FasTest-App

Rodriguez Angenelo Santiago, Romano Pablo Martín, Grilli Matias Nahuel
DNI: 37109237, DNI: 40389631, DNI: 31915669
Dia de cursada: Miércoles noche, Numero de Grupo: 4

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina

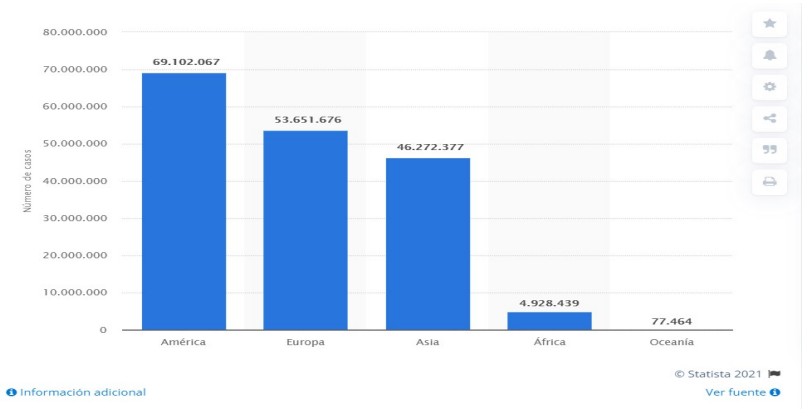
Resumen. FasTest-App es una aplicación móvil que corre sobre el sistema operativo Android (versión 7 en adelante), desarrollada en el marco de la pandemia del sars-cov-2. Su funcionalidad es realizar un chequeo rápido de síntomas compatibles con covid-19, y en caso de ser positivo, tener la posibilidad de dar aviso rápidamente tanto a emergencias como a contactos previamente definidos por el usuario.

Palabras claves: Android, Covid-19, Monitoreo Covid, Diagnóstico Rápido Covid, Aplicación.

1 Introducción

1.1 Desafíos en pandemia y contexto

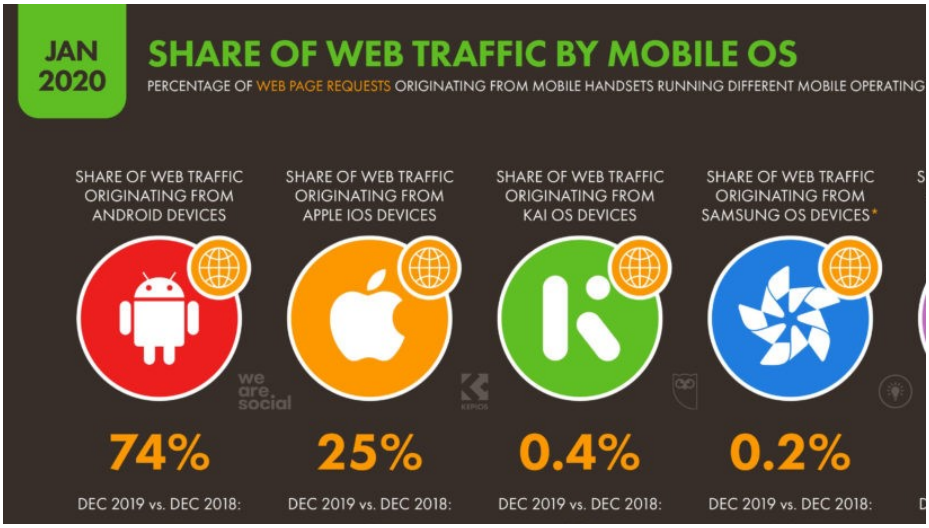
En un mundo ampliamente globalizado el sars-cov-2 irrumpió amenzando a la población mundial. La OMS determinó a dicha enfermedad como pandemia [1]. Los casos se multiplican por millones y a la fecha (junio de 2021) los números son contundentes [2]:



En estos tiempos donde las telecomunicaciones resultan imprescindibles y gran parte de las personas disponen de un teléfono móvil, el desafío en dar una respuesta a la nueva situación mundial es muy grande.

1.2 Soluciones inteligentes

La angustia de no saber si una persona está infectada con el virus del covid-19 y todo lo que ello conlleva para sus familiares y personas cercanas, es una preocupación frecuente en la sociedad.
FasTest-App nace para cubrir estas necesidades de manera muy sencilla.
Para desarrollarla decidimos utilizar Android 7 o superior, dada la supremacía de dicho sistema operativo en la actualidad [3].



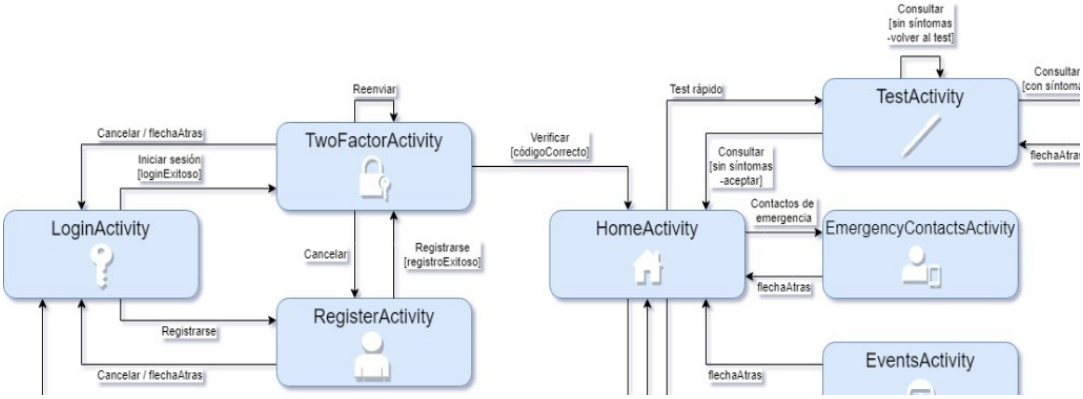
Volviendo a nuestra aplicación móvil, la misma puede ser descargada desde Playstore de Google.
El usuario debe registrarse con sus datos. A continuación de introducir su e-mail y contraseña. El login, al contar con un sistema de validación de doble factor, enviará un código al e-mail registrado para completar el ingreso a la app.
A continuación el usuario ingresa sus síntomas, y el sistema le da un rápido diagóstico. Si el mismo es covid-19 positivo, ofrece la posibilidad de comunicarse con emergencias y de dar aviso a través de SMS a sus contactos previamente definidos. Caso contrario se le informa que es negativo.
Asimismo se cuenta con un listado de eventos de cada sensor de la app para llevar un control de los mismos.

2 Desarrollo

2.1 Desarrollo

Dirección web del repositorio GitHub: https://github.com/MartinRomano-S/soa-tp2-grupo4

2.2 Diagrama funcional/navegación de las Activities.



2.3 Implementación de la ejecución concurrente del programa y mecanismo de sincronización

Los servicios que se conectan a internet fueron implementados con la interfaz AsyncTask, la cual nos permite gestionar el algoritmo para realizar el envío y recepción de información a internet, además de actualizar una barra de progreso cuando se termina de ejecutar para mantener al tanto al usuario visualmente que hay algo en ejecución, pero sin bloquear el proceso.
Además, tuvimos que implementar el patrón de diseño Singleton para controlar el acceso entre activities a las SharedPreferences. Esto permite tener solo una instancia del SharedPreferencesManager en toda la app. Además, agregando la palabra reservada synchronized en los métodos que modifican las preferences, nos aseguramos de que no haya dos métodos modificando las SharedPreferences a la vez.

2.4 Comunicación entre los componentes (Activites, Servicios,etc)

La comunicación entre activities se realizó mediante los “extras” de la clase Intent.
Primero declaramos el Intent, al cual le pasamos como argumentos la activity actual y la activity que se quiere iniciar de la siguiente manera:
Intent i = new Intent(RegisterActivity.this,TwoFactorActivity.class);

Luego le agregamos el parámetro extra:
i.putExtra("email", credentials.getEmail());

A continuación, en la activity que acabamos de iniciar, obtenemos el intent y el parámetro extra:
Intent i = getIntent();
email = i.getStringExtra("email");

En cuanto a los procesos en segundo plano implementados con AsyncTask utilizamos JSONObjects para enviar y recibir los datos ya que nos beneficiaba al a hora de trabajar con HttpRequests. Además, todos los objetos “model” que se enviaban por HttpRequest implementan una interfaz llamada “Jsonable” que obliga a implementar dos métodos para convertir el objeto a JSONObject y otro para llenar sus atributos desde un JSONObject, facilitando la gestión del envío de datos y recepción.
Para enviar las request a la API de la cátedra utilizamos el objeto URL junto con HttpURLConnection. Usamos BufferedOutputStream para enviar la información y, además, InputStream para recibir la respuesta del servidor tanto en casos de éxito como de error en la request. Toda esta lógica aplicada dentro de un AsyncTask para que se ejecute en segundo plano

2.5 Técnica que utilizó para la comunicación con el servidor (HttpURLConnection.

Se utilizó HttpURLConnection. Seteamos los headers correspondientes, utilizamos como base de URL http://so-unlam.net.ar/api/api/, y por Request le pasamos el nombre del servicio correspondiente. Luego abrimos e intentamos realizar la conexión, teniendo en cuenta que la misma puede ser fallida. Finalmente leemos la respuesta, la transformamos a un objeto JSON para trabajarla, y cerramos la conexión.

2.6 Implementación de la persistencia de los datos en la aplicación.

La persistencia de datos se realizó mediante SharedPreferences implementando el patrón de diseño Singleton para el acceso a las mismas. Cuando se cierra la aplicación y se vuelve a iniciar sesión los datos que estaban guardados se borran para permitir el ingreso de nueva información al usuario que inició sesión en ese momento.

2.7 Manual de Usuario de la Aplicación Android.

Adjuntado en el trabajo.

3 Conclusiones

- Recaudos utilizados para que la aplicación sea tolerante a fallos:

  Nos hemos percatado que antes de enviar cualquier tipo de request a internet, validar que haya una conexión Wi-Fi, Ethernet o Red Móvil activa.
  Asimismo a la hora de enviar SMS o realizar llamados, validar los permisos correspondientes de la app.
  Hicimos procesos que se conecten a internet en tareas de 2do plano con AsyncTask.
  Se tuvo en cuenta la sincronización a la hora de guardar los datos en SharedPreferences con Singleton.
  Tomamos el recaudo de desuscribirse de los listeners de los sensores en el método onPause de la Activity y volver a suscribirse en onResume.
  Al momento de enviar un request, se tuvo en cuenta validar el vencimiento del token de la API y pedir uno nuevo si está vencido antes de enviar dicho request para evitar el fallo en el servidor.
  Validación de longitudes mínimas y formatos válidos de los datos ingresados por el usuario.

- Problemas encontrados y resueltos:

  Un problema que tuvimos en el desarrollo de la aplicación fue el desarrollo de los diferentes layouts debido a la implementación de las constraints. Nos fue complejo entender el concepto de constraints y el porqué era necesario tenerlos en cuenta. Una vez comprendido esto no presentó mayores inconvenientes [4].

  Al momento de acceder a las nos dimos cuenta de que estábamos usando un método incorrecto que asociaba a la Preference con la Activity, con el inconveniente de que no se podía leer este dato desde otras activities. Para resolver este problema, aplicamos el método correcto [5] y una estrategia de sincronización con Singleton para que haya solo una instancia del objeto que

permita acceder a las SharedPrefrences y además agregar synchronized a los métodos que realizaban grabación sobre éstas.

También tuvimos inconvenientes con las clases deprecadas. Un ejemplo de esto ha sido NetworkInfo para validar la conexión a internet del dispositivo. Tuvimos que realizar una pequeña investigación para poder utilizar las clases recomendadas. Para el caso puntual de NetworkInfo, la solución fue usar en su lugar NetworkCapabilities [6].

En las AsyncTask al realizar actualizaciones visuales de la app, o incluso al solo pasarle el contexto, el propio IDE nos señalaba que esto era una mala práctica y podía desencadenar un error. La solución a esto fue implementar una interfaz <Asyncronable> que obligue a implementar métodos para actualizar la UI y así poder llamarlos desde AsyncTask sin los errores antes señalados.

Finalmente cuando implementamos los sensores nos anoticiamos de que continuaban funcionando, aún abandonando la Activity que realizaba las acciones al recibir algún estímulo. Esto se debía a que los sensores no estaban inicializados de manera correcta y que además nunca se detenía la escucha de los mismos.
La manera de resolverlo fue hacer que la suscripción se haga en el método onResume ya que se ejecuta siempre que se accede a la Activity en cuestión. Mientras tanto hicimos que en el método onPause, se desuscriba al cerrar la Activity.

- Lecciones aprendidas:

  El TP nos deja varias lecciones aprendidas. Como principal la importancia de trabajar en equipo sobre un mismo código, aprendiendo a dividirnos las tareas y utilizar git para trabajar concurrentemente y evitar conflictos. Ésta experiencia fue enriquecedora ya que no todos los integrantes del grupo contábamos con experiencia en esa metodología.
  Por otra parte, aprendimos conceptos completamente nuevos en desarrollo móvil, tales como: Manejo de sensores, Broadcast Receivers, Intents, por mencionar tan sólo algunos.
  Otros temas importantes aprendidos fueron:
  - Ordenamiento de código, para qué y cuándo implementar interfaces.
  - Patrón Singleton para limitar las instancias de objetos dentro de la aplicación.
  - Como escuchar diferentes tipos de sensores y saber cuándo desactivar la escucha de éstos.
  - No hacer actualizaciones de UI en métodos en segundo plano.
  - No enviar requests sin tener conexión a internet ya que demora más tiempo que validar la conexión.

4 Referencias

Bibliografía

[1] «Paho.org,» 2020. [En línea]. Available: https://www.paho.org/es/noticias/11-3-2020-oms-caracteriza-covid-19-como-pandemia.

[2] «statista.com,» 2021. [En línea]. Available: https://es.statista.com/estadisticas/1107712/covid19-casos-confirmados-a-nivel-mundial-por-region/.

[3] «Yiminshum,» 2020. [En línea]. Available: https://yiminshum.com/mobile-movil-app-2020/https://yiminshum.com/mobile-movil-app-2020/.

[4] android.cn-mirrors.com, «http://android.cn-mirrors.com,» 2020. [En línea]. Available: http://android.cn-mirrors.com/reference/android/net/NetworkCapabilities.html.

[5] A. Developers, «https://developer.android.com/training/constraint-layout/,» 2021. [En línea]. Available: https://developer.android.com/training/constraint-layout/.

[6] developer.android.com, «developer.android.com,» 2020. [En línea]. Available: https://developer.android.com/training/data-storage/shared-preferences?hl=es-419.