvil encontrándose como un proceso de segundo plano y el servidor es el encargado de dirigir la comunicación entre el cliente y el daemon [20].

En la figura 2.4 se muestra el esquema para establecer comunicación y empieza cuando el cliente comprueba si el servidor está en ejecución, caso contrario, inicia inmediatamente el proceso de servidor vinculando a la vez una conexión TCP mediante el puerto 5037 de manera local. Luego el servidor establece conexión por vía USB con el daemon que se ejecuta en segundo plano en el celular con sistema operativo Android y de esta manera el servidor dirige la comunicación entre el cliente y el daemon.

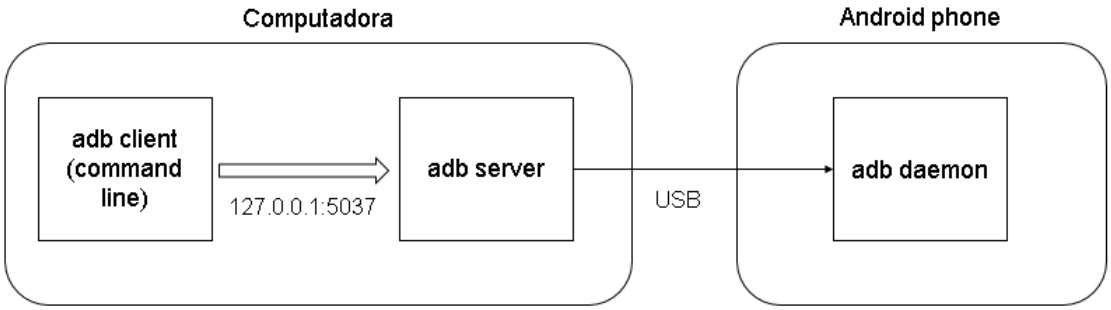


Figura 2.4. Diagrama de la comunicación computadora - Android phone usando ADB [19]

En general, el script descrito recolectará datos de memoria RAM y CPU por medio de la librería pure-python-adb y mostrará las gráficas estadísticas antes mencionadas y para iniciar este proceso se usa la línea de comandos para la ejecución del script.

2.2.3 Obtención del uso de batería

Para verificar el uso de batería por parte de la aplicación móvil ASI Ecuador, se usó el registro de uso de batería que tiene el dispositivo móvil, en el caso del dispositivo móvil que se usó para todas las pruebas realizadas el cual es un Samsung J2 Prime con sistema operativo Android versión 6.0 se presenta a continuación en la figura 2.5 el uso de batería por parte de la aplicación móvil ASI Ecuador y el componente de Bluetooth.

Figura 2.5. Uso de batería por las aplicaciones móviles e interfaces

Por medio de “Ajustes”, luego en “Mantenimiento dispositivo”, luego en “”Batería” y por último “Uso de batería” se verificará los datos de uso de bateria por algunas aplicaciones y componentes del dispositivo móvil. Cabe mencionar que este registro se lleva cabo desde la ultima vez que se cargó la batería ya que al volver a cargar de nuevo la batería, todo este registro se borra para realizar otro registro de uso de batería y toma aproximadamente un día para que muestre el nuevo registro.

2.2.4 Obtencion del uso de red

Por medio de esta prueba evidenciará cuántos datos de red usa la aplicación móvil ASI Ecuador y en qué momentos de su funcionalidad usa estos datos.

Se aplicará la siguiente arquitectura de análisis de la comunicación cliente-servidor observada en la figura 2.6 para obtener los datos de red con la diferencia de examinar la información mediante Wireshark que es una herramienta de análisis de tráfico.

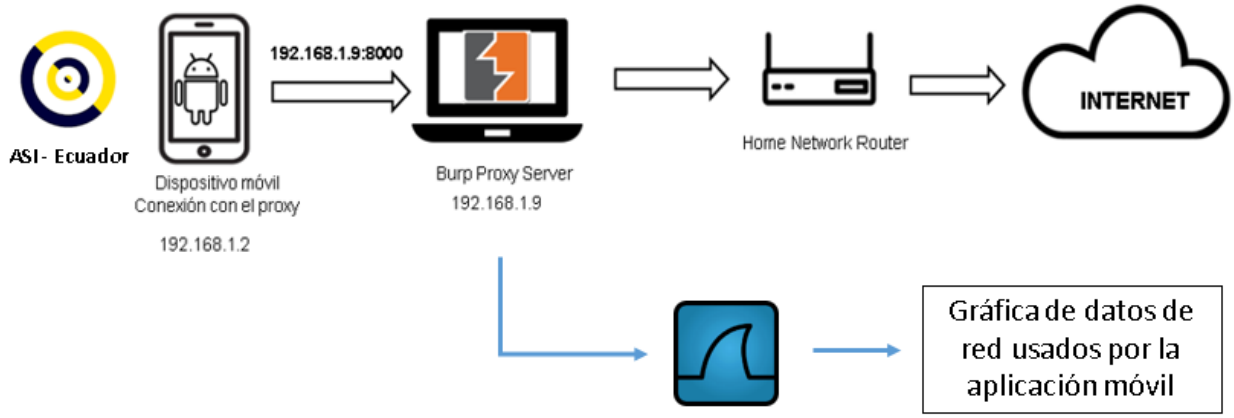


Figura 2.6. Arquitectura usada para el análisis de la comunicación cliente-servidor

2.3 Pruebas de caja blanca

La figura 2.7 muestra el funcionamiento de ASI, cuando los usuarios se encuentran cerca, intercambian un código aleatorio de manera automática a través de Bluetooth (1). Luego, dado el caso de que una de estas personas contrajo el virus, éste lo notifica a la autoridad de salud gubernamental (Ministerio de Salud Pública para Ecuador) y ellos ingresan su código de autorización en la base de datos del servidor en la nube, el cual está basado en el protocolo open-source DP3T.

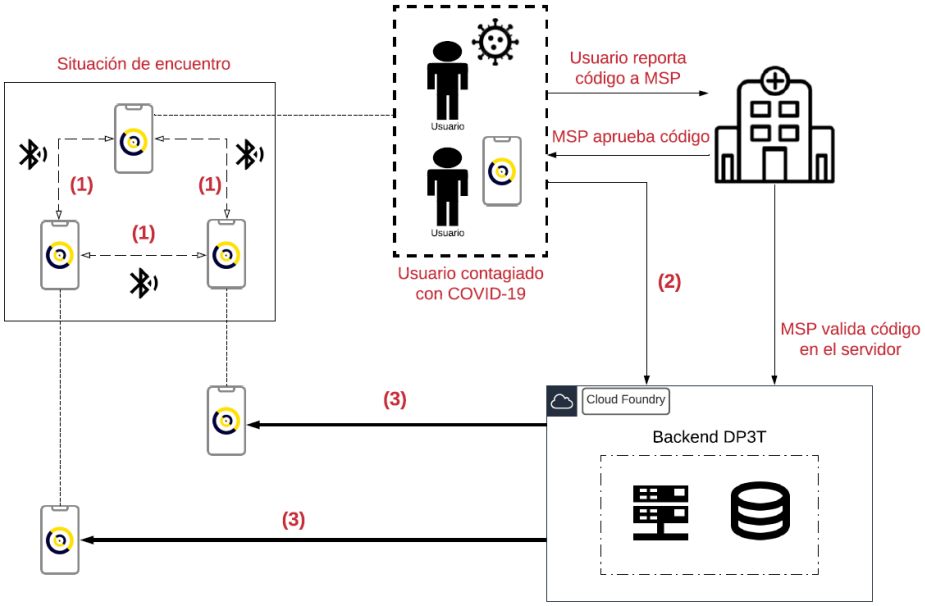


Figura 2.7. Funcionamiento de la aplicación móvil ASI Ecuador

Finalmente, el usuario contagiado procede a ejecutar la opción (2) para que el servidor notifique a los contactos involucrados (3) y estos tomen las debidas precauciones.

Mencionado lo anterior, las pruebas de caja blanca consistieron en adaptar, tanto el Frontend como el Backend de ASI, para que funcionen en un entorno local y luego estudiar, primero, el proceso de generación e intercambio de datos entre los dispositivos mediante Bluetooth, y segundo, las opciones que ofrece la aplicación para reportarse como COVID positivo y notificar de un posible contagio a los contactos.

Para realizar lo dicho recientemente, fueron necesarias dos herramientas de desarrollo de software como son Android Studio y Spring Tool Suite, las cuales se aprecian en la figura 2.8. En el caso de la primera, se la utilizó para el análisis del frontend de la aplicación, mientras que la segunda se la empleó para desplegar el backend como un servidor local, que atienda los requerimientos del usuario.

Figura 2.8. Herramientas de desarrollo de software

2.3.1 Diagrama de despliegue

El Lenguaje Unificado de Modelado (UML por sus siglas en inglés) permite construir de manera gráfica un sistema de software. Dentro de su variedad, ofrece un esquema en el que se visualizan los procesos en tiempo de ejecución que intervienen en cada dispositivo de hardware y cómo interactúan, demonimado diagrama de despliegue. Esto, aplicado al entorno simulado, se lo observa en la figura 2.9. En la izquierda, los dispositivos Android que utilizan los dos perfiles de usuario, intercambian sus códigos aleatorios mediante Bluetooth gracias al protocolo GAEN que usa la aplicación ASI Ecuador, y se comunica con el servidor local a través de solicitudes HTTP para los procesos de notificaciones de exposición al virus COVID-19. Dicho servidor se aloja en una máquina con distribución Linux (Ubuntu) y utiliza el framework Spring Boot de Java para comunicar la API con la base de datos PosgreSQL.

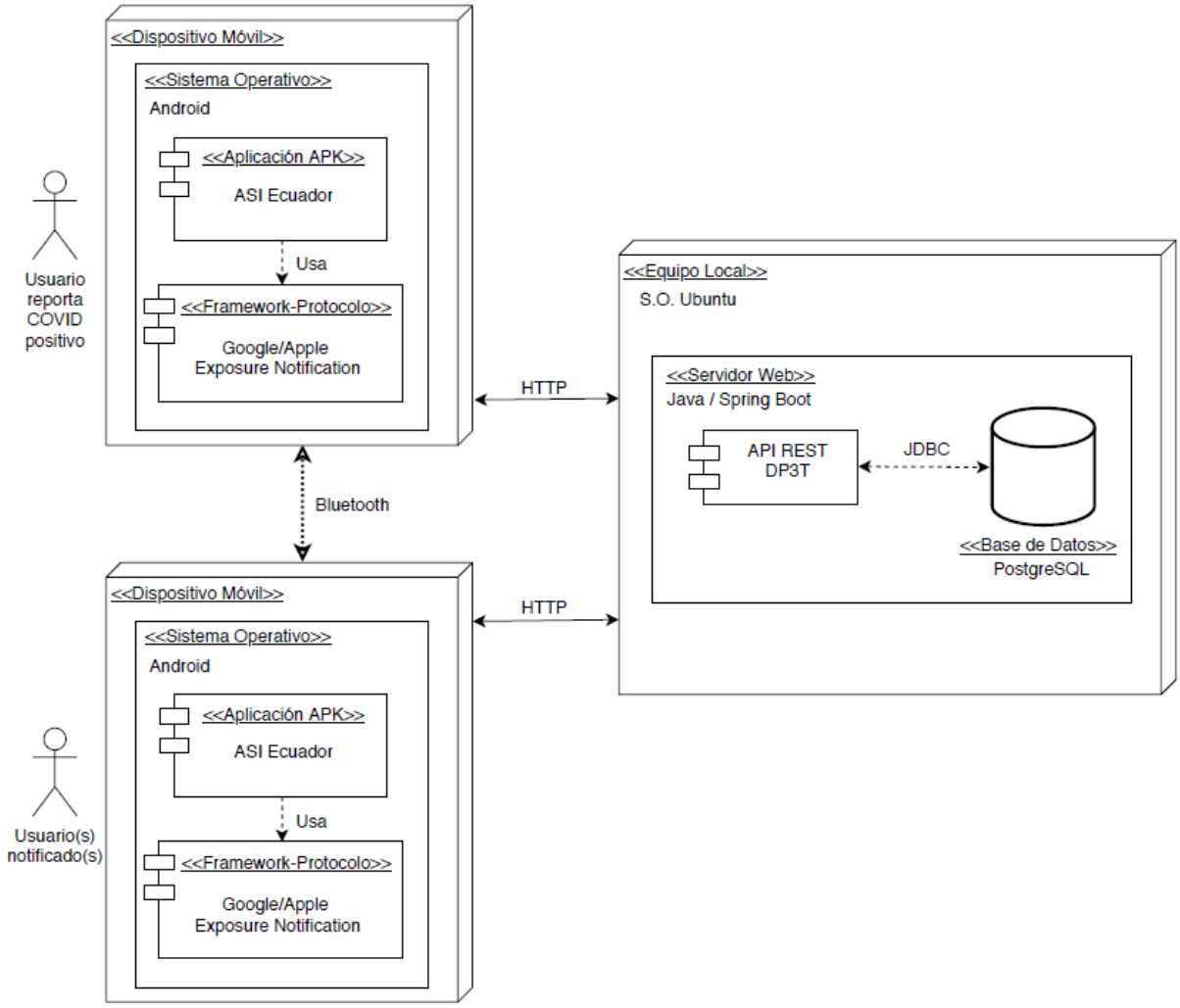


Figura 2.9. Diagrama de despliegue

2.3.2 Arquitectura de servicios

En la figura 3.20 se observan los componentes principales del frontend y backend de la aplicación réplica. Del lado del frontend se tienen solo dos módulos, el primero con el nombre APP donde se encuentran todas las clases concernientes con la apariencia de ASI Ecuador, tales como activities, fragments, layouts, views, strings, etc. El segundo módulo, llamado DP3T, cuenta con todas las clases para interactuar con la interfaz Bluetooth, generar las llaves aleatorias (las cuales se intercambian con los otros dispositivos), acceder a la base de datos SQLite para guardar los contactos, realizar las llamadas HTTP que se comunican con el servidor, entre otras.

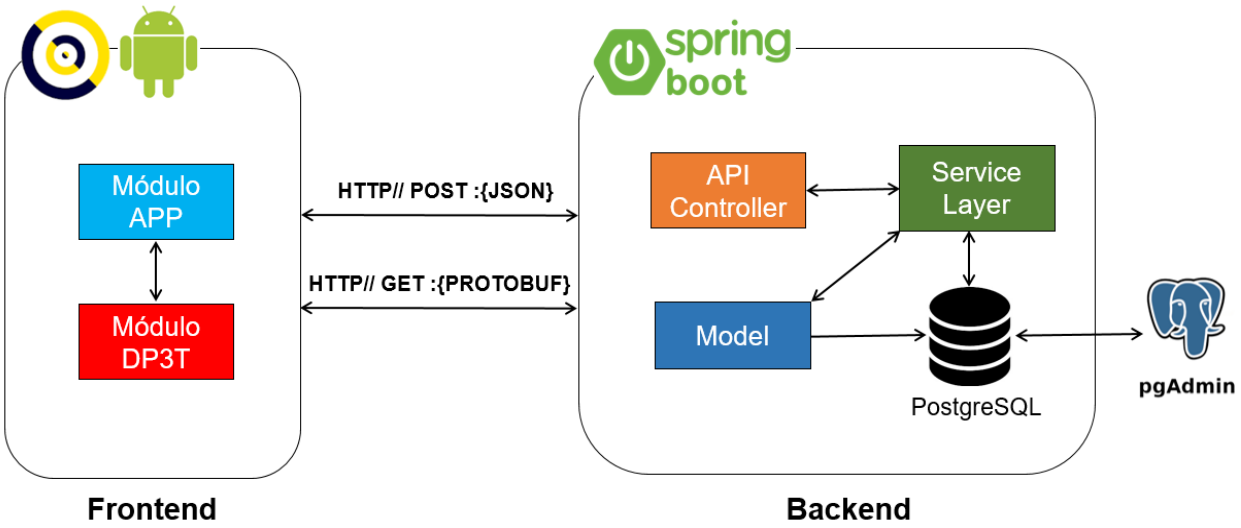


Figura 2.10. Frontend y Backend de la aplicación réplica

Por otro lado, el backend se lo implementó bajo el framework Spring Boot de Java, que cuenta con 3 capas, la capa API Controller, que contiene todas las rutas que atienden a los requerimientos del cliente, la capa Model, donde se define la estructura de los datos que se intercambian con el frontend, la capa Service, donde se encuentran las funciones para interactuar con la base datos, la cual es de tipo PostgreSQL, y donde su administración para validar a los usuarios que se reportan como COVID positivo, se la hace mediante la interfaz web pgAdmin.

2.4 Pruebas no-funcionales

Las pruebas no funcionales son un tipo de prueba de software que permite analizar varios aspectos referentes al software o aplicación móvil tales como rendimiento, usabilidad y seguridad. Este tipo de pruebas son muy importantes ya que demostrarían los efectos negativos que producirían en el cliente al usar la aplicación móvil o software en cuestión.

Para este caso se realizarán pruebas de rendimiento, usabilidad y comunicación cliente-servidor, todos estos aspectos son importantes y es preciso analizarlos por medio de las técnicas antes mencionadas.

2.4.1 Usabilidad

En lo que respecta a la usabilidad de la aplicación móvil ASI Ecuador, se detallan mediante las siguientes gráficas la interfaz de usuario por cada funcionalidad que realiza la aplicación y de esta manera determinar si es o no fácil de usar.

La figura 2.11 se muestra información de la aplicación móvil ASI Ecuador cuando se inicia por primera vez, por medio de los números ubicados en la parte superior dan el orden respectivo donde el número uno menciona sobre las funciones que realiza la aplicación referente a anticipar a los usuarios sobre el virus, el numero dos menciona sobre la forma de protección de datos garantizando la privacidad, el número tres hace hincapié de las tecnologías usadas para la detección de contactos y de los datos que

solicita al usuario y por último el número cuatro comunica a los usuarios sobre la manera de notificar sobre algún posible contagio.

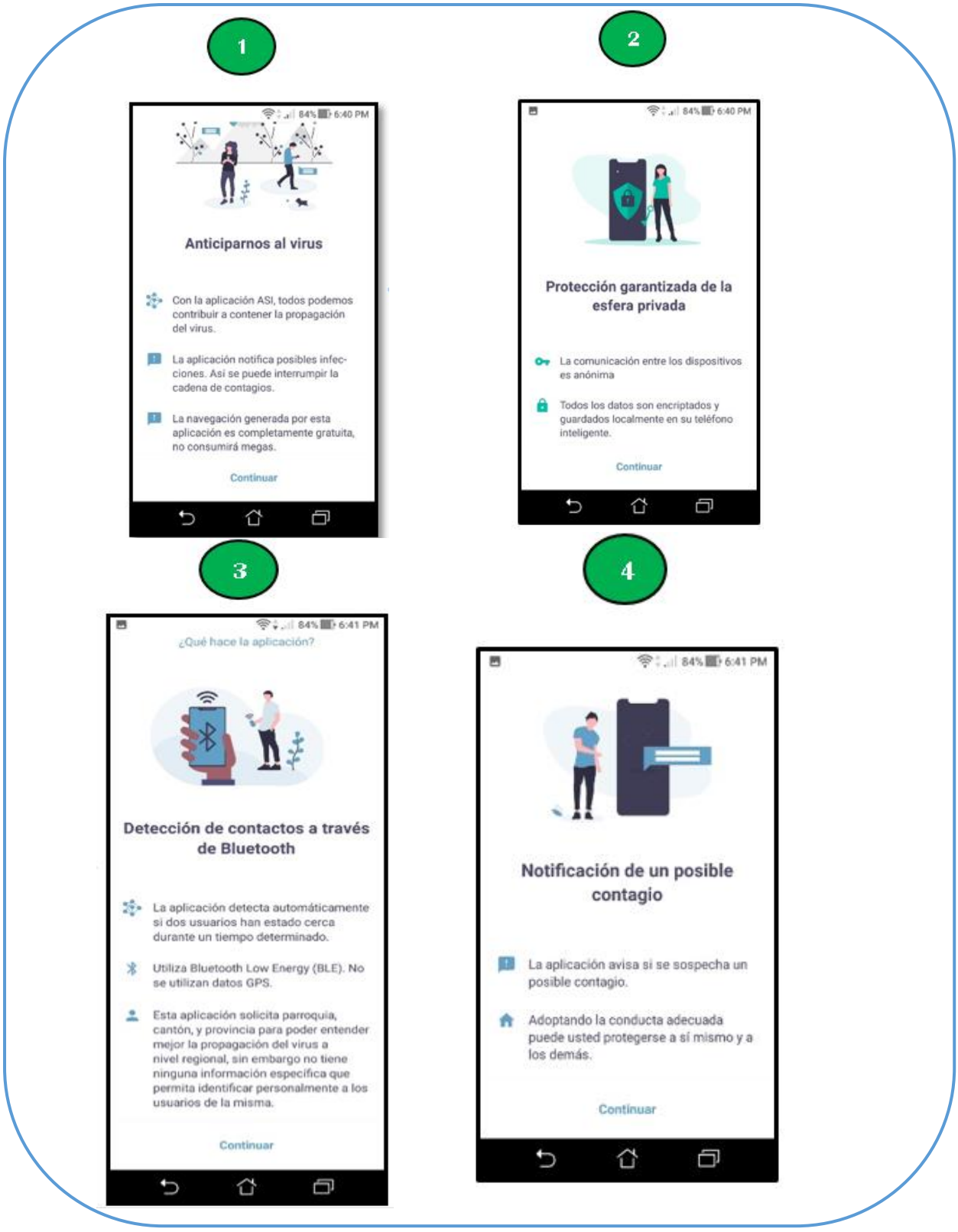


Figura 2.11. Información introductoria de ASI Ecuador

La figura 2.12 se muestra los permisos que el usuario debe otorgar a la aplicación móvil para que esta funcione correctamente, entre las cuales destacan el ignorar optimización de la batería, activación tanto del seguimiento de contactos donde usa Bluetooth Low Energy y la activación de configuración de la ubicación del dispositivo en la cual recalca que no se compartirá la ubicación del dispositivo recopilada por el GPS.

Figura 2.12. Permisos que se deben otorgar a ASI Ecuador

En la figura 2.13 muestra un formulario donde el usuario debe elegir la provincia, cantón y ciudad de residencia, además de la interfaz principal de ASI Ecuador.

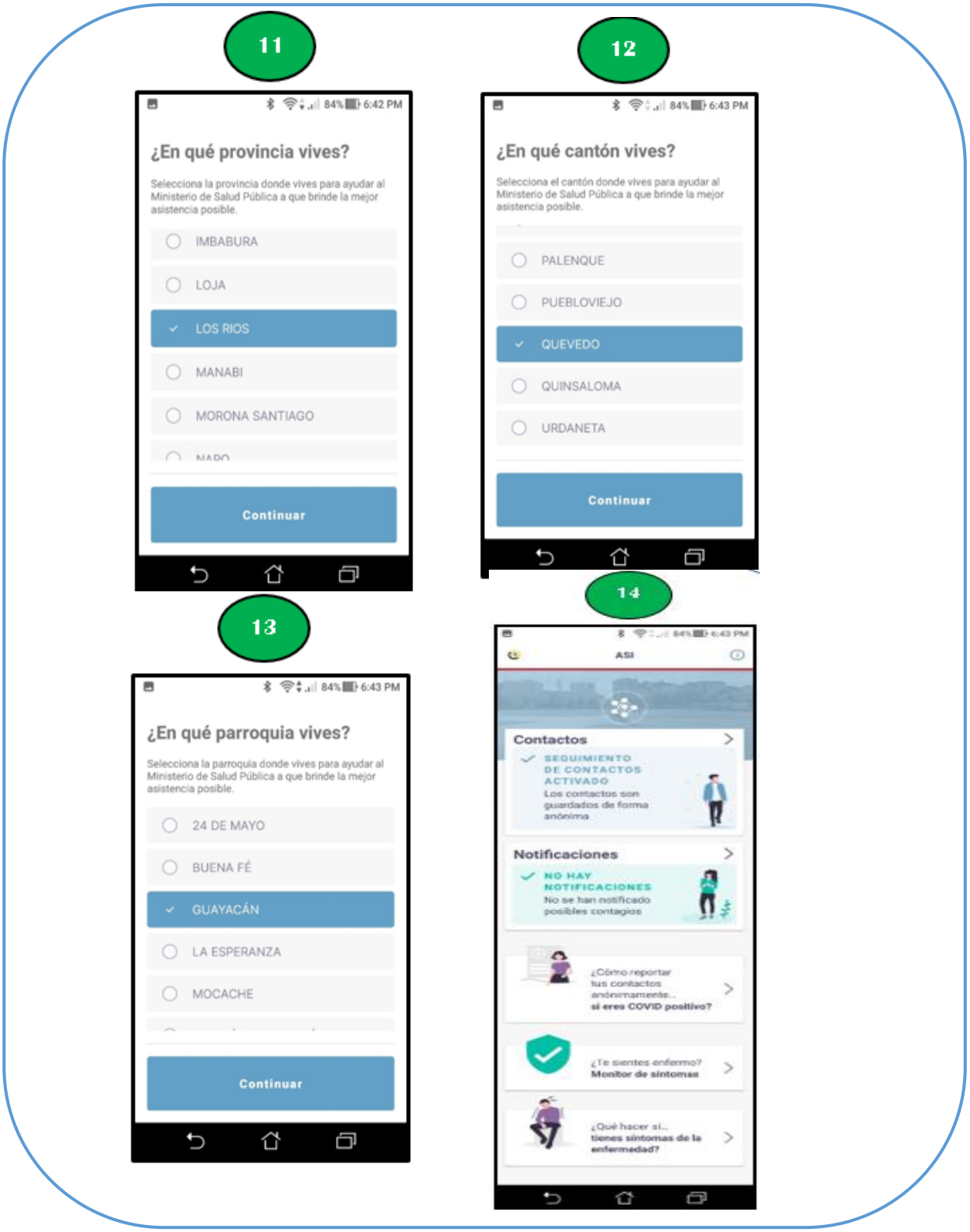


Figura 2.13. Ingreso de ubicación y ventana principal de ASI Ecuador

La interfaz principal de la aplicación móvil ASI Ecuador consta de cinco ítems en la cual el tercer ítem permite reportar anónimamente si el usuario es COVID positivo, en la figura 2.14 se muestra el proceso de notificación donde el cuarto paso corresponde a la acción de compartir el código COVID ya cuando este ha sido validado por una entidad médica autorizada.

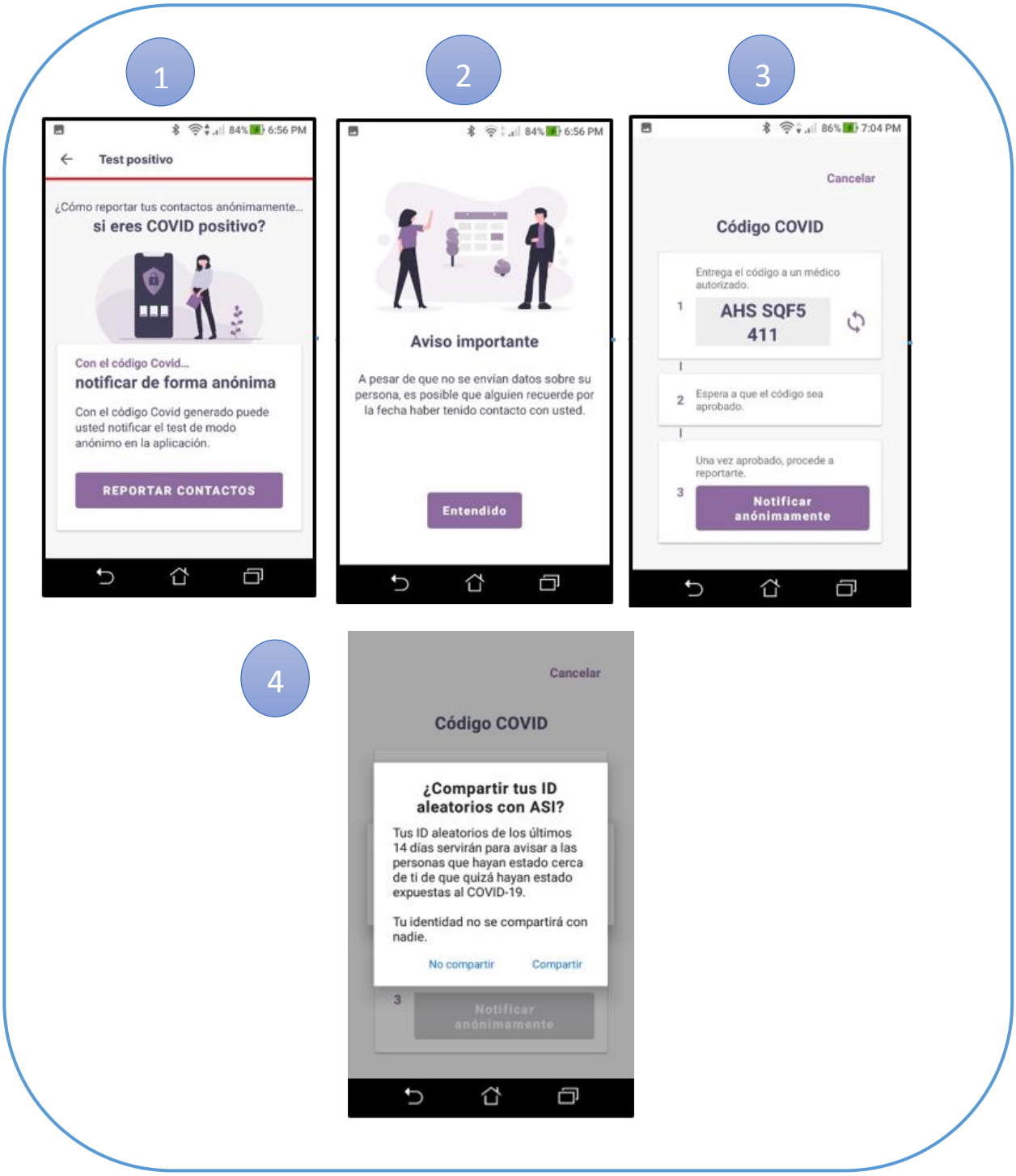


Figura 2.14. Notificación COVID positivo

Luego de compartir el código COVID ya autorizado, se muestran en la figura 2.15 avisos sobre agradecimiento por combatir el COVID, información sobre la culminacion de seguimiento de contactos luego de dar positivo y sobre instrucciones para autoaislarse.

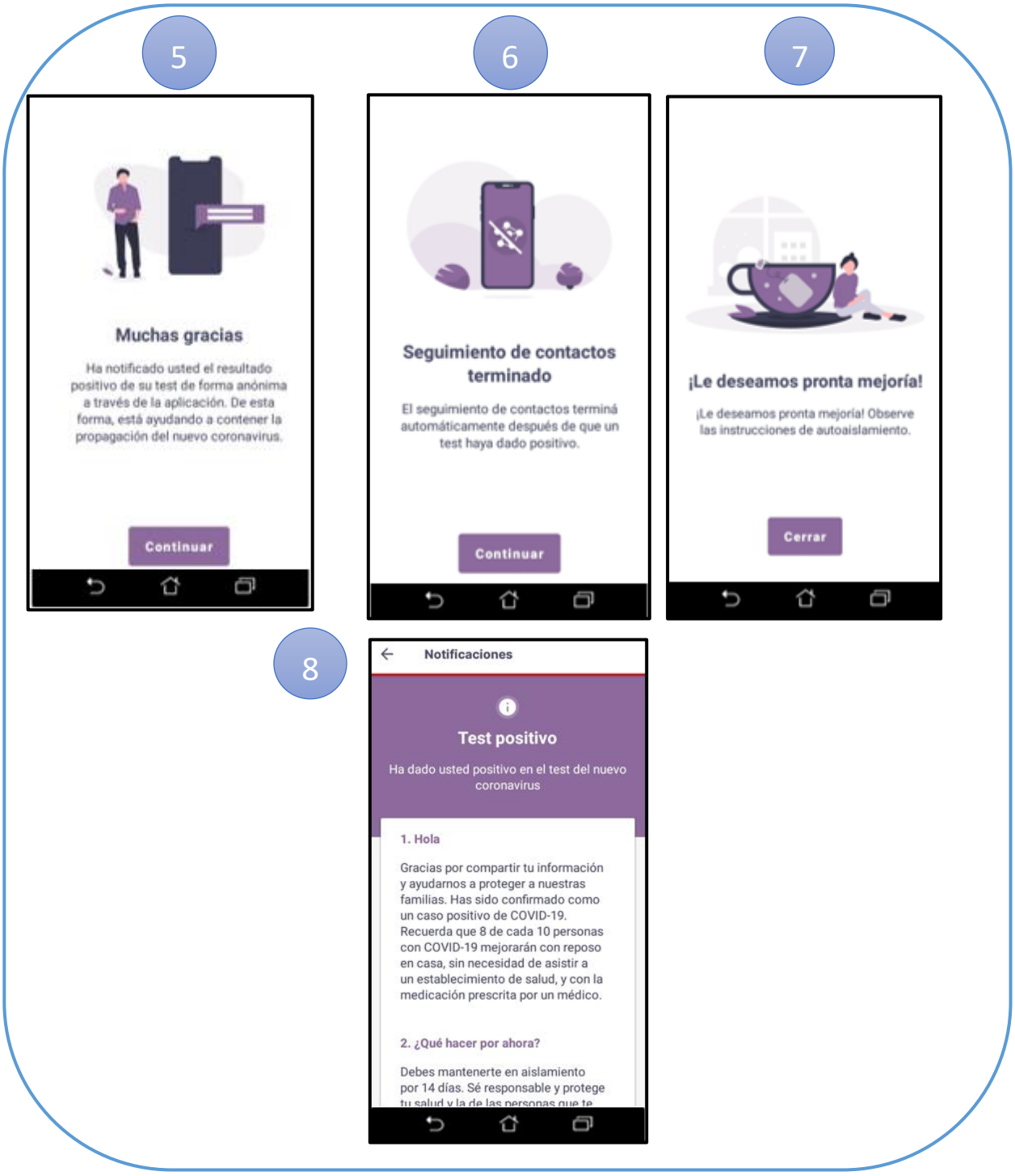


Figura 2.15. Finalización del proceso de notificación

Una vez que el usuario es positivo de COVID , se enviarán notificaciones a las personas que estuvieron cerca de este para que tengan precauciones. En la interfaz principal de la aplicación móvil ASI Ecuador del usuario positivo de COVID se muestra un mensaje donde menciona que el seguimiento de contactos terminó por lo cual no se guardarán mas datos, como el la figura 2.16.

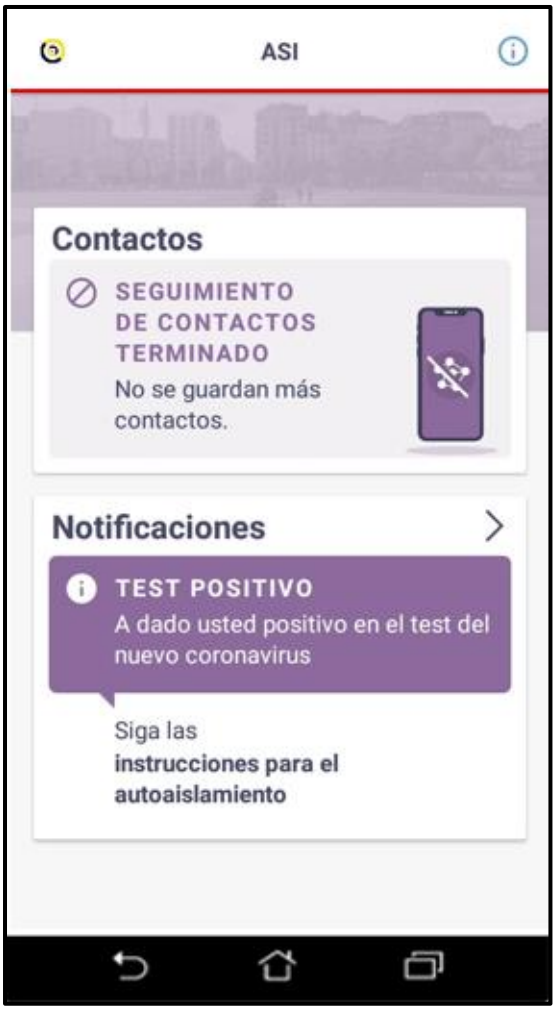


Figura 2.16. Ventana principal luego de reportar COVID positivo

El cuarto ítem permite registrar los síntomas que posee el usuario día a día, en la figura 2.17 muestra la opción de monitor de síntomas en donde pregunta al usuario sobre los síntomas principales del COVID y este debe elegir de acuerdo a su estado de salud en el que se encuentra en ese día, luego almacena estos datos y los coloca en un historial de síntomas mostrando la fecha en la que registró.

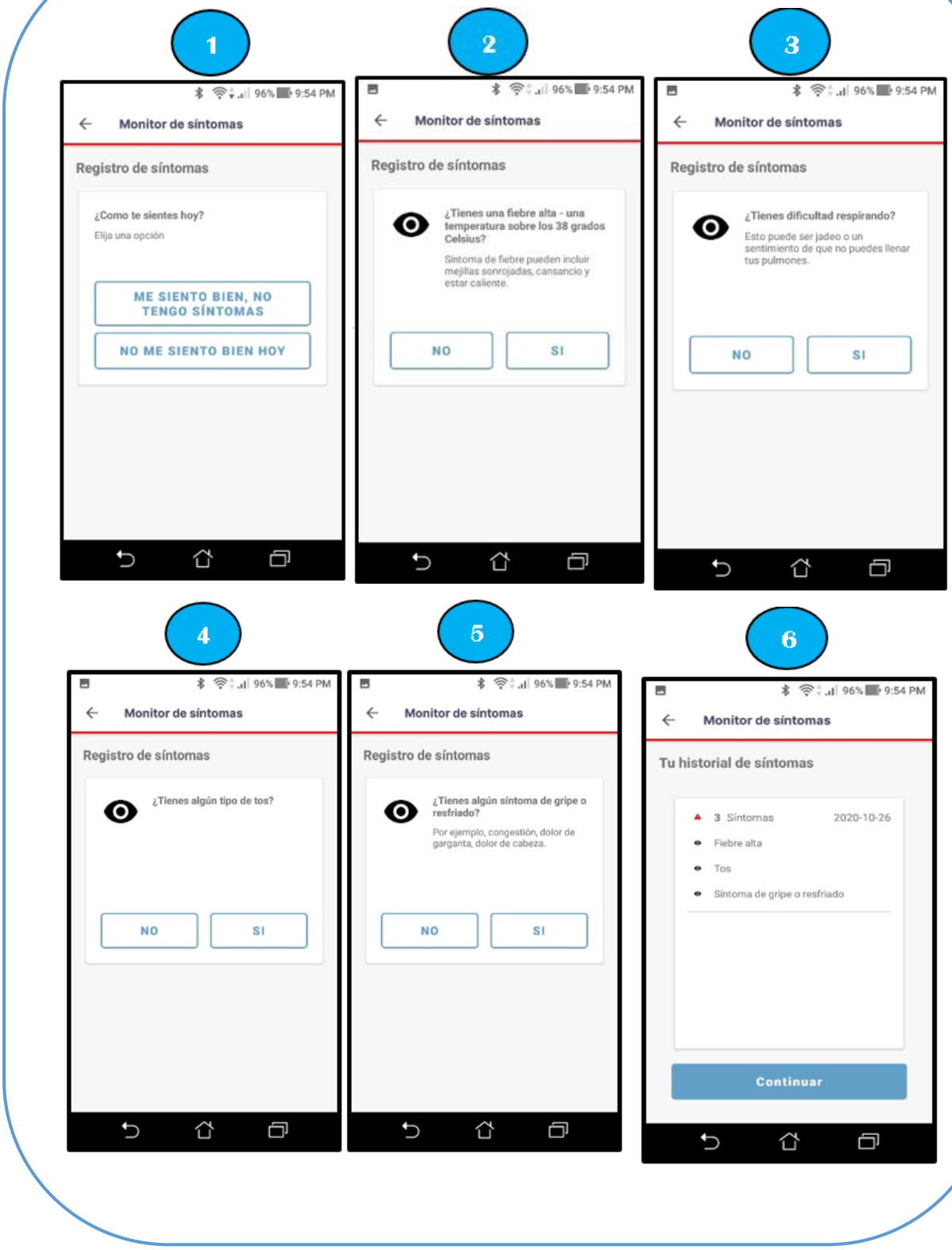


Figura 2.17. Opción Monitor de síntomas

La opción de notificaciones la cual corresponde el segundo ítem muestra las notificaciones de posibles contactos además sobre información adicional que complementan sobre normas vigentes ante el COVID como se muestra en la figura 2.18.

Figura 2.18. Opción Notificaciones

Además, en este ítem hay un cuadro denominado “MÁS INFORMACIÓN” el cual redirige a la página https://www.coronavirusecuador.com/, esta página contiene información sobre estadísticas, información del Ministerio de Salud, etc.

En la figura 2.19 se observa un resumen respecto al beneficio que brinda la aplicación móvil ASI Ecuador, además de los permisos que debe otorgarse para que la aplicación móvil funcione correctamente en el dispositivo Android y sobre lo que el usuario debe seleccionar en torno al lugar de residencia para que la aplicación móvil mencionada funcione correctamente, demostrando que es muy intuitiva y fácil de usar.

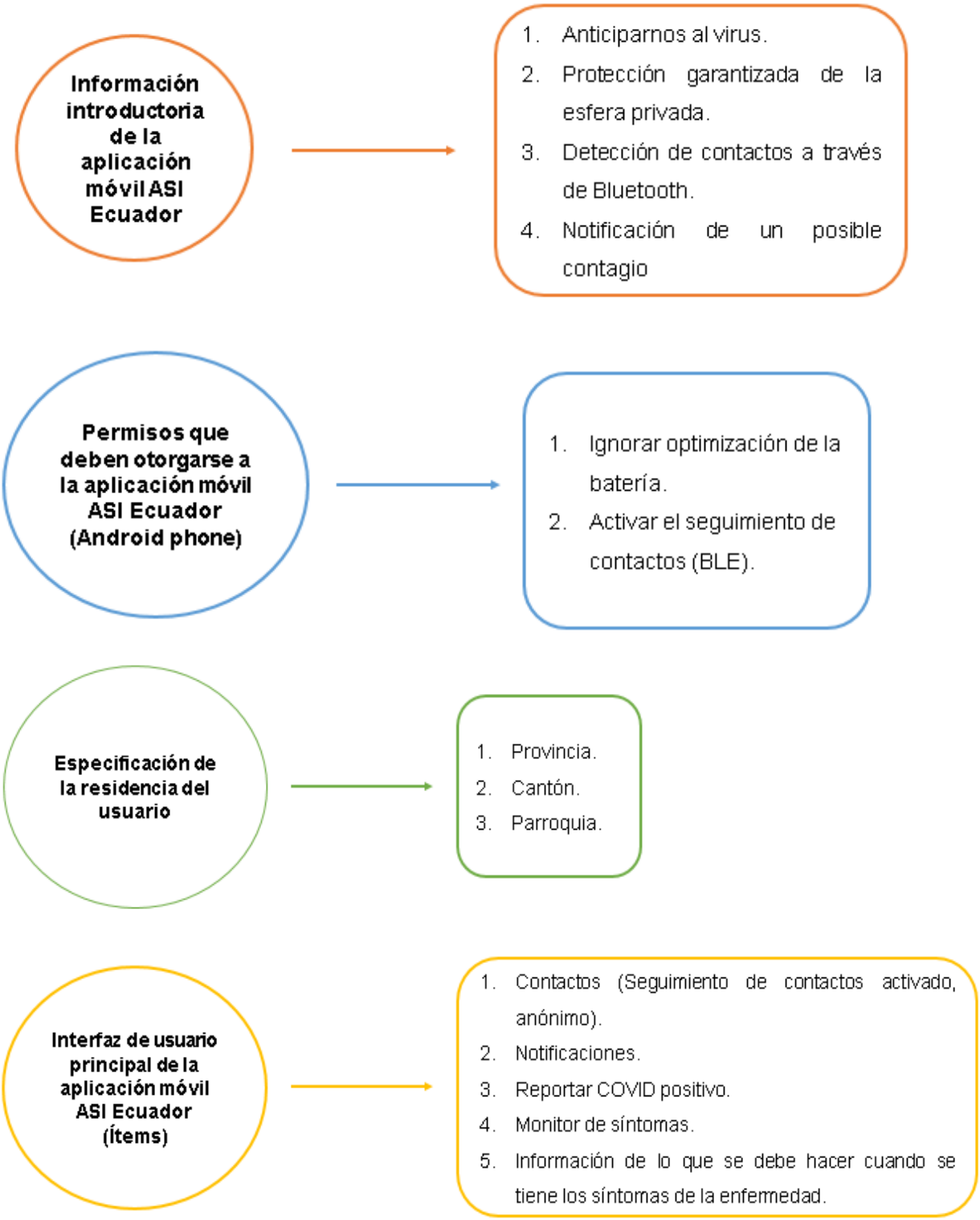


Figura 2.19. Resumen de usabilidad de ASI Ecuador

2.4.2 Rendimiento

Las pruebas de rendimiento en el caso de las pruebas de caja negra están enmarcadas solamente en el lado del cliente, es decir, el uso de la aplicación móvil ASI Ecuador instalada en el celular. Por medio del script desarrollado en Python se obtendrá gráficas en tiempo real del uso de la memoria RAM y porcentaje de CPU además del promedio y valores máximos y mínimos obtenidos en el transcurso de la prueba.

Para cada prueba de rendimiento de CPU y memoria RAM se midió el tiempo que dura el usuario en anotar los síntomas, notificar que tiene COVID, navegar por cada ítem y cuando la aplicación móvil se encuentra en segundo plano y con este tiempo obtenido se introduce en el script para realizar la prueba.

2.4.3 Comunicación cliente-servidor

Con respecto a la prueba de seguridad se realizará un pentesting a la aplicación móvil ASI Ecuador en lo que respecta al análisis de la comunicación cliente servidor o análisis de tráfico el cual es parecido a un ataque de “Man in the Middle” (MitM) [21] que permite ser intermediario para intervenir en el tráfico de datos permitiendo ver la comunicación hasta incluso modificar cierta parte de esta.

Se usará una distribución de Debian basada en GNU/Linux denominada Kali Linux, este es un sistema operativo que posee herramientas para la seguridad informática ofensiva. Para llevar a cabo esta función se usará la herramienta Burp Suite [22] que viene integrada en Kali Linux, esta herramienta posee un proxy por el cual se interceptará el tráfico de la aplicación móvil ASI Ecuador y el servidor.

Cabe mencionar que en el dispositivo móvil se debe configurar para que este envíe el tráfico de la aplicación móvil al proxy de Burp Suite, para ello se debe colocar en proxy manual e ingresar la dirección IP de la máquina con sistema operativo Kali Linux además del puerto de escucha que permite la conexión hacia el proxy de Burp Suite Server como se observa en la figura 2.20.

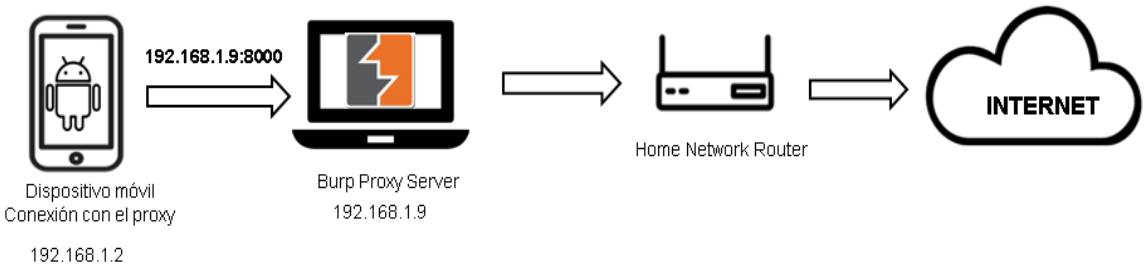


Figura 2.20. Análisis de seguridad básica

Se debe instalar un certificado de red en el celular, para ello se descarga el certificado desde la página de Burp ingresando la dirección http://burp, luego se debe cambiar la extensión del certificado por .cer ya que viene con extensión .der por defecto.

Con esta prueba se podrá corroborar si la aplicación móvil se fundamenta en la anonimidad que menciona al iniciar en la parte de datos informativos.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

En esta sección se mostrará el respectivo análisis enfocado a los dos tipos de pruebas ejecutadas, tales como pruebas de caja negra y blanca, luego de cada análisis se originarán los respectivos resultados que describirán el comportamiento de la aplicación móvil ASI Ecuador en torno a su rendimiento y estructura.

3.1 Análisis de las pruebas de caja negra

A continuación, se detalla el análisis realizado cuando la aplicación móvil ASI Ecuador usa memoria RAM, CPU, batería, red, bluetooth y sobre la comunicación cliente-servidor.

3.1.1 Análisis del uso de memoria RAM

Al ejecutar el script en Python para recolectar los datos de memoria RAM, se tomó en cuenta un tiempo promedio que le llevaría al usuario realizar cada funcionalidad en la aplicación móvil ASI Ecuador, en este caso se eligió las funcionalidades de notificar código COVID, anotar síntomas, aplicación móvil en background y navegar por cada ítem de la aplicación móvil objetivo y así ver cuando varía el uso de memoria RAM en el tiempo.

Para cada funcionalidad se eligió un tiempo promedio en el cual el usuario tardaría para llevar a cabo dicha funcionalidad, el tiempo ocupado para realizar cada funcionalidad se lo capturó mediante un cronómetro haciendo 10 pruebas en total y luego se promedió el tiempo que se usaría para ejecutar cada funcionalidad, tiempo utilizado en el script en Python para obtener la serie de tiempo de memoria RAM usado y promedio, así como también valores máximos y mínimos de memoria RAM registrados en el transcurso del tiempo.

Luego de tener el tiempo para colocarlo en el script se realizaron 20 ejecuciones de este capturando los datos mencionados anteriormente, pero se enfocó en el resultado del promedio de memoria RAM usado para lo cual se promediaron todos los 20 promedios de memoria RAM usados.

La figura 3.1 corresponde a una de las gráficas de uso de memoria RAM cuando el usuario notifica el código COVID, a esta funcionalidad se le asignó un tiempo promedio de 11 segundos para que el usuario complete la acción.

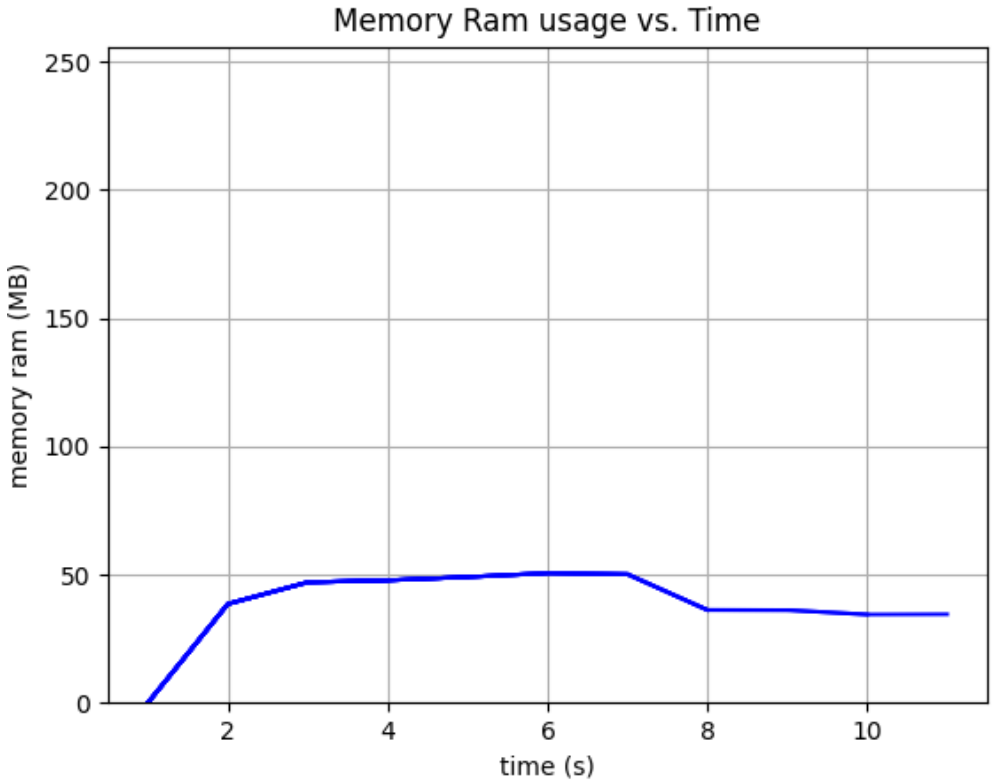


Figura 3.1. Uso de memoria RAM cuando se notifica el código COVID

La figura 3.2 corresponde a una de las gráficas de uso de memoria RAM cuando el usuario anota sintomas de la enfermedad, a esta funcionalidad se le asignó un tiempo promedio de 13 segundos para completar la acción

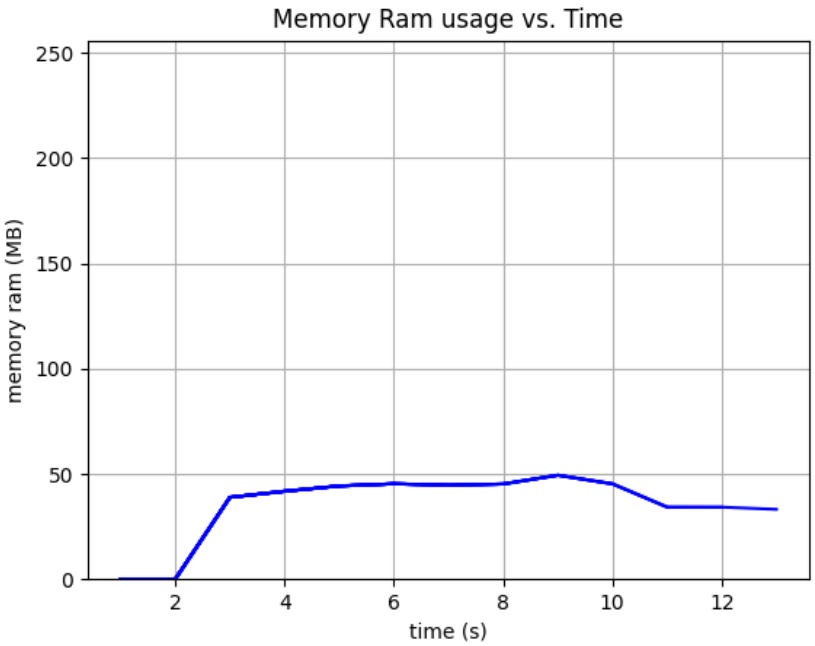


Figura 3.2. Uso de memoria RAM cuando se anota síntomas

La figura 3.3 corresponde a una de las gráficas de uso de memoria RAM cuando el usuario navega por cada item que se encuentra en la pantalla principal de la aplicación móvil ASI Ecuador, se asignó un tiempo promedio de 10 segundos para completar la acción.

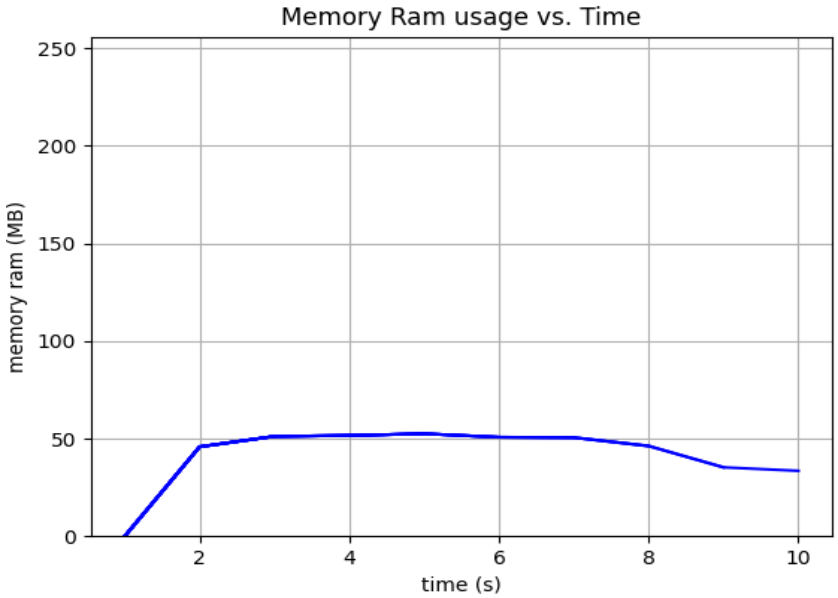


Figura 3.3. Uso de memoria RAM cuando se navega por la aplicación móvil

La figura 3.4 corresponde a una de las gráficas de uso de memoria RAM cuando la aplicación móvil esta en background, a esta parte se asignó un tiempo promedio de 10 segundos.

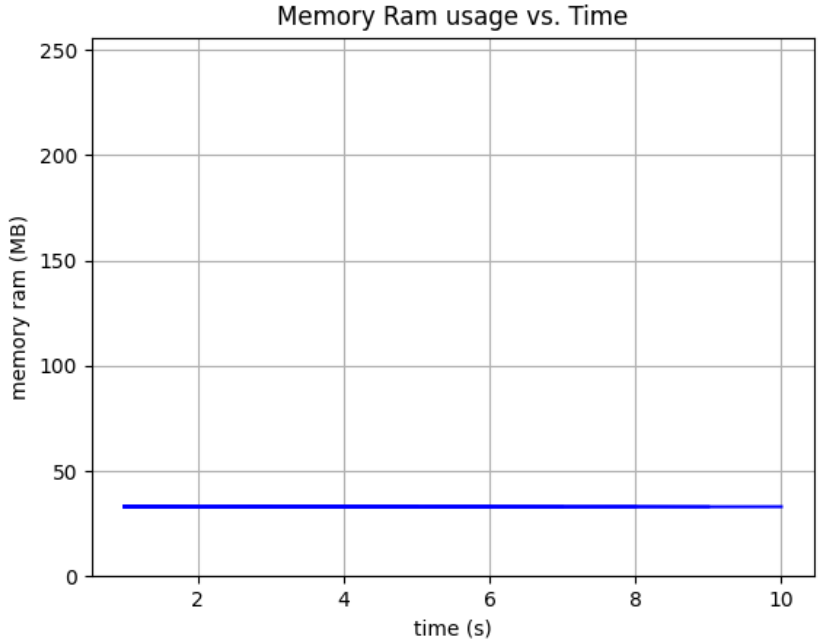


Figura 3.4. Uso de memoria RAM cuando la aplicación móvil está en background

Se puede notar que en cada figura vemos un comportamiento alcista, bajista, constante y no constante, y esto da a entender que la aplicación movil ASI Ecuador tiene un incremento de uso de memoria RAM cuando un segmento de la serie de tiempo tiene comportamiento alcista y de decremento cuando es de comportamiento bajista, además cuando la aplicación móvil esta en background se ve un comportamiento constante de inicio a fin en el uso de memoria RAM y en las otras funcionalidades se ve variaciones suaves comprendidas entre un limite mínimo de 0 MB y como limite máximo aproximado de 50MB.

3.1.2 Análisis del uso de CPU

De la misma manera se usaron las funcionalidades ya mencionadas anteriormente y los respectivos tiempos promedio obtenidos para realizar cada funcionalidad pero enfocandose en obtener el promedio de CPU usado.

La figura 3.5 corresponde a una de las gráficas de uso de CPU cuando el usuario notifica el código COVID, a esta funcionalidad se le asignó un tiempo promedio de 11 segundos para que el usuario complete la acción.

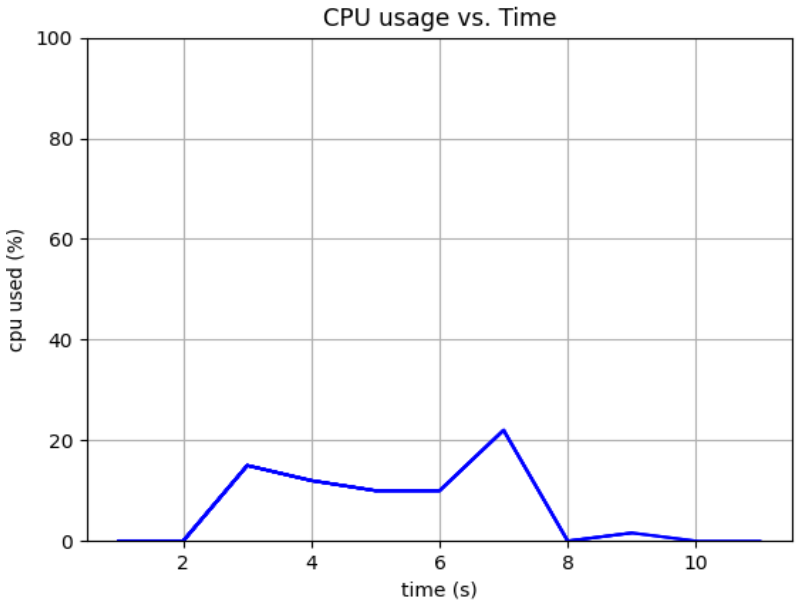


Figura 3.5. Uso de CPU cuando se notifica el código COVID.

La figura 3.6 corresponde a una de las gráficas de uso de CPU cuando el usuario anota síntomas de la enfermedad, a esta funcionalidad se le asignó un tiempo promedio de 13 segundos para que el usuario complete la acción.

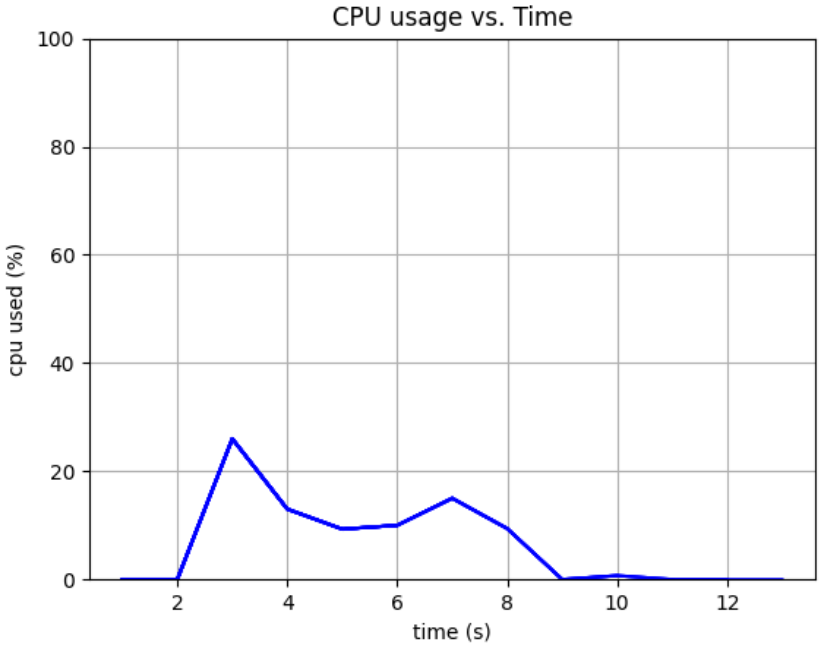


Figura 3.6. Uso de CPU cuando se anota síntomas

La figura 3.7 corresponde a una de las gráficas de uso de CPU cuando el usuario navega por cada item que se encuentra en la pantalla principal de la aplicación móvil ASI Ecuador, se asignó un tiempo promedio de 10 segundos para completar la acción.

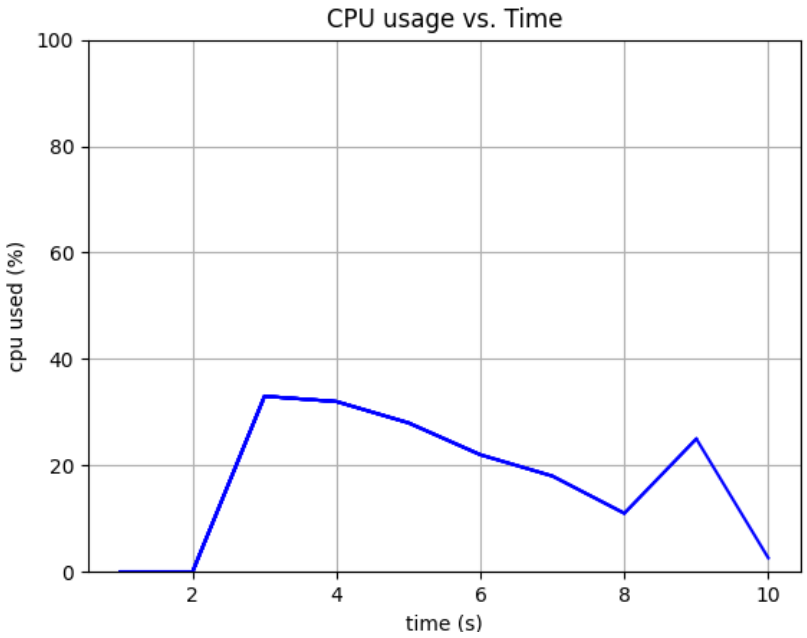


Figura 3.7. Uso de CPU cuando se navega por la aplicación móvil

Con respecto al uso CPU cuando la aplicación móvil esta en background, se mantenía en el valor de cero porciento durante todos los 10 segundos asignados.

Se puede notar que en cada figura vemos un comportamiento no constante, y esto da a entender que la aplicación movil ASI Ecuador cuando usa CPU muestra altos y bajos pero manteniedose hasta un límite míximo aproximadamente de 36%, además cuando la aplicación móvil esta en background no usa CPU y en las otras funcionalidades se ve variaciones.

3.1.3 Análisis del uso de batería

Con respecto al análisis del uso de batería se recogieron 5 veces los valores de la manera en cómo se detalló en la sección 2.2.3 usando el registro de historial de uso de batería que traen en los dispositivos móviles Android.

Los valores recogidos fueron por cada día es decir una semana laboral, luego estos valores fueron promediados para obtener un promedio general de batería usada por la aplicación móvil ASI Ecuador.

En la figura 3.8 se observa un historial gráfico de uso de batería cuando esta va descargándose a medida que pasan las horas y en la parte inferior detalla cuando tiempo ha estado funcionando la batería del dispositivo móvil y cuando ha ocupado Bluetooth en ese tiempo.

Figura 3.8. Historial del uso de batería

Al obtener los datos de uso de batería tanto de Bluetooth y la aplicación móvil ASI Ecuador se observó un alto consumo de batería por parte de Bluetooth en comparación con los demás demostrando que el dispositivo móvil puede descargarse rápidamente.

3.1.4 Análisis del uso de datos de red

Usando un filtro de direcciones IP se obtuvo la gráfica que proporciona Wireshark en la parte de “Estadísticas” y luego en “I/O Graph”.

Al filtrar se requiere colocar lo siguiente “ip.src==190.152.52.235” y “ip.dst==190.152.52.235” en la cual ip.src es la dirección de origen y por lo tanto representaría los datos que recibe la aplicación móvil desde el servidor mientras que ip.dst es la dirección de destino en donde la aplicación móvil envía datos a los servidores donde está el backend. Cabe mencionar que la dirección IP 190.152.52.235 es la dirección IP del servidor donde está alojado el backend de la aplicación ASI Ecuador.

En la figura 3.9 se muestra la gráfica de bytes enviados y recibidos en el transcurso del tiempo, en este caso en segundos. Donde la serie de tiempo de color verde claro (Receiving) representa el filtro “ip.src==190.152.52.235” y la de color café (Sending) representa el filtro “ip.dst==190.152.52.235”.

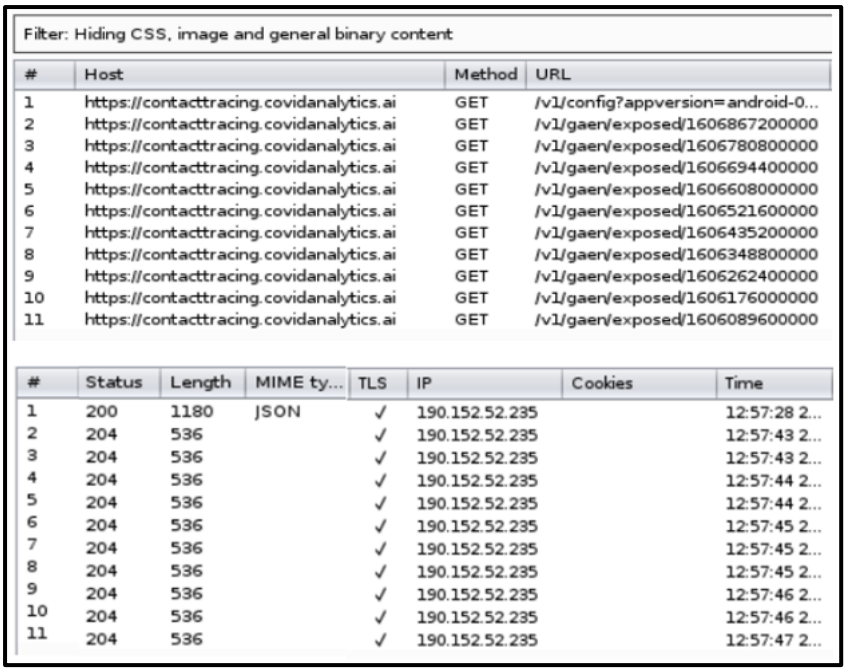
Figura 3.9. Gráfica de bytes vs. segundos (Network)

Además, la gráfica mostrada comprende desde el momento cuando se ingresa por primera vez a la aplicación móvil ASI Ecuador, proyectando un pico máximo de 5KB en Downlink o Receiving y esto pasa en el momento cuando se ingresan los datos de provincia, cantón y parroquia del usuario, luego en el segundo 140 aproximadamente se observa el momento cuando el usuario notifica el código COVID.

3.1.5 Análisis de la comunicación cliente-servidor

Una vez implementada la arquitectura para el análisis de seguridad, se debe constatar que el dispositivo móvil este conectado al servidor proxy de Burp Suite, cabe mencionar que esta prueba funciona con dispositivos móviles que tienen sistema operativo Android versión 6.0 o inferiores, en el caso de poseer un sistema operativo Android con versión superior a 6.0 pues se debería rootear al dispositivo móvil es decir, ser superusuario para obtener un control con privilegios en este.

En la figura 3.10 se muestra un registro del historial http generado entre la aplicación móvil ASI Ecuador (cliente) y el backend de dicha aplicación móvil (servidor) ya que el servidor proxy realiza el papel de intermediario entre la comunicación y a la vez escucha la comunicación entre estas dos partes. En el historial se observa información como el host, métodos http, urls usados por la aplicación móvil, códigos de estados htpp y la dirección IP destino es decir, la IP del servidor donde esta el backend de la aplicación móvil.



Filter: Hiding CSS, image and general binary content

#	Host	Method	URL
1	https://contacttracing.covidanalytics.ai	GET	/v1/config?appversion=android-0...
2	https://contacttracing.covidanalytics.ai	GET	/v1/gaen/exposed/1606867200000
3	https://contacttracing.covidanalytics.ai	GET	/v1/gaen/exposed/1606780800000
4	https://contacttracing.covidanalytics.ai	GET	/v1/gaen/exposed/1606694400000
5	https://contacttracing.covidanalytics.ai	GET	/v1/gaen/exposed/1606608000000
6	https://contacttracing.covidanalytics.ai	GET	/v1/gaen/exposed/1606521600000
7	https://contacttracing.covidanalytics.ai	GET	/v1/gaen/exposed/1606435200000
8	https://contacttracing.covidanalytics.ai	GET	/v1/gaen/exposed/1606348800000
9	https://contacttracing.covidanalytics.ai	GET	/v1/gaen/exposed/1606262400000
10	https://contacttracing.covidanalytics.ai	GET	/v1/gaen/exposed/1606176000000
11	https://contacttracing.covidanalytics.ai	GET	/v1/gaen/exposed/1606089600000

#	Status	Length	MIME ty...	TLS	IP	Cookies	Time
1	200	1180	JSON	✓	190.152.52.235		12:57:28 2...
2	204	536		✓	190.152.52.235		12:57:43 2...
3	204	536		✓	190.152.52.235		12:57:43 2...
4	204	536		✓	190.152.52.235		12:57:44 2...
5	204	536		✓	190.152.52.235		12:57:44 2...
6	204	536		✓	190.152.52.235		12:57:45 2...
7	204	536		✓	190.152.52.235		12:57:45 2...
8	204	536		✓	190.152.52.235		12:57:45 2...
9	204	536		✓	190.152.52.235		12:57:46 2...
10	204	536		✓	190.152.52.235		12:57:46 2...
11	204	536		✓	190.152.52.235		12:57:47 2...

Figura 3.10. Registro de las peticiones http entre la aplicación móvil y servidor

3.2 Resultados de las pruebas de caja negra

A continucacion se muestran los resultados de cada una de las pruebas de caja negra ejecutadas mediante gráficas de acuerdo al uso de memoria RAM, CPU, bateria, red y comunicación cliente-servidor.

3.2.1 Resultados del uso de memoria RAM

En la figura 3.11 se observa el promedio de uso de memoria RAM referente a las funcionalidades detalladas anteriormente, en donde la funcionalidad de navegar por los items usa 41.72 MB de promedio de uso de RAM mientras que las demás funcionalidades estan entre el rango de 30 a 40 MB de promedio de uso el cual representa un uso adecuado ya que el dispositivo móvil utilizado para las pruebas posee 2 GB de RAM, cabe mencionar que la aplicación móvil usa un promedio de 32.98 MB de memoria RAM cuando esta en background o ejecutándose en segundo plano.

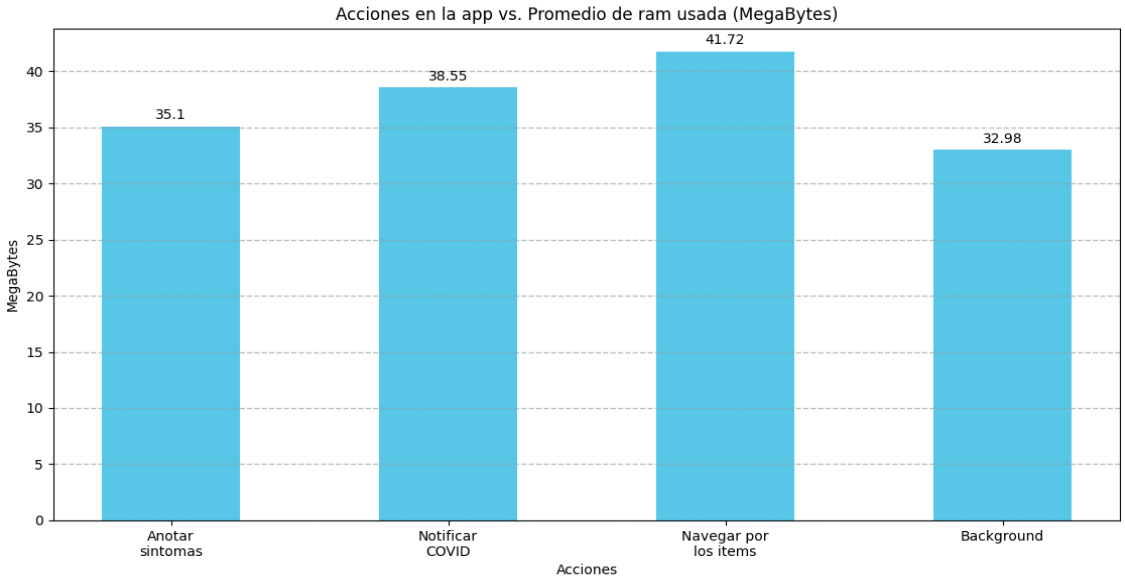


Figura 3.11. Promedio máximo de memoria RAM usada

3.2.2 Resultados del uso de CPU

En la figura 3.12 se observa el promedio de uso de CPU referente a las funcionalidades ya mencionadas, en donde la funcionalidad de navegar por los items usa un promedio de 17.16% de CPU el cual es mayor en comparación a las demás funcionalidades, cabe mencionar que la aplicación en segundo plano o background no usa CPU, demostrando que la aplicación móvil ASI Ecuador hace un uso adecuado de CPU.

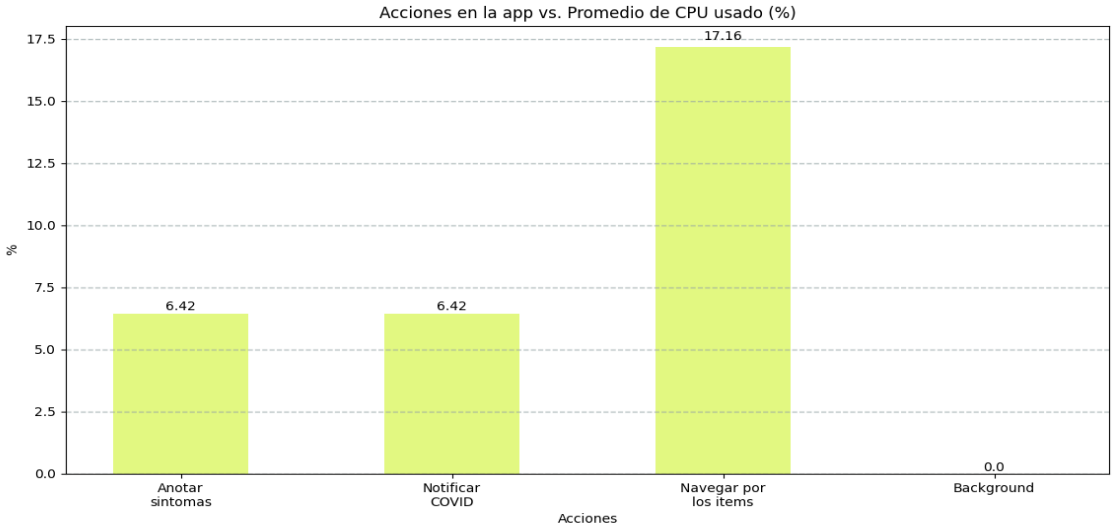


Figura 3.12. Promedio máximo de CPU usado

3.2.3 Resultados del uso de batería

Promediando el uso de bateria tanto de Bluetooth y de la aplicación movil ASI Ecuador, se consiguio el siguiente resultado sobre el uso aproximado de bateria contemplado en la figura 3.13

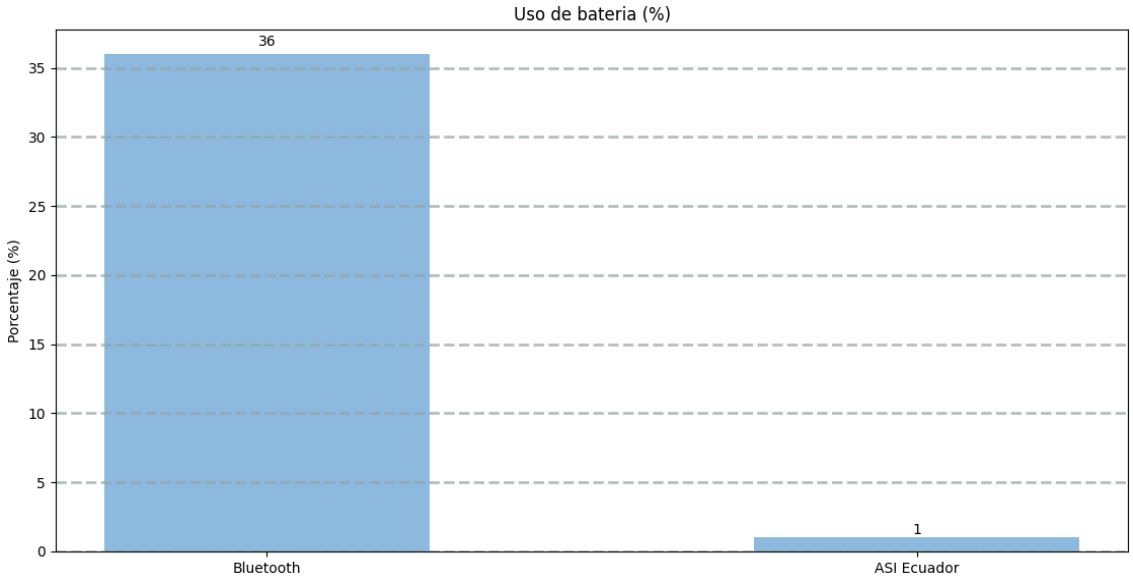


Figura 3.9. Promedio del uso de batería por Bluetooth y ASI Ecuador

El uso de batería por parte de Bluetooth es muy alto en comparación al uso de batería con respecto a la aplicación móvil ASI Ecuador y esto se debe a que las aplicaciones móviles que usan Bluetooth Low Energy (BLE) funcionan correctemante siempre y cuando este habilitado la geolocalización o GPS caso contrario no funcionarían como se esperaría, cabe mencionar que esto ocurre solamente en dispositivos móviles con sistema operativo Android.

Se instaló la aplicación ASI Ecuador en un dispositivo móvil iphone con sistema operativo iOS y no presentaba esta particularidad de BLE dada en Android ya que al habilitar solamente bluetooth, la aplicación ASI Ecuador no enviaba notificaciones de activación de geolocalización mientras que en dispositivos móviles Android la aplicación ASI Ecuador si envía estas notificaciones de activar la geolocalización.

3.2.4 Resultado del uso de datos de red

Este análisis de datos de red se realizó desde el momento en que la aplicación móvil se usa por primera vez, y se observó que recibe 5000 bytes aproximadamente al iniciar la aplicación ya que la aplicación móvil envía preguntas sobre el lugar donde reside el usuario y el usuario prosigue a contestar, luego en la próxima interacción de datos se observa un uso aproximado de 2500 bytes de datos de envío para validar el codigo COVID pero en este ejemplo el código COVID no fue autorizado por una entidad de salud aunque para este análisis sirve ya que permite observar el uso aproximado de datos de red.

Solamente en estos casos la aplicación móvil usa datos de red para realizar peticiones y recibir respuestas del servidor y en la gran parte del tiempo de uso de la aplicación móvil en cuestión no recibe ni pide datos al servidor lo cual es factible ya que ocupa datos de red solamente cuando lo necesita y en cantidades muy pequeñas.

3.2.5 Resultados importantes respecto a la comunicación cliente-servidor

Luego de analizar cada método http originado por la aplicación móvil al momento de interactuar con esta, se origino un método POST el cual correponde al momento de validar el código COVID y compartir los IDs registrados en los intercambios de llaves entre más dispositivos móviles.

En la figura 3.14 se observa la estructura de un json cuando el usuario decide compartir los IDs registrados en los intercambios de pares de llaves en los últimos 14 días.

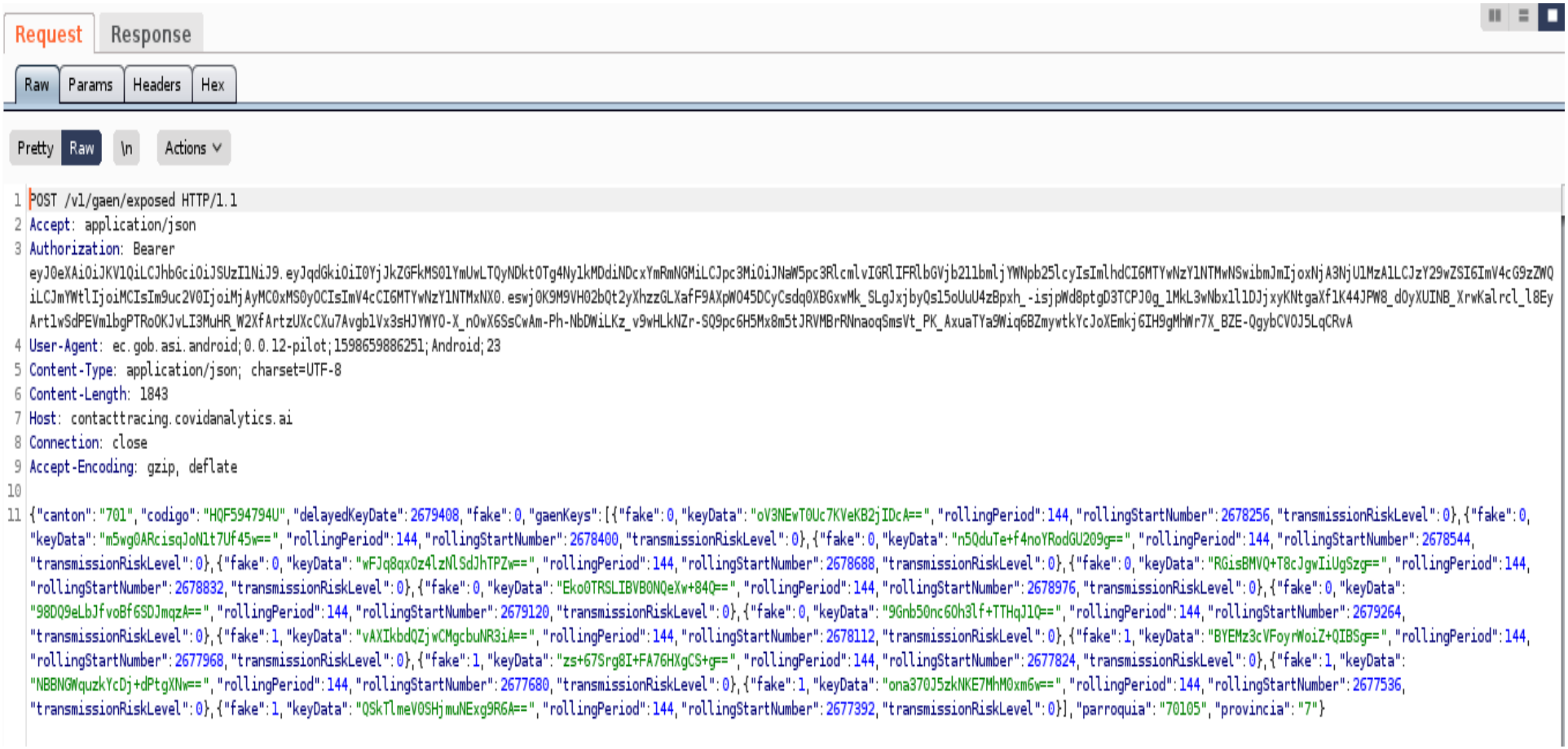


Figura 3.14. Información enviada al servidor cuando se notifica COVID positivo

Al comienzo del json se observa la clave denominada cantón cuyo valor es 701, mientras que en la parte final se observa las claves de parroquia y provincia cuyos valores corresponden a 70105 y 7 respectivamente, estos valores corresponden a datos de la ubicación del usuario. El número 7 que corresponde a provincia y representa a la organización territorial que comprende a las provincias de El Oro, Loja y Zamora Chinchipe mientras que el valor de 70105 se encuentra denotado por la parroquia El Cambio.

Los demás datos del json tienen que ver con los pares de llaves o IDs intercambiados los últimos 14 días.

Por lo tanto se deduce que estos datos de ubicación enviados en el momento en que el usuario dedice compartir servirían para cuantificar el número de personas infectadas por organización territorial y así llevar un control general a nivel nacional sobre los contagiados por COVID.

3.3 Pruebas de código fuente (caja blanca)

Teniendo en cuenta el código fuente almacenado en el siguiente enlace: https://minka.gob.ec/asi-ecuador , el primer paso consistió en usar el programa Android Studio para instalar la aplicación en los dispositivos móviles, como se observa en la figura 3.15.

Figura 3.15. Instalación de ASI Ecuador usando un dispositivo móvil físico

Sin embargo, el proceso no resultó exitoso debido a que surge un problema relacionado con una API (propiedad de Google) que solamente la pueden utilizar las aplicaciones oficiales registradas y desplegadas en la tienda de Play Store, las cuales pertenecen a los diferentes organismos de salud gubernamentales a nivel mundial. Dicho error se lo puede visualizar en la figura 3.16.

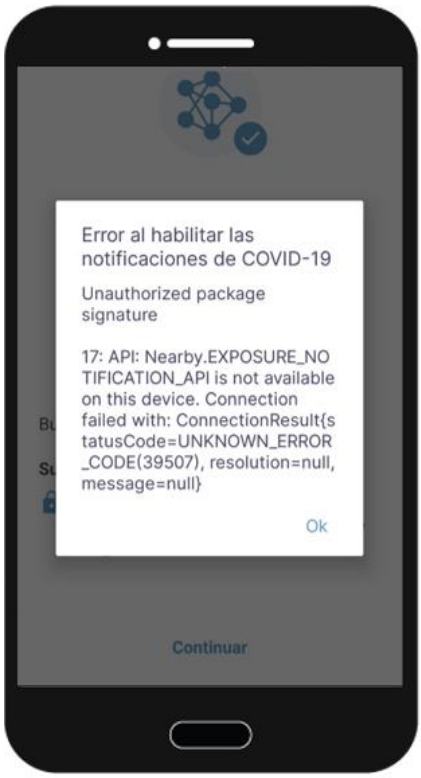


Figura 3.16. Error al habilitar las notificaciones COVID

La solución para esto consistió en utilizar una versión del protocolo principal para el funcionamiento de rastreo de conctactos, llamado DP3T, que no haga uso de la API de Google Play. Dicha versión está etiquetada bajo el nombre ***prestandard*** en el siguiente enlace: https://github.com/DP-3T/dp3t-sdk-android , como se aprecia en la figura 3.17.

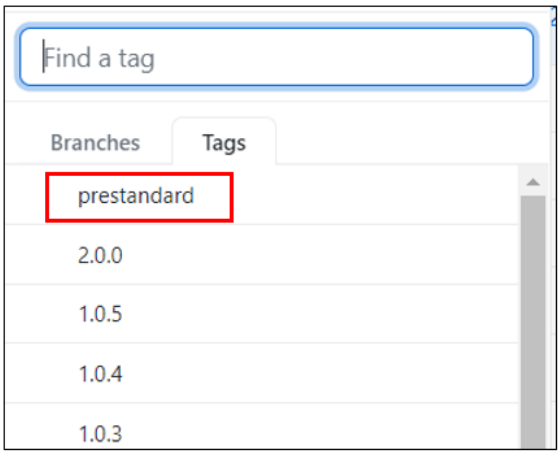


Figura 3.17. Versiones DP3T

Con lo anterior presente, entonces se pudo adaptar el código fuente tanto de la versión oficial como el de prestandard para crear una aplicación réplica que tenga la apariencia de ASI Ecuador y unicamente la funcionalidad del protocolo DP3T sin los servicios de Google, donde ahí sí se lograron realizar las pruebas de caja blanca y estudiar el intercambio de información mediante Bluetooth, además de la comunicación con un servidor local para las opciones de reportarnos como COVID positivo y notificar un posible contagio a los contactos con que se tuvo cercanía.

Figura 3.18. Aplicación replica (ASI Ecuador – DP3T)

La figura 3.19 muestra los componentes necesarios para tener un entorno simulado donde se probó la aplicación réplica. Se aprovecharon los recursos de comunicación que brindan los routers caseros y de esa forma, tener en una misma red a los dispositivos móviles y al servidor local que atienda los solicitudes de la aplicación.

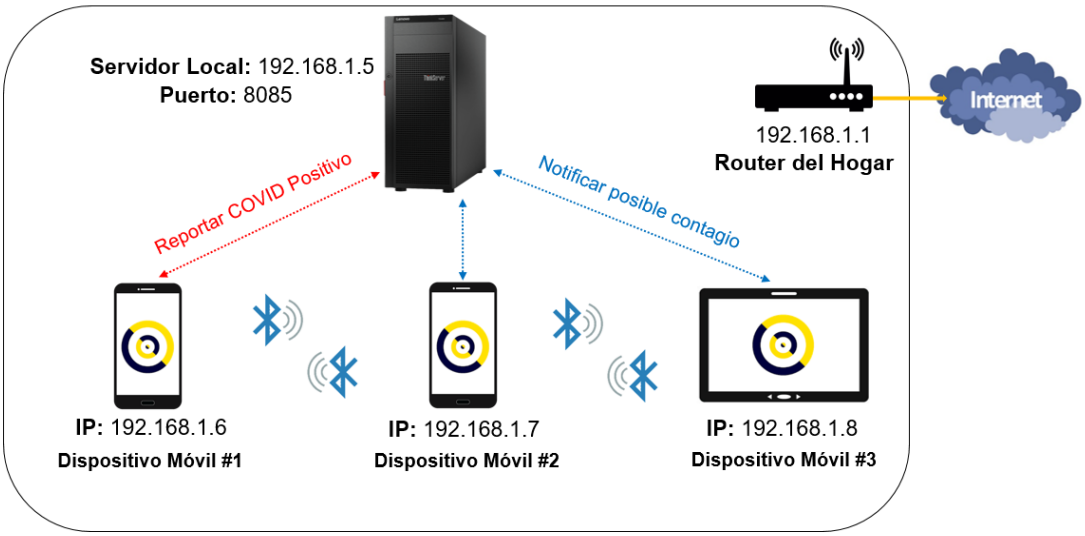


Figura 3.19. Arquitectura de red del entorno simulado

La primera prueba de caja blanca consistió en estudiar el proceso de generación e intercambio de llaves aleatorias para el rastreo de contactos. Luego, la segunda prueba se enfoca en analizar la comunicación con el servidor, especificamente para sus métodos http de POST y GET.

3.4 Resultados de las pruebas de caja blanca

Estos se obtuvieron analizando el código fuente de la aplicación en conjunto con la herramienta Logcat de Android Studio, que ayudó principalmente a imprimir mensajes por consola para identificar el orden de ejecución de los módulos y conocer cómo se ejecutaban las clases con sus respectivos métodos. En lo que respecta al backend, se utilizó el programa Spring Tool Suite 4, y extrayendo las partes más importantes del código fuente, se logró desplegar la API de manera local.

Estos procesos iterativos permitieron obtener los resultados que se describen en los siguientes apartados.

3.4.1 Generación de llaves aleatorias

La generación de llaves aleatorias ocurre en el módulo DP3T de la aplicación. El cual consta de dos clases para la interacción Bluetooth, BleServer (Ble, acrónimo de Bluetooth Low Energy) y BleClient (se explica luego), la primera de estas utiliza la clase CryptoModule, que a su vez usa una clase propia de Java llamada KeyGenerator, que está configurada para crear una llave secreta de 32 bytes usando el algoritmo de encriptación SHA-256, a continuación se muestra un posible ejemplo:

b13c222aee08a37740c907388d6ab44799266e9755ab0943ef79631da8b12990

Teniendo la llave anterior, luego usa un método generador de EphIDs (IDs efímeras), que encripta la llave ahora usando el algoritmo AES (Advanced Encryptation Standard) y finalmente utilizandola como “semilla”, crea un arreglo de 96 llaves aleatorias, pero ahora de 16 bytes (tamaño de tramas Bluetooth). El método le retorna una de esas 96 llaves al BleServer y este queda listo para escanear y buscar otros dispostivos que tengan instalada la aplicación y de esa forma intercambiar llaves.

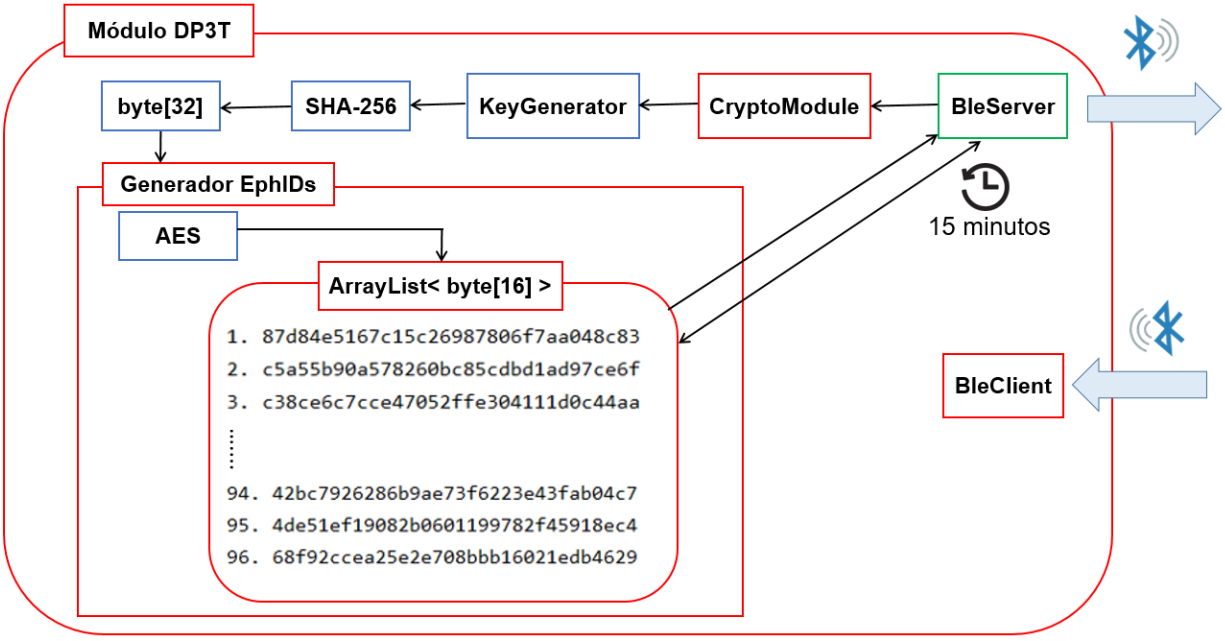


Figura 3.20. Flujo del proceso de creación de llaves aleatorias

Cabe mencionar que se crean 96 llaves debido a que cada 15 minutos el BleServer utiliza otra de estas, entonces para un día, 24 horas/día * 4 llaves/hora = 96 llaves/día. Mientras tanto al día siguiente crea otra llave secreta de 32 bytes y repite el proceso. La figura 3.20 esquematiza de mejor manera lo anteriormente descrito.

Luego, partiendo del escenario en que se tiene conectado un dispositvo (Huawei para este caso) a la PC, la figura 3.21 muestra la herramienta Logcat (embebida en Android Studio) que imprime la llave aleatoria de 32 bytes que utiliza en dicho día, además de la EphID de 16 bytes que hace de broadcast en la búsqueda de otros dispositivos para ese momento.

Figura 3.21. Uso de Logcat para mostrar las llaves aleatorias

3.4.2 Intercambio de llaves aleatorias

En el intercambio de las llaves aleatorias de 16 bytes, el BleServer cumple el rol de emisor y a su vez el BleClient cumple el papel de receptor, como se aprecia en la figura 3.22. Cuando esta última clase detecta una conexión con otro dispositivo, activa el método onDeviceFound, y guarda las llaves en un ArrayList que recibe objetos de tipo Handshake. La clase Handshake tiene otros atributos (a parte de la llave de 16 bytes) tales como intensidad de señal, potencia y dirección Bluetooth, pero para efectos practicos solo se muestran las llaves en la figura. Donde la razón de que aparezcan repetidas ocurre por el tiempo en que estan cercanos los dispositivos, ya que el BleServer hace un escaneo periódico de aproximadamente 1 minuto.

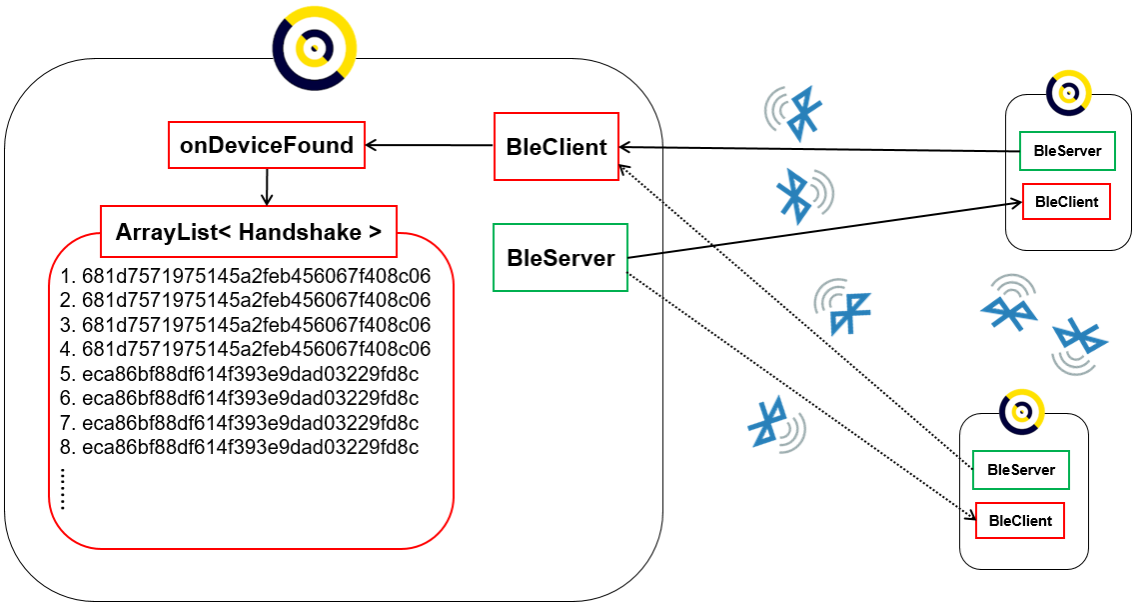


Figura 3.22. Registro de llaves en interacciones BLE

Cuando el BleClient detecta que ya no se agregan más Handshakes al ArrayList, utiliza una clase llamada Database para filtrar todos los handshakes con llaves repetidas y guardarlos en la base de datos SQLite de la aplicación bajo la denominación de “Contactos”, como se muestra en la figura 3.23.

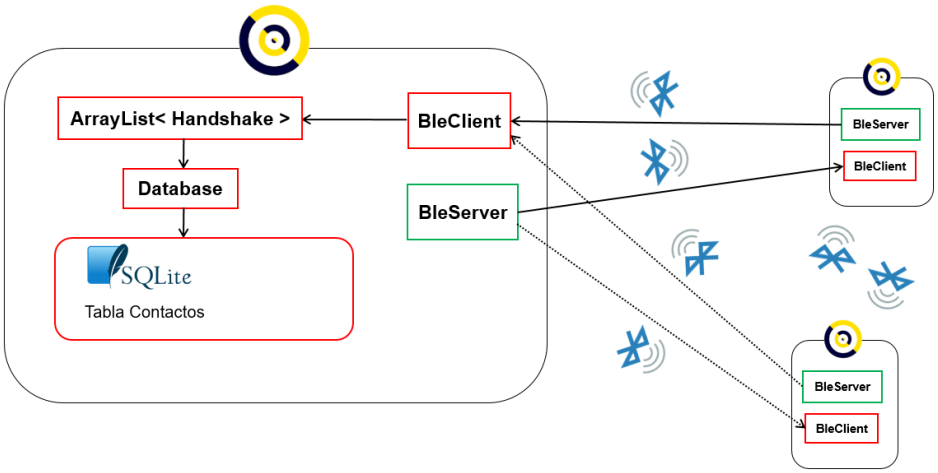


Figura 3.23. Registro de contactos en la base de datos interna

Continuando con el escenario de la sección anterior, se aprecian los mensajes que imprime Logcat (figura 3.24) cuando se acerca un segundo dispositivo que también tiene instalada la aplicación réplica. Ahí podemos notar que la llave actual que utiliza este primer celular es ***c532c297583262d0c810288ada857d7d***, mientras que la llave que recibe del segundo celular es ***8c9e37255af36d7f0e81f96ffe3c1733***.

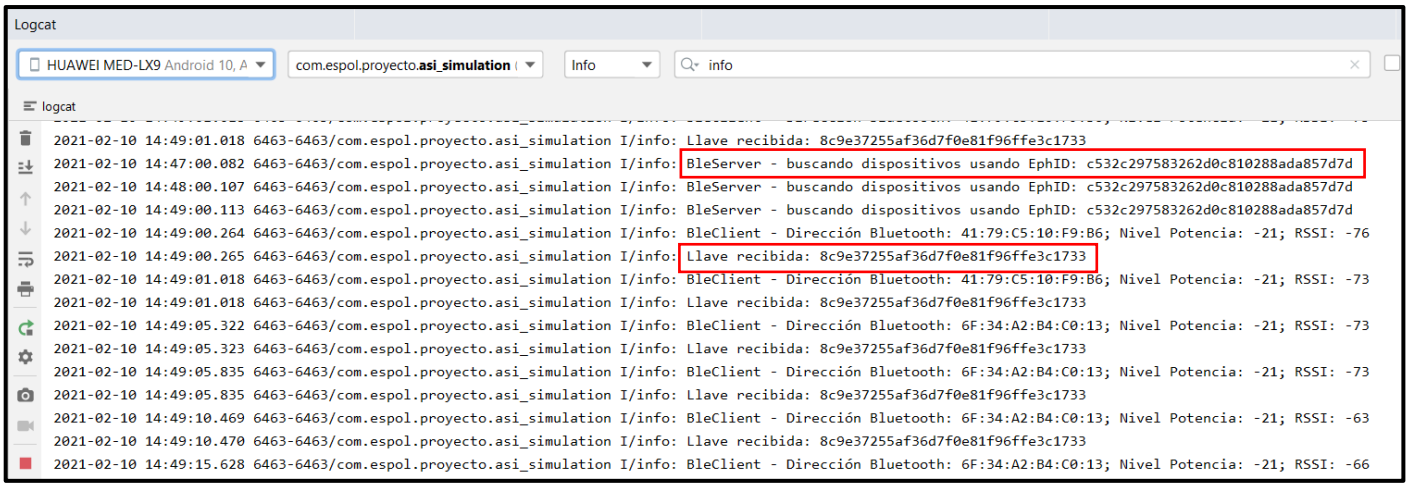


Figura 3.24. Mensajes Logcat en primer dispositivo

A continuación (figura 3.25), se observan los mensajes impresos por el segundo dispositivo (modelo ASUS), donde se cumple el rol a la inversa del apartado anterior, es decir, utiliza ***8c9e37255af36d7f0e81f96ffe3c1733*** y recibe del primer celular la llave ***c532c297583262d0c810288ada857d7d.*** Con lo cual se puede concluir que el proceso de intercambio mediante BLE resultó exitoso.

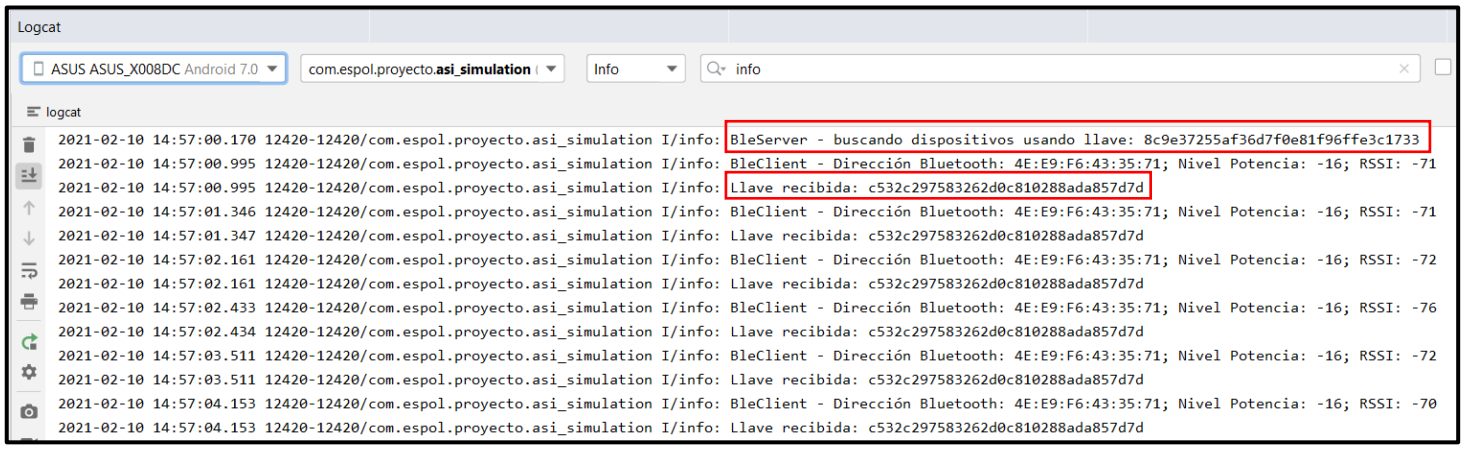


Figura 3.25. Mensajes Logcat en segundo dispositivo

3.4.3 Reportar COVID positivo

En la figura 3.26 se aprecia la interfaz principal de la aplicación móvil, para utilizar la opción de reportarse como usuario con COVID positivo en el servidor, se debe seleccionar el tercer recuadro con la etiqueta ***"¿Qué hacer si… el resultado del test es positivo?"***. Luego de seguir con el flujo de la aplicación, aparecera una ventana como en la figura 3.27, donde el aspecto a señalar ahí es el código COVID en el rectángulo, ***XLU 357U ZK3*** para este caso, el cual debe estar registrado previamente en la base de datos del servidor para que este proceso tenga éxito, esto se lo detallará en la siguiente sección, y para la explicación a continuación se supondrá que ya se realizó.

Figura 3.26. Interfaz principal de ASI Ecuador

Figura 3.27. Reportar COVID positivo

El módulo DP3T tiene una clase con el mismo nombre, esta a su vez utiliza la clase CryptoModule para obtener la llave de 32 bytes que está en uso, después esta llave es codificada bajo el sistema de numeración posicional Base64, acto seguido se crea un objeto de tipo ExposeeRequest, que define todos los datos del archivo JSON que se envían a través de la web. El objeto es retornado a DP3T y ahora llama a una clase con el nombre de BackendReportRepository, que contiene el método para interactuar con la API gracias a la clase Retrofit2.

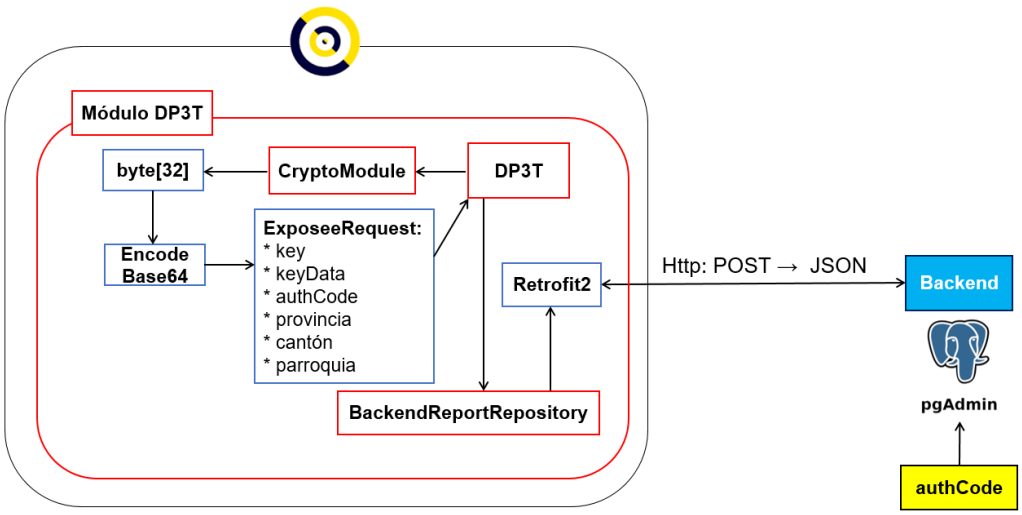


Figura 3.28. Esquema petición POST

La figura 3.29 muestra un mensaje de un interceptor http colocado en el código fuente para imprimir la comunicación cliente-servidor. Se puede destacar el método POST usado, la dirección ip y puerto del backend, así como la ruta de la API, el objeto JSON encerrado en el rectángulo rojo y la respuesta exitosa del servidor con el código 200.

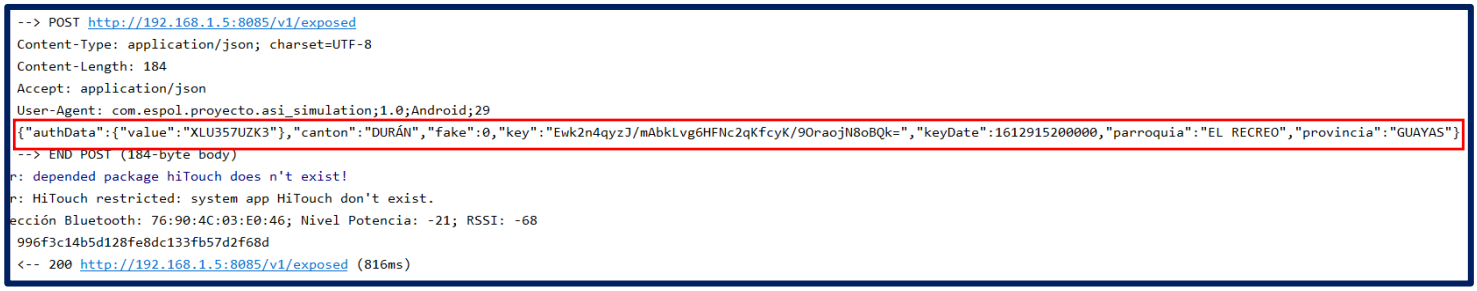
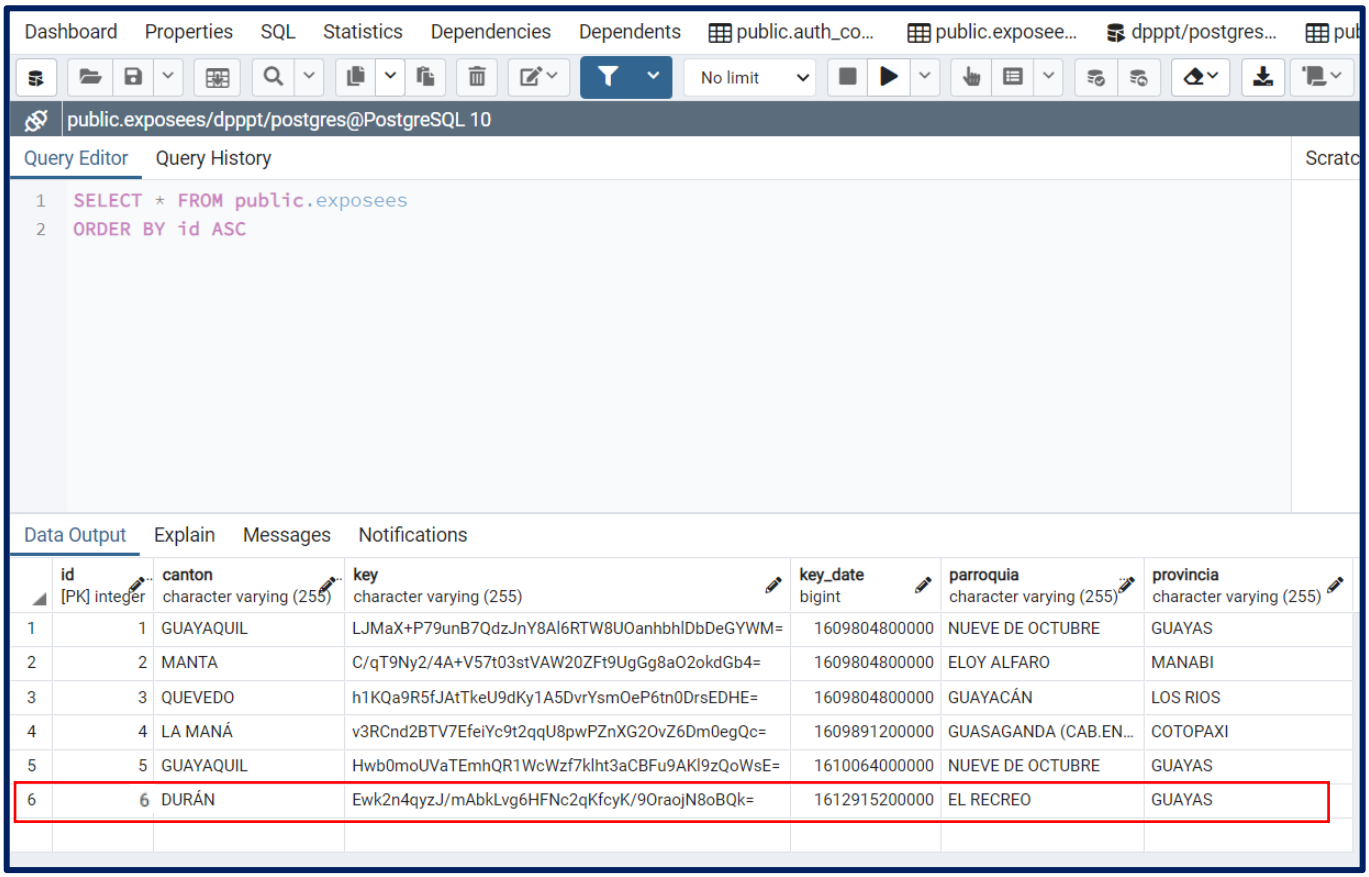


Figura 3.29. Mensaje POST del interceptor Http

Por otro lado, en la base de datos del servidor local se guarda el registro en la respectiva tabla como se muestra en la figura 3.30.



	id [PK] integer	canton character varying (255)	key character varying (255)	key_date bigint	parroquia character varying (255)	provincia character varying (255)
1	1	GUAYAQUIL	LJMaX+P79unB7QdzJnY8Al6RTW8UOanhbhlDbDeGYWM=	1609804800000	NUEVE DE OCTUBRE	GUAYAS
2	2	MANTA	C/qT9Ny2/4A+V57t03stVAW20ZFt9UgGg8aO2okdGb4=	1609804800000	ELOY ALFARO	MANABI
3	3	QUEVEDO	h1KQa9R5fJAtTkeU9dKy1A5DvrYsmOeP6tn0DrsEDHE=	1609804800000	GUAYACÁN	LOS RIOS
4	4	LA MANÁ	v3RCnd2BTV7EfeiYc9t2qqU8pwPZnXG2OvZ6Dm0egQc=	1609891200000	GUASAGANDA (CAB.EN...	COTOPAXI
5	5	GUAYAQUIL	Hwb0moUVaTEmhQR1WcWzf7klht3aCBFu9AKl9zQoWsE=	1610064000000	NUEVE DE OCTUBRE	GUAYAS
6	6	DURÁN	Ewk2n4qyzJ/mAbkLvg6HFNc2qKfcyK/9OraojN8oBQk=	1612915200000	EL RECREO	GUAYAS

Figura 3.30. Registro en la base de datos del backend

3.4.3.1 Validar código de autorización

Para efectuar este proceso se utiliza pgAdmin 4, que es una interfaz web para interactuar con bases de datos PostgreSQL. La figura 3.31 muestra cuando se ingresa el query en la tabla *auth_codes* del database *dpppt* para registrar el código de autorización del usuario de las secciones anteriores.

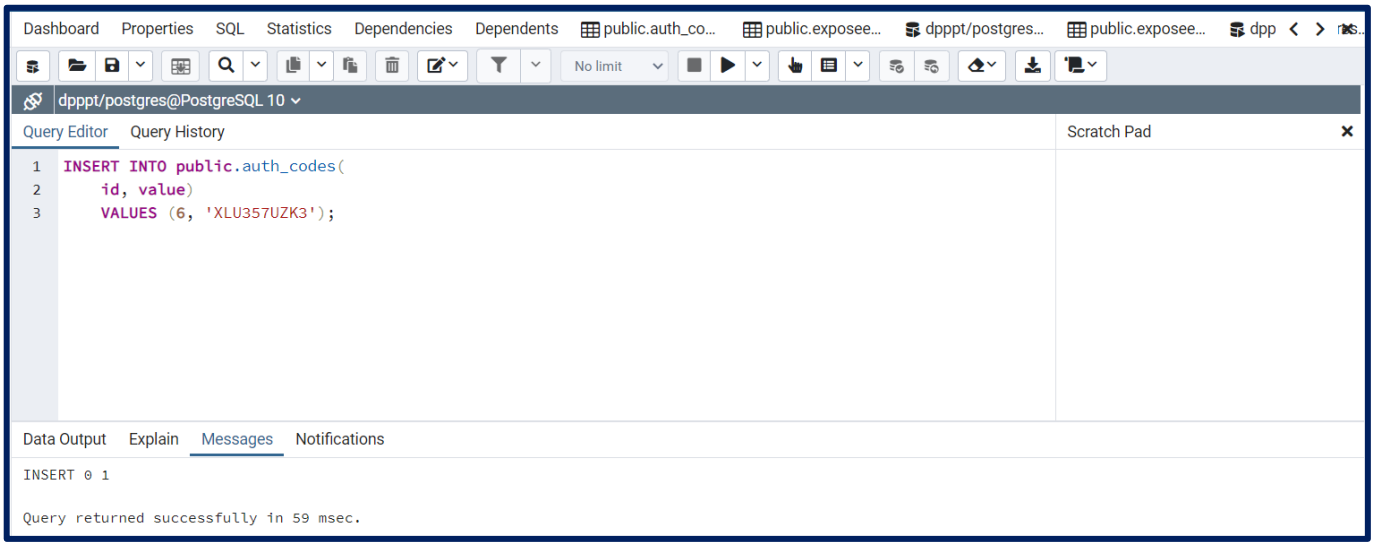


Figura 3.31. Ingreso del código COVID en la tabla auth_codes

Una vez que el proceso resulta exitoso, se consultan todos los valores de la tabla *auth_codes* para asegurarse que el código queda registrado, tal y como se aprecia en la figura 3.32.

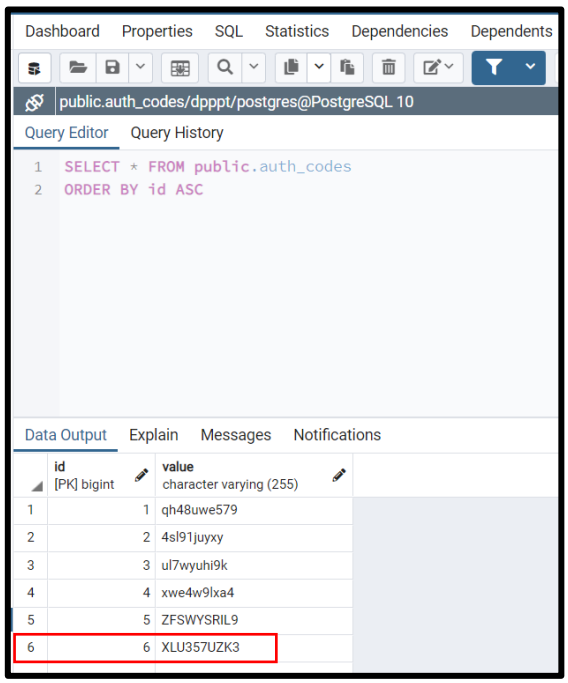


Figura 3.32. Consulta de los valores de la tabla auth_codes

3.4.4 Notificar posible contagio a los contactos

La aplicación móvil realiza este proceso de manera automática aproximadamente cada hora, para ello cuenta con una clase llamada SyncWorker en el módulo DP3T, que en conjunto con las clases BackendBucketRepository y Retrofit2 efectúan la solicitud GET para obtener un archivo en formato *protobuf* (protocol buffers) con todos los usuarios que se han reportado como COVID positivo. Vale mencionar que para obtener consultas inmediatas, se modificó el código para que también lo haga cada que el MainActivity de la aplicación se encuentre en estado onPause.

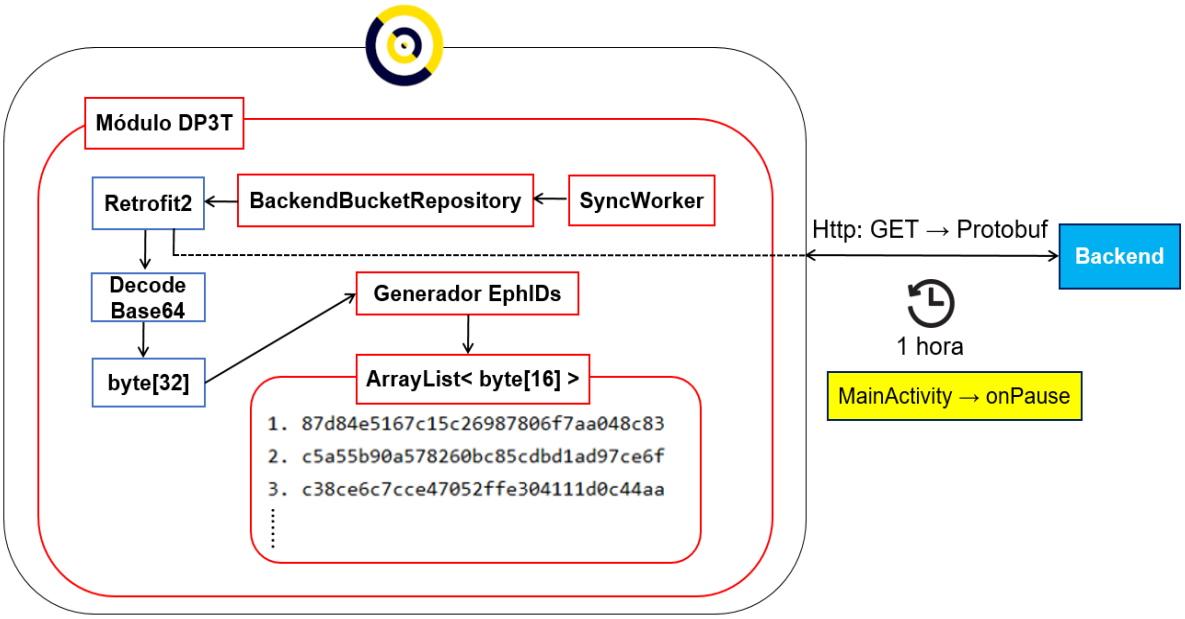


Figura 3.33. Esquema petición GET

Teniendo el archivo, por cada usuario, extrae la llave aleatoria y la decodifica bajo la notación Base64, obteniendo la llave de 32 bytes, con esto, repite el proceso que efectua el BleServer para crear la lista con las 96 llaves aleatorias de 16 bytes, como se observa en la figura 3.33.

Una vez creado el ArrayList, utiliza la clase Database para acceder a la tabla Contactos y comparar las respectivas llaves intercambiadas mediante Bluetooth con los dispositivos que tuvo cercanía. En caso de haber una coincidencia, la misma aplicación crea una notificación de posible contagio, como se visualiza en la figura 3.34.

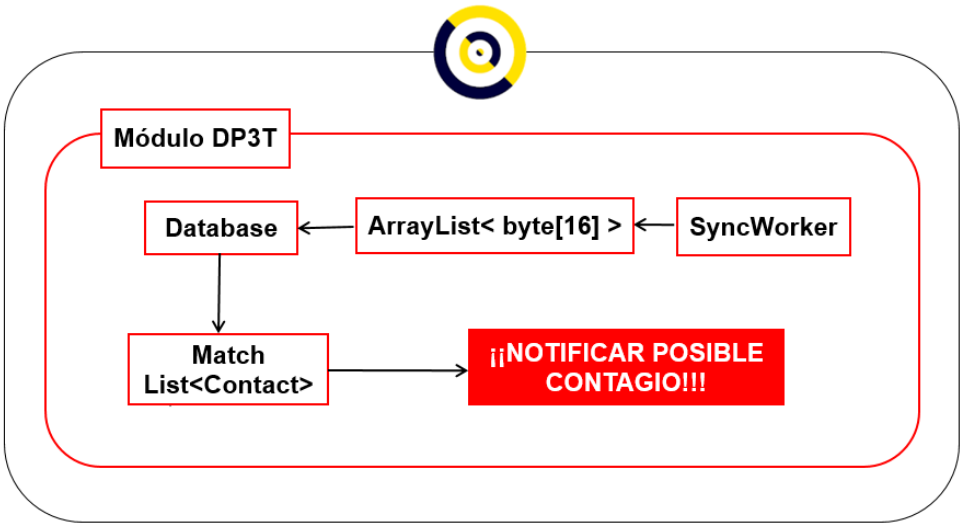


Figura 3.34. Esquema de creación de la notificación de posible contagio

Por otro lado, en las figuras 3.35 y 3.36 se aprecian capturas de pantalla del segundo dispositivo con la respectiva notificación y algunos consejos (luego de hacer presionar el botón Continuar) para prevenir la propagación del virus .

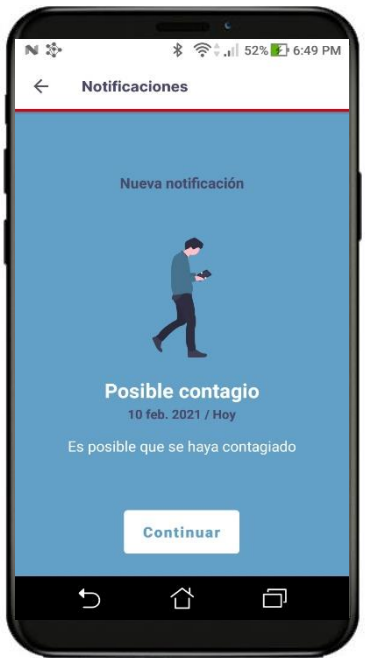


Figura 3.35. Notificación de posible contagio

Figura 3.36. Consejos para prevenir la propagación del virus

Para finalizar, en la figura 3.37 aparece otro mensaje de interceptor http en Logcat con la solicitud GET donde se destacan la ruta al servidor con el parámetro de la fecha en número Unix, el intercambio en formato Protobuf y la respuesta exitosa con código 200.

```
I/okhttp.OkHttpClient: --> GET http://192.168.1.5:8085/v1/exposed/1613080800000
I/okhttp.OkHttpClient: Accept: application/x-protobuf
I/okhttp.OkHttpClient: User-Agent: com.espol.proyecto.asi_simulation;1.0;Android;24
I/okhttp.OkHttpClient: --> END GET
I/System.out: [socket][8] connection /192.168.1.5:8085;LocalPort=-1(10000)
I/art: Do full code cache collection, code=252KB, data=229KB
I/art: After code cache collection, code=222KB, data=179KB
I/System.out: [socket][/192.168.1.9:51487] connected
I/okhttp.OkHttpClient: <-- 200 http://192.168.1.5:8085/v1/exposed/1613080800000 (1326ms)
I/okhttp.OkHttpClient: X-BATCH-RELEASE-TIME: 1613080800000
I/okhttp.OkHttpClient: X-Protobuf-Schema: resources/exposed.proto
I/okhttp.OkHttpClient: X-Protobuf-Message: com.espol.asi_simulation.Backend_ASI_Simulation.Model.proto.ProtoExposedList
I/okhttp.OkHttpClient: Content-Type: application/x-protobuf
I/okhttp.OkHttpClient: Transfer-Encoding: chunked
I/okhttp.OkHttpClient: Date: Thu, 11 Feb 2021 00:03:29 GMT
I/okhttp.OkHttpClient: Keep-Alive: timeout=60
I/okhttp.OkHttpClient: Connection: keep-alive
I/okhttp.OkHttpClient: <-- END HTTP (binary 265-byte body omitted)
```

Figura 3.37. Mensaje GET del interceptor Http

3.5 Presupuesto para la implementación de las técnicas de análisis usadas

La tabla 3.1 muestra el presupuesto total para la implementación de las técnicas de caja negra y blanca usadas en el análisis de la aplicación móvil para contact tracing ASI Ecuador.

Tabla 3.1. Presupuesto para implementación

DETALLE	VALOR
Computadora (12GB de RAM mínimo, procesador Intel Core i7)	$1200
Desarrollo de software (programador)	$200
Un dispositivo móvil	$185
Consumo mensual internet hogar (5MB velocidad)	$30
Instalación internet	$60
TOTAL	**$1675**

En la figura 3.38 se muestra una comparativa entre los costos de implementación del análisis mencionado con otras alternativas que hay en el mercado tecnológico

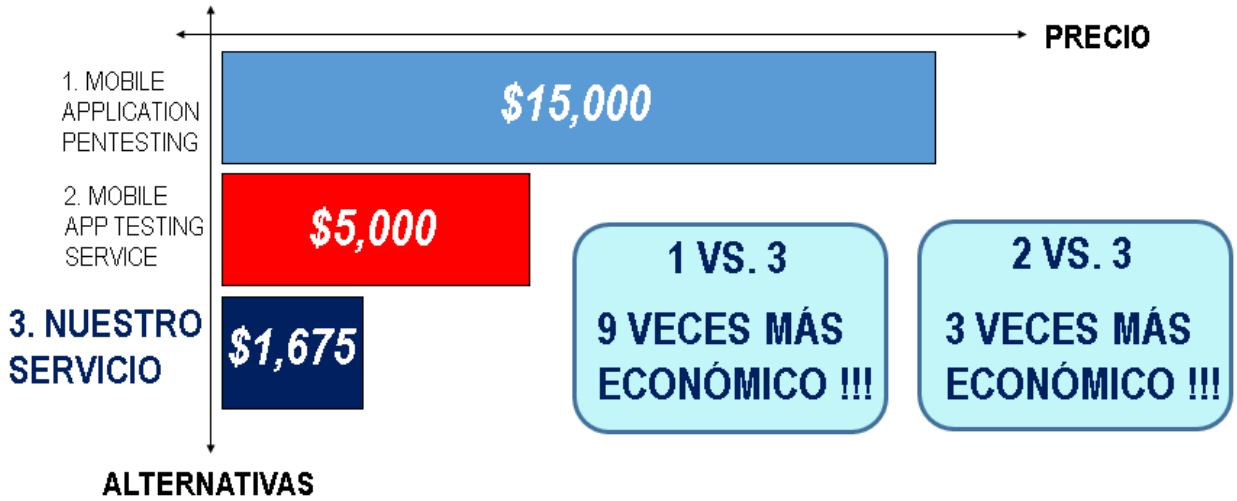


Figura 3.38. Comparativa de la solución planteada vs. otras del mercado

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- Se logró obtener indicadores de uso de CPU, memoria RAM, batería y red permitiendo conocer los valores máximos que tiende a ocupar la aplicación móvil ASI Ecuador en un dispositivo móvil.
- Hay un alto consumo de batería por parte de la interfaz bluetooth, aproximadamente 36% como promedio en los dispositivos móviles Android, debido al funcionamiento de la geolocalización para el funcionamiento de la tecnología BLE que permite el rastreo de contacto.
- En los dispositivos móviles con sistema operativo IOS no es necesario activar la geolocalización para que BLE funcione en comparación con los dispositivos móviles Android que si necesitan activar la geolocalización.
- El promedio máximo de memoria RAM usada es de 41.72MB mientras que el de CPU usado es de 17.16%, además el promedio de la tasa de datos usados tanto en uplink y downlink es de 2.5KB/s concluyendo que la aplicación móvil ASI Ecuador ocupa pocos recursos de hardware en los aspectos mencionados.
- Las técnicas usadas para el análisis a este tipo de aplicaciones móviles son más económicas en comparación a las que se encuentran en el mercado tecnológico como son el pentesting y testing de aplicaciones móviles.
- El entorno simulado permitió conocer que la aplicación contiene un componente que actúa como broadcaster (BleServer) anunciando su código encriptado y también cuenta con un receptor (BleClient) que permite establecer la conexión y realizar el proceso para registrar a los contactos.
- En las ocasiones que la aplicación interactúa con el servidor, no se comparte información personal de los dispositivos, lo que garantiza la seguridad y privacidad del usuario.

Recomendaciones

- Este tipo de análisis donde se interactúan con las interfaces de Bluetooth y red de un dispositivo móvil es necesario tenerlo de manera física ya que si se pretende virtualizar no se obtendrán resultados.
- Para este análisis solo se usaron dispositivos móviles Android, pero es necesario conseguir otros dispositivos con sistema operativo IOS ya que hay grandes diferencias entre estos dos sistemas en el funcionamiento de BLE.
- Si no se posee conocimiento de alguna librería, software, framework es necesario invertir tiempo en entenderlo para seguir en el trayecto del proyecto.
- El entorno simulado se lo puede extender desplegando el código del servidor local en alguna plataforma en la nube como AWS, Google Cloud o Azure, lo que permitiría realizar pruebas con usuarios en distintas localidades.
- Una segunda manera de extender el entorno simulado puede ser configurando un puerto especifico en el router del hogar para que el servidor local usé su dirección IP pública y tenga comunicación con el resto de internet.
- Como trabajo futuro, se podría desarrollar un aplicativo móvil o de escritorio que integre y automatice todas las pruebas unitarias desarrolladas a través de la metodología de caja negra.
- Para el proceso de análisis del código, se recomienda usar una computadora con memoria RAM de mínimo 8 GB, debido a que programas necesarios como Android Studio, consumen gran cantidad de recursos.

BIBLIOGRAFÍA

[1] Guo, J. & Li, J. (2020). COVID-19 Contact-tracing Apps: A Survey on the Global Deployment and Challenges. Department of Electrical and Electronic Engineering, Imperial College London. Londres, Inglaterra.

[2] Shubina, V. Ometov, A. Lohan, E. (2020). Technical Perspectives of Contact-Tracing Applications on Wearables for COVID-19 Control. Electrical Engineering Unit, Tampere University. Tampere, Finland.

[3] Bay, J. Kek, J. Tan, A. Sheng Hau, C. Yongquan, L. Tan, J. Anh Quy, T. (2020). BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders. Government Technology Agency. Singapore.

[4] Ahmed, N. Michelin, R. Xue, W. Ruj, S. Malaney, R. Kanhere, S. Seneviratne, A. Hu, W. Janicke, H. Jha, S. (2020). A Survey of COVID-19 Contact Tracing Apps. Cyber Security Cooperative Research Centre, University of New South Wales, CSIRO, Edith Cowan University. Australia.

[6] Dshs.texas.gov (2020, septiembre 10). Rastreo de contactos [Online]. Disponible en: https://www.dshs.texas.gov/coronavirus/tracing-sp.aspx

[7] Who.int (2017, mayo). Rastreo de los contactos en situaciones de brotes epidémicos [Online]. Disponible en: https://www.who.int/features/qa/contact-tracing/es/

[8] El Comercio (Ecuador, 2020, noviembre 11). Optimismo en la OMS ante primeros grandes avances en vacuna contra el covid-19 [Online]. Disponible en: https://www.elcomercio.com/actualidad/optimismo-oms-vacuna-covid19-pfizer.html

[9] Fernández, H. (2019). La importancia de las apps móviles en las empresas [Online]. Disponible en: https://economiatic.com/importancia-apps-moviles-empresas/

[10] Ochoa, R. Fajardo, D. Sánchez, M. Osorio, M. (2019, marzo). Implementación de Aplicaciones Informáticas en la Industria Agrícola del Aguacate. Instituto Tecnológico de Ciudad Guzmán / Tecnológico Nacional de México. México.

[11] Santamaria, G. Hernández, E. (2015). Aplicaciones Médicas Móviles: definiciones, beneficios y riesgos. Universidad del Norte de Barranquilla. Colombia.

[12] Howell O'Neill, P. Ryan-Mosley, T. Johnson, B. (2020, mayo 7). A flood of coronavirus apps are tracking us. Now it's time to keep track of them. MIT. Estados Unidos.

[13] Hatke, G. Montanari, M. Appadwedula, S. Wentz, M. Meklenburg, J. Ivers, L. Watson, J. Fiore, P. (2020, junio 28). Using Bluetooth Low Energy (BLE) Signal Strength Estimation to Facilitate Contact Tracing for COVID-19.
MIT Lincoln Laboratory. Estados Unidos

[14] Caja blanca vs Caja negra - Tester moderno. (2021). Retrieved 4 February 2021, from https://www.testermoderno.com/caja-blanca-vs-caja-negra/

[15] Functional Testing: A Complete Guide with Types and Example. (2021). Retrieved 4 February 2021, from https://www.softwaretestinghelp.com/guide-to-functional-testing/

[16] A Complete Non-Functional Testing Guide for Beginners. (2021). Retrieved 4 February 2021, from https://www.softwaretestinghelp.com/what-is-non-functional-testing/

[17] Overview — Matplotlib 3.3.3 documentation. (2021). Retrieved 4 February 2021, from https://matplotlib.org/3.3.3/contents.html

[18] (2021). Retrieved 4 February 2021, from https://developer.android.com/studio/debug/dev-options?hl=es-419

[19] pure-python-adb. (2021). Retrieved 4 February 2021, from https://pypi.org/project/pure-python-adb/

[20] (2021). Retrieved 4 February 2021, from https://developer.android.com/studio/command-line/adb?hl=es-419

[21] Ataque Man in The Middle. (2021). Retrieved 4 February 2021, from https://www.kaspersky.es/blog/que-es-un-ataque-man-in-the-middle/648/

[22] Introducción a Burp Suite. (2021). Retrieved 4 February 2021, from https://medium.com/@ArtsSEC/introduccion-a-burp-suite-4f3d14b32af