COVID-PersonLimiter

Acosta Bernascone, Ramiro[1]; Funes, Norman Federico[1]; Giasone, Santiago Claudio[1]
41.129.076, 41.204.592, 41.894.276
Comisión: Miércoles, Grupo: 17

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

Resumen: El virus COVID-19 ha causado severos problemas sanitarios, economicos y sociales en todos los paises del mundo. Afortunadamente, hoy en dia podemos decir que ha sido casi totalmente controlado. Sin embargo, los cuidados deben seguir existiendo. Por esta razon, la Organización de las Naciones Unidas, basandose en los ultimos estudios realizados por la Organización Mundial de la Salud, y con el concenso unanime de todos los paises que la integran, ha determinado que la erradicacion total y completa del virus COVID-19 será efectiva siempre y cuando se cumplan con los ultimos protocolos establecidos. Se le ha encomendado a los alumnos del grupo 17 de la materia Sistemas Operativos Avanzados de la Universidad Nacional de La Matanza que desarrollen una aplicación que cumpla irrestrictamente con dichos protocolos referidos a acceso a eventos, atracciones, espacios publicos, puntos de interes y cualquier otro sitio que implique una potencial aglomeracion de personas.

Palabras claves: Covid, Post Pandemia, Aplicación, Android, Sensores, Eventos, Actividades, MVP.

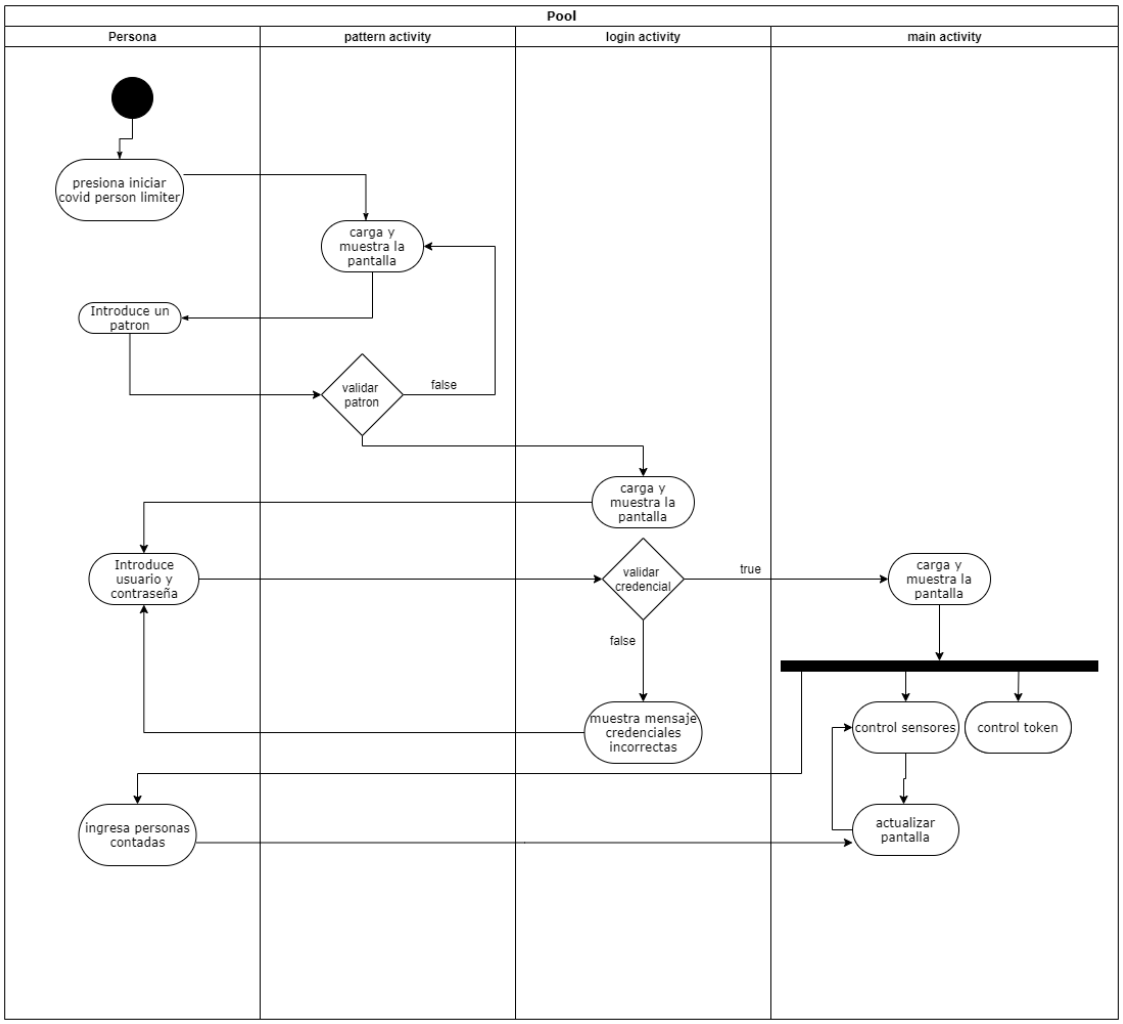
1 Introducción

- COVID-PersonLimiter es una aplicación para celulares que cuenten con Sistema Operativo Android, desarrollada en lenguaje Java, la cual permite a organizadores de eventos, dueños de restaurantes, comercios, mercados, museos, lugares de ocio, y cualquier otro espacio y/o sitio que pueda implicar un potencial aglomeramiento de personas, tener un control efectivo sobre la cantidad de ingresos de personas al lugar en cuestion. Debido a las ultimas restricciones impuestas a nivel mundial por las Organización de las Naciones Unidas, cada Espacio Potencial de Aglomeracion (desde ahora llamados EPA) mencionados anteriormente, deberá ser monitoreado ya sea por los mismos dueños o empleados o por una autoridad publica. De este modo, un EPA no podrá superar una determinada cantidad de ingresos de personas en forma simultanea, para evitar la aglomeracion de las mismas y evitar posibles focos de contagio del virus.
  Este limite de ingresos, a su vez, variará en funcion de la temperatura ambiente que exista en el EPA en dicho momento. Los ultimos estudios realizados por la Organización Mundial de la Salud han arrojado como resultado que en lugares donde la temperatura ambiente es baja, los contagios del virus entre personas aumentan, y a medida que la temperatura asciende, se disminuye la proporcion de contagios. Las ultimas resoluciones adoptadas por todos los paises fueron las siguientes:
  Si la temperatura ambiente es menor a 5°C (41°F) – Implica un aforo del 30% del EPA.
  Si la temperatura ambiente es mayor o igual a 5°C (41°F) y menor o igual a 15°C (59°F) – Implica un aforo del 50% del EPA.
  Si la temperatura ambiente es mayor a 15°C (59°F) – Se puede utilizar la capacidad maxima del EPA.
  Por lo tanto, COVID-PersonLimiter es una aplicación que determinará cuantas personas pueden ingresar a un determinado EPA en funcion de la capacidad maxima de dicho lugar (prefijada) y de la temperatura ambiente que haya en ese momento.

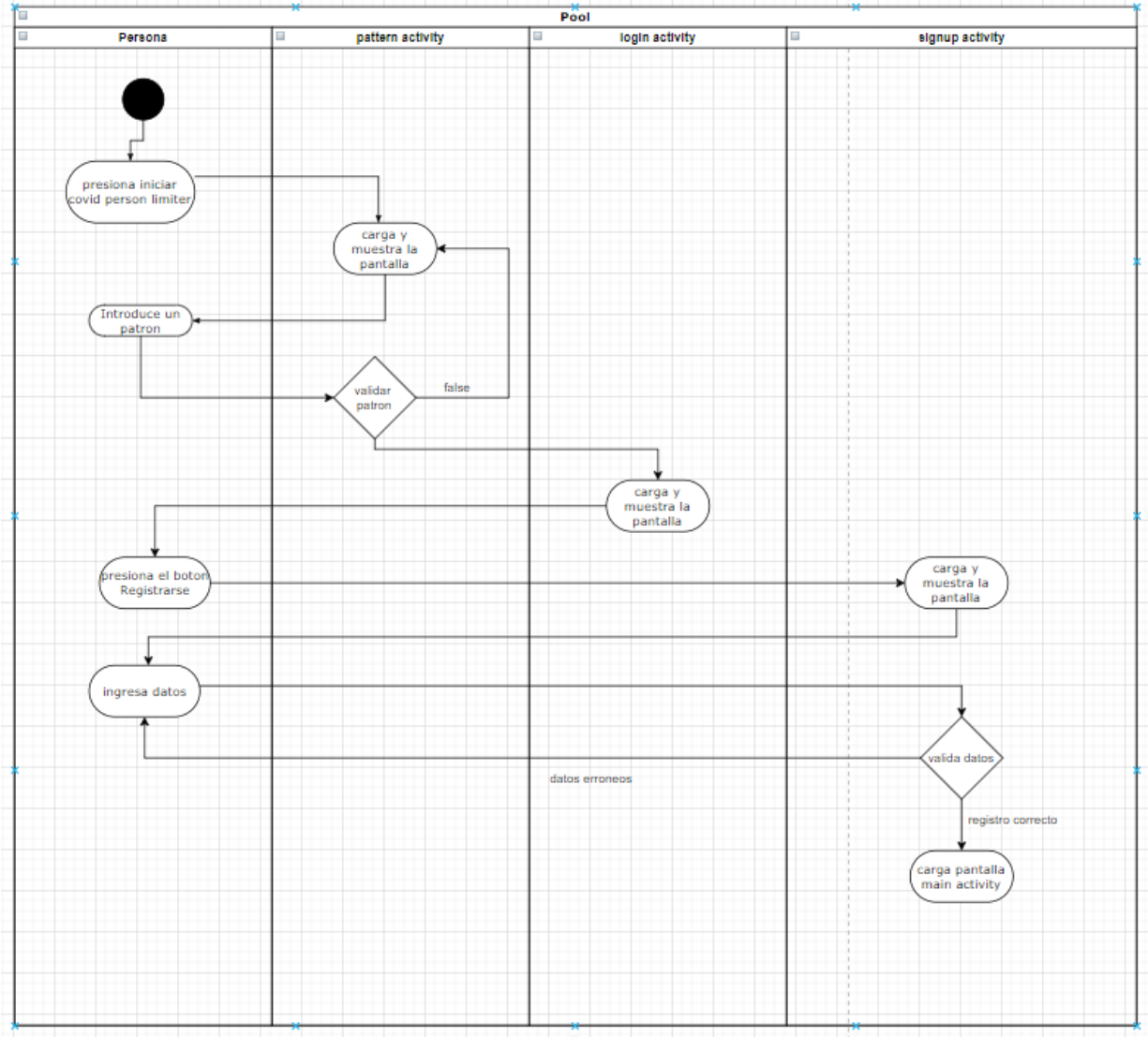
2 Desarrollo

1. Direccion del repositorio en GitHub: https://github.com/santiagogiasone/COVID-PersonLimiter.git

2. Diagramas de navegacion/funcional de las activities:
   Flujo Normal



Registro de usuario



3. Descripcion de la ejeucion concurrente del programa. ¿Se utilizaron mecanismos de sincronizacion?
Se utilizaron mecanismos de sincronizacion para la aplicación. Mediante un Thread, se guardan los datos de la cantidad de inicios de sesion correctos y fallidos que hubo en la aplicación con la librería SharedPreferences, implementada en la clase SharedPreferencesThread. El guardado se realiza de forma concurrente a cuando se inicia la MainActivity. Tambien se crearon Service para la implementacion de un BroadCast Receiver, a traves de la clase BatteryInfoService, la cual cuando inicia la aplicación muestra por pantalla el porcentaje de bateria que tiene el celular.

4. Descripcion de la comunicación entre componentes (Activities, Servicios)
La comunicación entre componentes sigue la estructura del Patron MVP (Model-Views-Presenters). Las views ofician de las pantallas visuales con las que interactua el usuario. En ella se muestran los datos, reciben los input del usuario y se muestran las salidas de la aplicación. Los models son los encargados de los datos de la aplicación, conexiones con SharedPreferences, clases que tienen la configuracion de los mensajes hacia el servidor, basicamente el modelado de los datos que posee o interactua la aplicación. Por su parte, los Presenters son el intermediario entre las anteriores mencionadas. Son los que se encargan de pasar la informacion de un lado hacia otro. Tambien tienen en si toda la logica del programa y las operaciones que hace. Se comunican tanto con los Models como con las Views. Por su parte, las Views y los Models no pueden comunicarse entre si (debe haber un Presenter en el medio).

5. ¿Qué técnica utilizó para la comunicación con el servidor (HttpConnection, Retrofit, etc) ?. Detalle como lo implementó desde la generación de la solicitud al servidor hasta la recepción de la respuesta.

Para conectarse con el servidor se utilizo la libreria de Retrofit, con la cual nosotros creamos nuestro modelo de conexion con el servidor para que se mantenga consistencia entre todos los request de la aplicacion. Para los request, retrofit requiere que se cree una clase implementadora donde nosotros le aclaramos que parametros queremos enviar y que tipo de solicitud es (GET,POST,PUT,DELETE), luego de esto nosotros le aclaramos a la instancia que respuesta asincronica esperamos mediante otra clase en la cual validamos que parametros recibimos del servidor. Para el actualizar el token de acceso nosotros implementamos un metodo en el cual el Modelo del Usuario valida si el token esta por expirar en los proximos 30 segundos y en base a eso genera un nuevo token si es necesario y luego mediante un callback ejecutamos un fragmento de codigo que el objeto padre que llamo a este metodo envio.

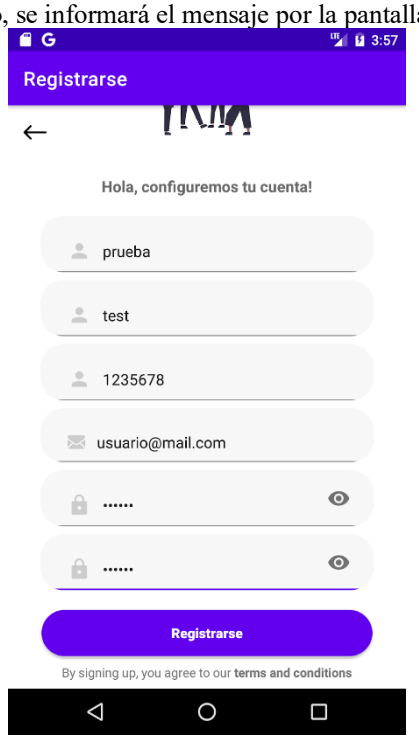
6. Describir cómo se realizó la persistencia de los datos en la aplicación. sobre este punto, ¿qué pasa cuando se cierra la aplicación?
Se utilizo la librería SharedPreferences. A su vez, se delego la ejecucion del guardado de los datos a un hilo. La clase SharedPreferencesThread es la encargada de esta logica. Implementa el metodo run() propio de los hilos para guardar la informacion necesaria. Cuando se cierra la aplicación, los datos quedan persistentes.

7. Escribir un manual de usuario, en donde se describa como se utiliza la aplicación.

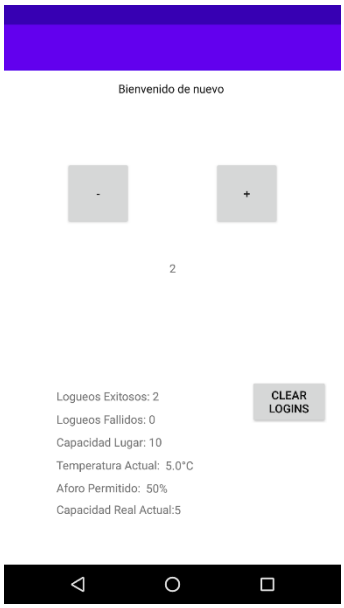
Cuando inicia la aplicación, se muestra la primer Activity de autenticación. Para poder autenticarse, se debe dibujar el patrón como se muestra la siguiente captura. En caso de ingresar un patron incorrecto, el patron ingresado se pintara de color rojo para indicar el error.

Luego se mostrará la segunda Activity de Login. La misma tiene dos campos de texto y dos botones. En caso de que el usuario ya se encuentre registrado, se deberá ingresar su mail y su contraseña y se deberá presionar el botón ingresar, la cual la llevará a la MainActivity de la aplicación. En caso de que el usuario no se encuentre registrado, deberá crear una cuenta presionando el botón Registrarse, el cual lo llevará a la Activity de Registro.

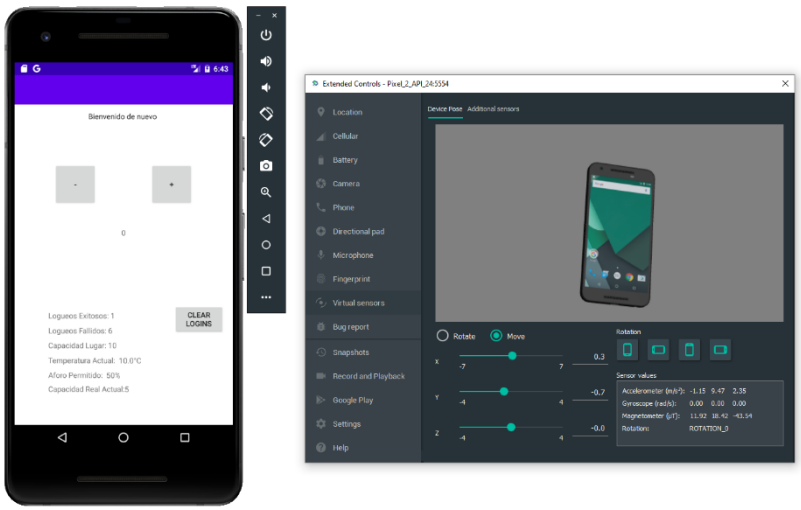
Dentro del formulario de registro se deben completar todos los campos y luego presionar el botón registrarse. En el caso de que el registro sea exitoso aparecerá el mensaje “cuenta registrada correctamente” y se llevará a la MainActivity. Caso contrario, si existe un error en los datos o se utilice algun dato que ya se haya registrado en otro momento, se informará el mensaje por la pantalla



La MainActivity consta de un mensaje de bienvenida, dos botones (mas y menos), un contador, y una serie de datos. Para contar las personas que ingresan o egresan a un evento, se utilizan los botones (mas en caso de ingreso de una persona y menos en caso de un egreso). Estos botones iran sumando o restando al contador, el cual estará limitado por los datos que están debajo de la aplicación:
Capacidad del lugar: Es la capacidad fija del lugar del evento. Esta seteada en 10 debido a que se pensó en un evento pequeño, por ej. un museo pequeño.
Temperatura Actual: Es la temperatura ambiente del lugar. Dicha temperatura se actualiza según los datos que provea el sensor. Debido a que la mayoría de celulares no tiene dicho sensor, se recomienda probar la app con un emulador.
Aforo permitido: Como se explico en la introducción, variara según la temperatura ambiente.
Capacidad Real Actual: es la capacidad maxima de personas que pueden ingresar al evento en ese momento. Varia en función de la temperatura/aforo permitido. El contador no podrá sobrepasar este valor.
También, vemos la cantidad de Logueos Exitosos y Fallidos desde que se instalo la app. Este contador puede ser reiniciado con el botón Clear Logins.



Al sacudir rápidamente el dispositivo (como un shake), por medio del sensor acelerómetro que detecta dicho movimiento, el contador de personas se reiniciará a 0.



3 Conclusiones

1. Comente acerca de los recaudos que tuvo que realizar, para que la aplicación sea tolerante a fallos
Se tuvo que validar varias cuestiones, como los inputs de los EditTexts (que no esten vacios), los casteos en los tipos de datos, pasar los contextos adecuados, calibracion de sensores (por codigo) para que detecten bien los cambios, pensar bien como realizar la estructura del proyecto (definir bien cuales iban a ser las clases y como iban a interactuar mediante el patron MVP). Chequear la conexión a internet antes de enviar mensajes al servidor. Configurar bien los intent y los ciclos de vida de cada Activity (que cada Activity invoque o cree a la que corresponde), entre otras cosas.

2. Durante el desarrollo ¿Surgieron problemas? ¿Cómo fueron resueltos? Detallen las lecciones aprendidas durante el desarrollo.
El problema mas grande que tuvimos fue que cada uno de los estudiantes se centró en una parte particular del proyecto y lo desarrollo a su modo de hacer las cosas, pero al momento de juntar esos desarrollos en el proyecto se hizo complejo debido a que alguno pensó pasar los datos de una forma y otro pensó recibirlos de otra, sumado a que se trataba de un proyecto que ninguno de los tres había trabajado antes (específicamente el desarrollo de una aplicación mobile). Se pudo solucionar con reuniones/llamadas cuando cada uno tenia su desarrollo ya avanzado y se pudo definir sobre la marcha como implementar y acoplar dichos desarrollos. La lección aprendida mas grande es que se debió definir de antemano como iba a desarrollar cada uno su parte para así al momento de acoplar todo en el proyecto no se tenía que estar cambiando cosas sobre la marcha y así poder llegar a una solución mas limpia y clara de cara a la entrega final.

4 Referencias

1. Ing. Esteban A. Carnuccio, Ing. Mariano Volker, Ing. Raúl Villlca, Ing. Matías Adagio: “Apunte Teórico sobre el Sistema Operativo Android” San Justo, (2021).

2. Tin Megali “How to Adopt Model View Presenter on Android”(2016): https://code.tutsplus.com/tutorials/how-to-adopt-model-view-presenter-on-android--cms-26206.

3. Android developers (2021), https://developer.android.com/.