Glosario de clases

Victor Gerardo Rodríguez Barragán

05 de septiembre de 2023

Contents

1 Practica 01 - Notificaciones

1.1 MainActivity.java

Es la activity que se crea por defecto en un nuevo projecto en Android studio en esta activity yo realizo las conexiones hacia otras activity y hago la autentificacion del login.

1.2 DashboardActivity.java

Es la activity que se crea por defecto en un nuevo projecto en Android studio en esta activity yo realizo las conexiones hacia otras activity con el uso de botones, uno es para mandar una notificacion y el otro para realizar un cierre de sesion.

1.2.1 onClick

Es un metodo que se utiliza para realizar una accion cuando se presiona un boton en este caso solo utilizo el boton de logout para cerrar sesion, para mandar la notifcacion utilizo un metodo que se llama **displayNotification** el cual crea una notificacion llamando a la clase **NotificationHelper** y le pasa como parametros el titulo de la notificacion y el mensaje que se va a mostrar.

1.3 NotificationHelper.java

Es una clase que se crea para poder crear notificaciones, en esta clase se defino los id de las notificaciones y canales, tambien se definen los metodos por defecto para crear una notificacion.

1.3.1 IDs

Para la seccion de IDs tenemos que tener 2 la ID para notificaciones con: **NOTIFICATION_CHANNEL**: Es una variable constante que sirve para identificar el canal o el "medio" por el cual se agrupan y configuran las notificaciones. **CHANNEL_ID**: es una variable constante que se utiliza para identificar el canal o el "medio" por el cual se agrupan y configuran las notificaciones.

1.3.2 createChannel()

Es un metodo que se utiliza para crear un canal de notificacion, en este metodo se definen las caracteristicas del canal como el nombre, la descripcion y la importancia de las notificaciones que se van a mostrar en este canal, asi como otros parametros para personalizar el canal, aunque primero que nada se verifica que la version de android sea mayor o igual a la version Oreo.

1.3.3 getChannelNotifcation(title, message)

Es un metodo para agrupar las notificaciones en un canal, en este metodo se definen las caracteristicas de la notificacion como el titulo, el mensaje, el icono, el sonido, la vibracion, el color, la prioridad, la visibilidad, etc, creo uno de estos por cada canal que se va a utilizar.

1.3.4 setPendingIntent()

Es una funcion que se utiliza para realizar una accion cuando se presiona una tecla dentro de la notificacion, en mi caso solo existen las opciones de **SI** si es algo urgente y **NO** si no es algo urgente, en caso de presionar **SI** se abre la activity de **EmergenciesActivity** y en caso de presionar **NO** se abre la activity de **RegisterActivity**.

1.4 EmergenciesActivity.java

Es una activity simple, pensada en que se va a utilizar para registrar a un paciente en caso de que sea una emergencia, por lo tanto contiene un formulario corto y vital para poder dar atencion urgente al paciente.

1.5 RegisterActivity.java

Es un formulario para registrar a un paciente, en este formulario se piden datos como el nombre, la edad, el sexo, el tipo de sangre, entre otras cosas que son importantes para poder dar la atencion adecuada al paciente, en este caso mando una notificacion extra para avisar que se ha registrado a un paciente y pronto se le dara atencion.

2 Practica 02 - Alarmas

En esta practica se utilizo el mismo codigo que en la practica 01, solo se realizo una modificacion en la clase **RegisterActivity**, en esta clase se agregaron dos botones para poder programar una alarma, uno para programar uno establece la fecha y el otro para establecer la hora.

2.1 setDateAlarm()

Metodo que se utiliza para establecer la fecha de la alarma, en este metodo se usa la libreria **Calendar** para poder seleccionar la fecha y se muestre al usuario en un **DatePickerDialog** el cual es un calendario simple.

2.2 setHourAlarm()

Muy similar al metodo anterior, solo que en este caso se utiliza la libreria **TimePickerDialog** para poder seleccionar la hora de la alarma.

2.3 programAlarm()

Es un metodo que se utiliza para programar la alarma, se pasan como parametros extra el mensaje, la hora y los minutos en el que se realizara la alarma, acto seguido se abre como "activity" el programa de reloj de android con la alarma ya programada.

3 Practica 03 - Firebase

En esta practica se utilizo el mismo codigo que en la practica 01 y 02, solo se realizo una modificacion en la clase **RegisterActivity**, en los cuales se agrego un boton para poder guardar los datos del paciente en la base de datos de **Firebase**.

3.1 token

Es una variable que se utiliza para guardar el token del dispositivo, este token se obtiene de la consola de **Firebase** y se guarda en la variable para poder mandar la notificacion a ese dispositivo.

3.2 sendPushNotification()

Es un metodo que se utiliza para mandar una notificacion a un dispositivo usando **Firebase**, en este metodo se definen las caracteristicas de la notificacion.

3.3 getHeaders()

Es un metodo que se utiliza para obtener los headers de la notificacion, en este caso solo se utiliza el **Content-Type** y el **Authorization**.

3.4 onNewToken()

Es un metodo que se utiliza para obtener el token del dispositivo, en este metodo se guarda el token en la variable **token**.

3.5 saveToken()

Es un metodo que se utiliza para guardar el token en la base de datos de **Firebase**, en este metodo se obtiene el token y se guarda en la base de datos.

3.6 onMessageReceived()

Es un metodo que se utiliza para recibir una notificacion, en este metodo se obtiene el titulo y el mensaje de la notificacion y se muestra en un **Toast**.