UNIVERSIDADE DA CORUÑA

PROGRAMACIÓN DE SISTEMA 21/22 Q1

Desarrollo de una aplicación de control de aforo en Android

Portiñas

Autores: Brais Barboza Ordoñez
Julián Barcia Facal (julian.bfacal@udc.es)
Fecha:*A Coruña, 13 de enero 2022*
Versión:*3.0*

Índice

Capítulos **Página**

Cuadro 1: Tabla de versiones.

Versión	Fecha	Autor
1.0	06/10/2021	Julián y Brais
2.0	09/11/2021	Julián y Brais
3.0	13/01/2022	Julián y Brais

1. Introducción

Esencialmente la aplicación controlará el acceso del personal de un edificio. Con una estación de control en cada acceso, el personal de forma casi transparente podrá acceder al edificio o marcharse del mismo. El personal contara con un dispositivo con la tecnología NFC (mismamente su teléfono) el cual a la hora de entrar al edificio escanearan contra la aplicación de la puerta, que les garantizara el acceso en caso de cumplir los requisitos necesarios.

1.1. Objetivos

El objetivo principal de este trabajo es gestionar el control de acceso de un edificio o recinto genérico, de tal modo que se tenga un control del número de personas que entran y salen de dicho recinto. Además, buscaremos también crear un lector de identificación, de tal modo que cada usuario pueda ser reconocido de manera individual e inequívoca. De esta manera mediante un teléfono, a través de la app, podremos agilizar los trámites de aforo y acreditación.

1.2. Motivación

Debido a la situación de pandemia en la que nos encontramos el control de aforo se ha vuelto un requisito indispensable en muchas ocasiones para poder cumplir las legislaciones continuamente cambiantes. Esta aplicación busca servir, de una manera simplificada, como ayuda para cubrir estas nuevas necesidades que la pandemia ha generado consigo.

1.3. Trabajo relacionado

Antes de realizar este trabajo se ha buscado información sobre otros proyectos similares, con objetivos y motivaciones similares a las nuestras o que simplemente nos sirvan como un punto de referencia para nuestro diseño. En un primer lugar nos centramos en tener una idea general de lo que podría abarcar este tipo de proyecto. Tomando como referencia otros trabajos [1], se ha conseguido clarificar cuáles pueden ser los requisitos e incidencias, a pesar de que el producto final no sea exactamente como el nuestro.

Una vez sentadas las bases, se ha hecho uso de proyectos que nos sirvieran como clara referencia para nuestra implementación. Ideas para una interfaz de mayor sencillez y accesibilidad o como hacer un almacenamiento de los datos de forma más eficiente han sido ideas que se han consultado en [2].

Dado que se va a hacer uso de una base de datos en tiempo real, en concreto la que nos proporciona *Firebase*, se han indagado sobre posibles trabajos que tuvieran el uso del servicio de *Firebase* y su base de datos como objetivos principales. De este modo el trabajo de Marina Castellote García [3], ha servido como buena referencia para estas labores.

Por último, como buscamos que nuestro sistema sea capaz de utilizar el sistema de RFID para la identificación de usuarios a su entrada y salida del aforo, el trabajo de Marcos Manuel Gallego Ferrer [4], es una gran referencia no tan solo por su descripción e implementación de los sistemas de RFID ya mencionados sino también por las grandes similitudes en el producto final y objetivos que se pretenden obedecer.

2. Análisis de requisitos

2.1. Funcionalidades

La aplicación como funcionalidad central tiene el control de aforo de un recinto. Si bien esta idea es bastante simple en un primer momento, la esencia del proyecto reside en su implementación. Sobre un método clásico, como un usuario controlando manualmente el aforo con la ayuda de la app, se proponen una serie de funcionalidades que se implementarán de forma progresiva:

- Posibilidad de configurar el edificio, controlando las entradas y las salidas además del aforo
- Actualizar el estado (entrada o salida) del sistema mediante dispositivos móviles con tecnología NFC.
- Mantener un histórico a disposición de la administración de los accesos y salidas del recinto.
- Identificar cada histórico con un valor único, pudiéndose acceder a ellos mediante el uso de este.
- Posibilidad de borrar y cambiar de base de datos y su identificador, teniendo otra automáticamente, completamente nueva y con otro identificador.
- Informar al usuario en su dispositivo en caso de que un intento de acceso haya sido denegado.

2.2. Prioridades

El proyecto tiene como idea central la escalabilidad del mismo. Siendo una idea sencilla permite una alta cantidad de incrementos en función de los requisitos de una organización. Por ello las prioridades serán:

1. Crear un histórico local en un único acceso mediante un único terminal de forma manual.
2. Implementación de un sistema de sincronización entre varios terminales para pasar a tener un histórico combinado, ampliando así el posible número de entradas y salidas, además de que el histórico está almacenado en *cloud*.

3. Identificación de cada sistema de manera única de tal modo, que se pueda conectar a su base de datos en cualquier momento con el uso de su identificador.

4. Implementación de una aplicación para el acceso mediante dispositivos individuales, permitiendo el control en varios accesos de modo automático.

5. Registro de intentos de acceso y salida.

Cabe destacar que independientemente de las prioridades previamente descritas, el proyecto pretende aumentar de tamaño capa a capa, de forma que cada nivel individual sea plenamente funcional por sí mismo.

3. Planificación inicial

3.1. Iteraciones

1. Implementar una aplicación que muestre el aforo en el momento actual e incremente o decremente de manera manual, según las personas salgan o entren.

2. Utilizar NFC [5] para ser capaces de identificar a cada persona que entre de manera individual.

3. Conseguir que este recuento se haga de manera automática al detectar la entrada y salida de usuarios.

4. Hacer uso de una base de datos en tiempo real, de tal modo que los accesos y salidas se contabilicen en linea, de forma que el aforo esté actualizado y sincronizado de forma simultánea en varios puntos de acceso y salida.

5. Mantener un registro de accesos al registro a nivel individual.

6. Posibilidad de creación de grupos con franjas horarias de accesos.

3.2. Responsabilidades

La estructura del proyecto y la repartición de tareas se han hecho de la siguiente manera:

1. La creación del modelo inicial, compuesto por un contador que guardaba sus datos en modo local, y con un funcionamiento plenamente *offline* a quedado a cargo de Julián.

2. La extrapolación de este modelo a un contexto en línea, mediante el uso de una base de datos en tiempo real *(Firebase)* ha quedado a cargo de Julián.

3. La identificación de cada empleo de la aplicación, con un código único y el cual se vincula con una base de datos propia en *cloud* ha sido implementada por Julián.

4. La identificación de cada dispositivo, mediante el uso de NFC ha quedado a cargo de Brais.

5. La adaptación de la aplicación a un esquema como el *Model View Controller* ha quedado a manos de Brais.

6. El desarrollo de una capa *front-end*, sencilla y entendible para un nuevo usuario se ha hecho de forma conjunta, ayudándose de *testers* o terceros, teniendo su interacción con la aplicación como referencia para cambios significativos.

7. La actualización de este documento se ha hecho de manera mutua, añadiendo cada uno de los integrantes aspectos y matices reseñables, a medida que se avanzaba en el desarrollo del proyecto.

3.3. Hitos

El desarrollo del proyecto se va a organizar en *sprints*. Este modelo es el que más se adapta a la idea inicial de nuestro proyecto. Además, su uso compagina en gran medida con el uso de *git-flow* y su modelo de control de versiones. Cada Sprint en nuestro caso se corresponderá con secciones de desarrollo importantes junto con diversos *features*.

- *Fase de planificación:* Esta fase se centrará en crear una primera versión del proyecto, diseñando de manera esquemática funcionalidades e interfaz que serán utilizadas próximamente. En esta sección se incluye también la creación de este documento, que será actualizado a lo largo del proyecto

- *Fase de desarrollo de la aplicación offline:* Esta fase centra el foco en integrar las funcionalidades y el diseño de la anterior iteración. Al final de esta etapa, se podría un realizar un test para comprobar el correcto funcionamiento de las funciones principales de la aplicación. El test se hará de manera manual, sobre un dispositivo físico, para evitar errores que puede ocultar el emulador.

- *Integración de base de datos en tiempo real:* En este apartado se busca que una vez construida la base sobre un modelo local, consigamos transmitir los datos a una base de datos en *cloud*. El correcto funcionamiento de este se hará tanto en el dispositivo como a nivel *online*. A nivel local los cambios efectuados deben de seguir siendo visibles. A su vez, los datos deben sufrir los mismos cambios en la base de datos en tiempo real y estos se deben poder consultar con el simple uso de un navegador web.

- *Identificación y conexión de dispositivos:* Buscaremos en este apartado que cada ejecución de la aplicación cuente con un código único vinculado también a su propia base de datos en tiempo real. Con el conocimiento de este código, se debe permitir también la conexión entre dispositivos de tal modo que sincronicen sus datos en una única base de datos.

- *Integración del sistema de lectura:* Esta sección se focalizará en integrar el método de NFC en las funcionalidades de la aplicación. Al finalizar esta tarea se podría realizar un test para comprobar la correcta lectura y reconocimiento de los dispositivos.

- *Información de la petición en el dispositivo individual:* Informar al usuario de si su intento de acceso ha sido aprobado, o de la denegación del mismo con una explicación breve.

- *Pruebas:* A medida que se avanzaba en la parte de desarrollo, se realizarán pruebas para verificar el funcionamiento del modelo planteado y comprobar con qué ajustes sería más óptimo. Paulatinamente se intensificarán las pruebas para poder obtener una mayor cantidad de datos y así poder decidir que avances priorizar. Al finalizar el desarrollo total, se realizarán pruebas de resistencia para comprobar la estabilidad del sistema

- *Posibles nuevas funcionalidades y cierre del proyecto:* Esta última sección se centrará en añadir nuevas tareas, pero de pequeño tamaño que no pueden comprometer el cierre final del proyecto. Esta última tarea viene acompañada de pruebas sobre el funcionamiento completo de la aplicación.

3.4. Incidencias

Si bien la naturaleza del proyecto es simple en planificación, el desarrollo de un sistema completo puede resultar de mayor dificultad de la prevista. Estos problemas se traducen en unos retrasos en las planificaciones debido al aumento de los tiempos dedicados a cada tarea. De este modo, durante el desarrollo de este proyecto se han producido una serie de incidencias:

- *Uso de ProgressBar:* Durante la realización de este proyecto, para crear una interfaz más visual como se ha dicho la cuenta del aforo se acompaña de un círculo dinámico. El problema encontrado con este elemento es su implementación debido al desconocimiento previo de su funcionamiento para el equipo de desarrollo. Variantes fuentes [6], [7] serán empleadas para solventar este cometido.

- *Uso de Real Time Database:* Para la implementación síncrona del programa, se necesita hacer uso de una base de datos en tiempo real [8], al igual que el caso anterior, el equipo de desarrollo no cuenta con conocimientos previos por lo cual se debe utilizar cierto tiempo en su aprendizaje.

- *NFC:* Sumado a esto, la comunicación NFC puede traer consigo distintas problemáticas, entre ellas la más evidente: se necesitarán varios dispositivos con dicha característica para poder comprobar el funcionamiento del sistema.

- *Ajustes:* En esta aplicación se hace uso de una sección de ajustes, implementada con una *Preference Activity* y su *fragment* asociado. La sincronización

del campo de aforo que se muestra en dichos ajustes, con el guardado de forma inicial y con los diferentes elementos de la interfaz que deben cambiar ha resultado de mayor complicación de la prevista, produciendo retrasos en la producción.

4. Diseño

4.1. Arquitectura

Se diseñará el proyecto con la arquitectura CLEAN en mente. Aprovechándonos de la orientación a eventos de Android, implementaremos un modelo vista presentador, donde un evento en el presentador, modificara la vista y llamara al modelo. En cuanto a la sincronización entre dispositivos, se implementará mediante el uso de un servidor principal con una interfaz API REST.

4.2. Persistencia

Como se ha comentado en un primer momento solo tenía soporte para guardar sus datos a nivel local, de tal modo que la existencia de varias entradas y salidas no se contemplaba. Con el uso de la base de datos en tiempo real, los datos de entrada, salida e identificación de dispositivo ahora se guardan en línea. Esto nos permite, por lo tanto, que varios dispositivos que se encuentran usando la aplicación tenga conexión y relación entre ellos.

4.3. Vista

Con respecto a la vista nuestra aplicación prioriza la simplicidad y sencillez por lo que nos centraremos en el uso de modelos presentes en multitud de otras aplicaciones, haciendo el uso de la nuestra mucho más intuitivo para el usuario:

1. ***Splash-Screen:*** Al ejecutar la aplicación, esta muestra durante un breve periodo de tiempo un *Splash-Screen* con el logo de la aplicación.

Figura 1: Splash Screen

2. ***Introducir aforo:*** Solo en la primera ejecución de la aplicación o si no se ha introducido un valor de aforo total todavía, la aplicación nos mostrará una pantalla correspondiente a la actividad que cubre este propósito.

Figura 2: Introducir Aforo

3. ***Fragmentos:*** Para las funcionalidades principales en esta aplicación se va a hacer uso de distintos fragmentos:
    - *Contador de aforo:* Este fragmento es el principal y es el que muestra los datos almacenados en la base de datos con respecto al aforo. Además, para mayor claridad visual se acompañan de una *ProgressBar*, que nos da una idea del porcentaje de ocupación. Justo debajo se sitúan dos botones que son los que nos proporcionan las funcionalidades de registrar una salida y entrada respectivamente.

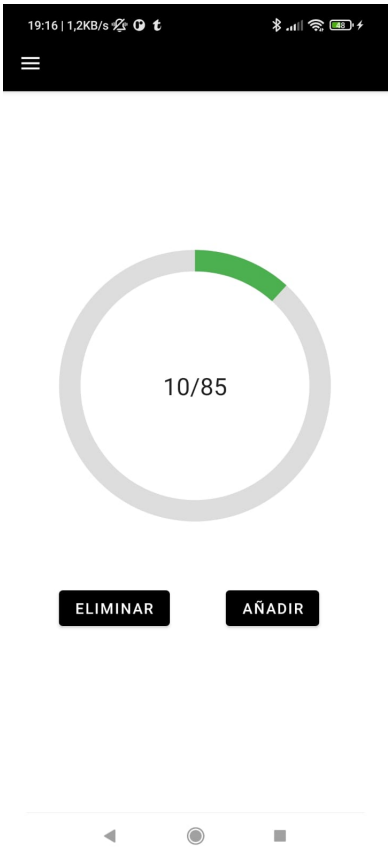

Figura 3: Fragmento con contador en tiempo real de aforo

    - *Código identificador:* En este fragmento el usuario podrá ver el código que le corresponde a su dispositivo. En esta misma vista el usuario cuenta con un botón al lado de un campo rellenable. Estos permiten al usuario conectarse a otro dispositivo, a su base de datos, para sincronizar sus valores y cambios. Además, se cuenta con un botón adicional para reiniciar el identificador y su base de datos vinculada.

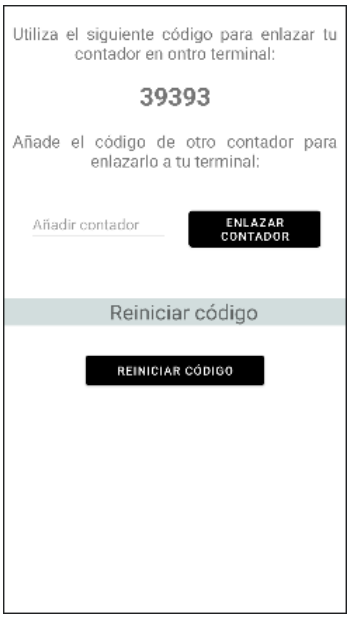

Figura 4: Fragmento con código identificador

- *NFC:* En este fragmento el usuario podrá ver el código que le corresponde a su dispositivo. En esta misma vista el usuario cuenta con un botón al lado de un campo rellenable. Estos permiten al usuario conectarse a otro dispositivo, a su base de datos, para sincronizar sus valores y cambios. Además, se cuenta con un botón adicional para reiniciar el identificador y su base de datos vinculada.

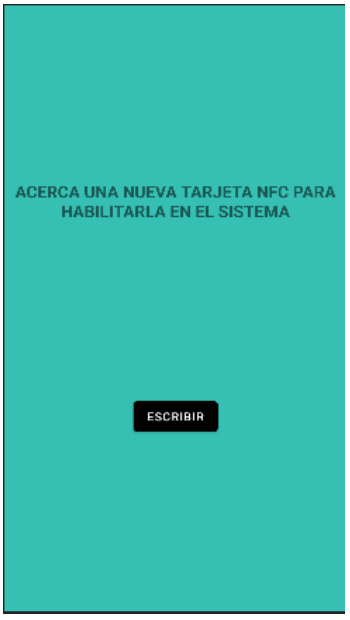

Figura 5: Fragmento para conectar NFC

El botón de regreso en cualquier de estos fragmentos devuelve un Toast indicando al usuario que presionando de nuevo cerrará la aplicación.

4. ***Navigation Drawer:*** Para el simplificar el acceso a los diversos fragmentos y otras funcionalidades se hace uso de un *Navigation Drawer* que permite al usuario saltar entre estas distintas secciones. En él además de los fragmentos anteriormente mencionados también se encuentran:
   - *Ajustes*: Contamos con un fragmento adicional para ajustes. En él se puede modificar el aforo total del edificio. Al presionar el botón de regreso el usuario volverá al fragmento principal, donde podrá ver que se han realizado los cambios. En el caso de que la capacidad total que se haya introducido sea menor que el recuento del número de personas que se encuentran en el interior el sistema reinicia este recuento a 0, en el caso general simplemente modifica la capacidad manteniendo el conteo.
   - *Compartir*: Este elemento mediante el uso de un intent le mostrará al usuario diversas opciones para compartir el enlace del proyecto.
   - *Sobre Nosotros*: Este elemento derivará al usuario directamente a la dirección web de la wiki de nuestro proyecto, abriendo mediante el uso de intent en el navegador dicha página.

Figura 6: Fragmento de ajustes para modificar aforo total

Figura 7: Header Navigation Drawer

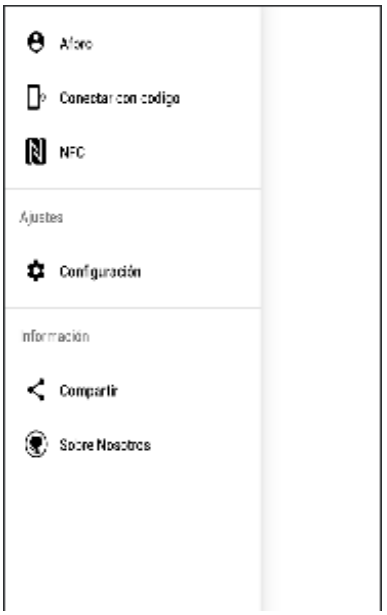

Figura 8: Opciones menú

4.4. Comunicaciones

Para mantener una base de datos en tiempo real, en la cual podamos guardar toda la información requerida para el correcto uso de la aplicación haremos uso de *Firebase* [9], más concretamente vamos a hacer uso de su opción de *Real Time Database* [8]. Como el desarrollo de esta aplicación solo tiene fines didácticos y no se pretende hacer pública, las reglas de seguridad de esta base de datos se pueden definir sencillamente como públicas, a pesar de los riegos de seguridad que esto conlleva.

4.5. Sensores

Para esta aplicación y más concretamente, para su funcionalidad de lectura automática, vamos a hacer uso del sensor de NFC. Como se comenta previamente en el documento, existe la posibilidad de registrar ,los accesos y salidas al sistema mediante tarjetas NFC. La aplicación cuenta con una pantalla propia para la habilitación de las mismas que graba un identificador único en cada una, con el fin de registrar cuando esta se encuentra dentro o fuera del recinto. En cuanto a su funcionamiento, la lectura de una tarjeta por el dispositivo, genera un Intent, que, de estar abierta la aplicación sera procesada como un ingreso. En caso de estar en la actividad de activación, el dispositivo se pone en modo escritura, dando feedback cuando la escritura sea completada, tanto siendo correcta como fallida.

4.6. Trabajo en background

Ya que en Android las peticiones externas se deben ejecutar fuera del hilo principal, las interacciones con el servidor, se implementan mediante listeners.

- *OnComplete*: Que se ejecuta cuando se completa la petición get().
- *OnDataChanged*: Que se llama al detectar un cambio en la base de datos actualizando los valores en pantalla de los valores asociados.

4.7. Pruebas Realizadas

Se ha optado por un desarrollo del testeo manual, probando las funcionalidades a medida que se implementaban, comprobando los distintos casos de uso. Esto se debe a que la envergadura del proyecto no justifica la implementación de un testeo automático.

5. Trabajo Futuro

A pesar de que la aplicación cumple con los requisitos y la planificación más básica e importante, existen ciertos aspectos y funciones que se podrían considerar e implementar en próximas revisiones:

- *Identificación*: En futuras implementaciones se podría considerar que la identificación de los usuarios se haga gracias a valores como un correo o nombre de usuario los cuales el usuario tenga que introducir para hacer uso de la aplicación, sustituyendo el código de 5 dígitos actual.
- *Más Información*: La información de las entradas y salidas es reducida en este momento, estas solo se contabilizan como un número sobre un total, no dan información sobre las horas de estos sucesos. Estos datos pueden ser da gran utilidad por lo cual avances en este sentido pueden resultar recomendables.
- *Denegación o admisión de acceso*: Una vez incluida más información sobre los usuarios, la denegación o admisión de estos en el recinto se podrá hacer considerando más factores. Ejemplos como la admisión solamente en determinadas franjas horarias o solo con permiso de acceso especial podrían ser de gran utilidad en el uso real.
- *Implementación alternativa*: A pesar de contar con el uso de NFC en nuestra implementación y la implementación manual, el uso de otra alternativa o método puede ser de gran valor. Por esto, el empleo de códigos QR por ejemplo, que identifiquen y contabilicen al usuario pueden resultar una buena opción a considerar en el futuro.

- *Implementación de Pruebas automáticas*: Para futuros desarrollos de la aplicación es fundamental el uso de bibliotecas de testeo automáticas como jUnit.

Referencias

[1] López, A. M., "Diseño y programación de una aplicación móvil para la gestión de información en bares y restaurantes." https://idus.us.es/bitstream/handle/11441/114919/TFG-3397-MORA

[2] Víctor Tello Carrascal, Jesús Álvarez Coll, M. A. M. D. G. G., "Sistema de control de aforo en espacios cerrados." [Archivos de video] Disponible en: https://eprints.ucm.es/id/eprint/66909/1/%C3%81LVAREZ%20COLL%2051203_JESUS_ALVAREZ_COLL_TFG_164_GII_GIC_Sistema_de_Control_de_Aforo_en_Espacios_Cerrados_2021_784051_747739898.pdf, 2020.

[3] GARCÍA, M. C., "Desarrollo de una aplicación android de apuestas utilizando firebase para la sincronización de datos." Disponible en: http://repositori.uji.es/xmlui/bitstream/handle/10234/174192/TFG_2017_Castellote%20Garcia_Marina.pdf?sequence=1, 2017.

[4] Gallego Ferrer, M. M., "Safe events: control de aforos mediante rfid." Disponible en: https://m.riunet.upv.es/bitstream/handle/10251/89155/GALLEGO%20-%20Safe%20Events%3A%20control%20de%20aforos%20mediante%20RFID.pdf?sequence=1&isAllowed=y, 2017.

[5] "Conceptos básicos de nfc." Disponible en: https://developer.android.com/guide/topics/connectivity/nfc/nfc?hl=es-419.

[6] "Funcionamiento progressbar." Disponible en: https://developer.android.com/reference/android/widget/ProgressBar.

[7] "Progressbar en android." Disponible en: https://www.develou.com/progressbar-en-android.

[8] "Documentación real time database firebase." Disponible en: https://firebase.google.com/docs/database?hl=es.

[9] Moure, B., "Firebase android kotlin + android studio." [Archivos de video] Disponible en: https://www.youtube.com/playlist?list=PLNdFk2_brsRcaGhfeeiVkW72qTYcn_nfQ, 2020.