



SIM CARD

TIME 06



TIME 06



ANDREINA OLIVEIRA

Scrum Master



ANDRÉ RONDÍ

Developer



EDIVALDO JUNIOR

Developer



STHEFANYE OLIVEIRA

Developer



WESLLEN COSTA

Developer



PRODUCT OWNER'S

ÁUREA MELO

UEA

OSCAR NETO

Eldorado

EDGARD SILVA

UEA

FERANDO ITO

Eldorado



Objetivo

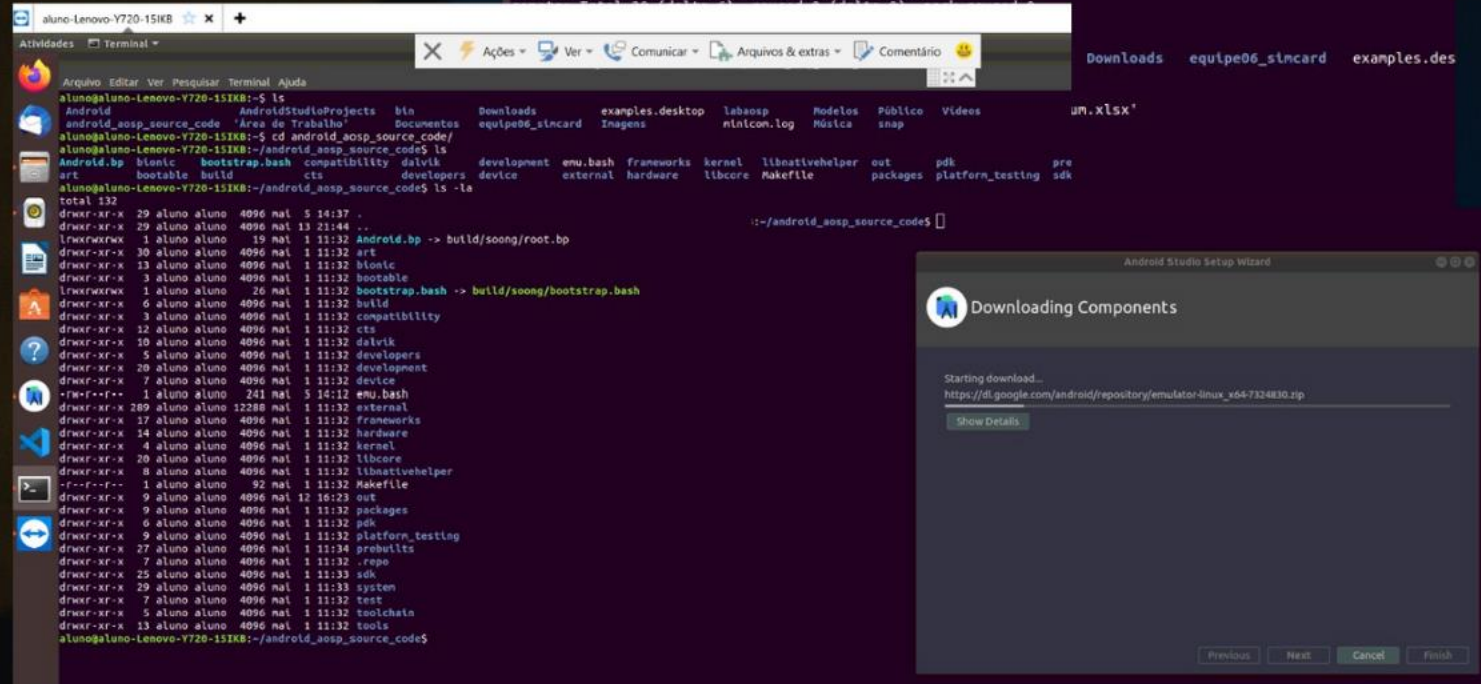
Adicionar no AOSP um recurso que salve um identificador do SIM Card para detectar e notificar se houver troca por outro SIM Card.

Critérios de Aceitação


- Identificar ICCID.
- Detectar trocas de SIM Card.
- Notificar trocas de SIM Card.
- Controle de SIM Card à nível AOSP.

AMBIENTE

Git
aosp_x86_64-eng
Visual Studio
Android Studio
Android 9.0 Pie
API Level 28



PESQUISAR API

developers  Plataforma Android Studio Google Play Jetpack Mais ▼ Portuguê... ▼ [Fazer login](#)



AndroidXRef Pie 9.0.0_r3

href: /frameworks/base/telephony/java/android/telephony/TelephonyManager.java

Home | History | Annotate | Line# | Navigate | Download ☐ only in **TelephonyManager.java**

```
1 /*
2  * Copyright (C) 2008 The Android Open Source Project
3  *
4  * Licensed under the Apache License, Version 2.0 (the "License");
5  * you may not use this file except in compliance with the License.
6  * You may obtain a copy of the License at
7  *
8  *     http://www.apache.org/licenses/LICENSE-2.0
9  *
10 * Unless required by applicable law or agreed to
11 * distributed under the license is distributed on
12 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, e
13 * See the license for the specific language gover
14 * limitations under the license.
15 */
16 package android.telephony;
17
18 import static com.android.internal.util.Preconditi
19
20 import android.annotation.IntDef;
21 import android.annotation.Nullable;
22 import android.annotation.RequiresPermission;
23 import android.annotation.SdkConstant;
24 import android.annotation.SdkConstant.SdkConstantT
25 import android.annotation.SuppressAutoDoc;
26 import android.annotation.SuppressLint;
27 import android.annotation.SystemApi;
28 import android.annotation.SystemService;
29 import android.annotation.TestApi;
30 import android.annotation.WorkerThread;
31 import android.app.ActivityThread;
```

Desenvolvedores Android > Docs > Reference

Classificar e avaliar  

TelephonyManager Added in API level 1

Kotlin | Java

```
public class TelephonyManager
    extends Object

java.lang.Object
    ↳ android.telephony.TelephonyManager
```

IDENTIFICAR ID SIM CARD

```
@RequiresPermission(android.Manifest.permission.READ_PHONE_STATE)
public String getSimSerialNumber(int subId) {
    try {
        iPhoneSubInfo info = getSubscriberInfo();
        if (info == null)
            return null;
        return info.getIccSerialNumberForSubscriber(subId, mContext.getOpPackageName());
    } catch (RemoteException ex) {
        return null;
    } catch (NullPointerException ex) {
        // This could happen before phone restarts due to crashing
        return null;
    }
}
```



RECUPERAR ID SIM CARD

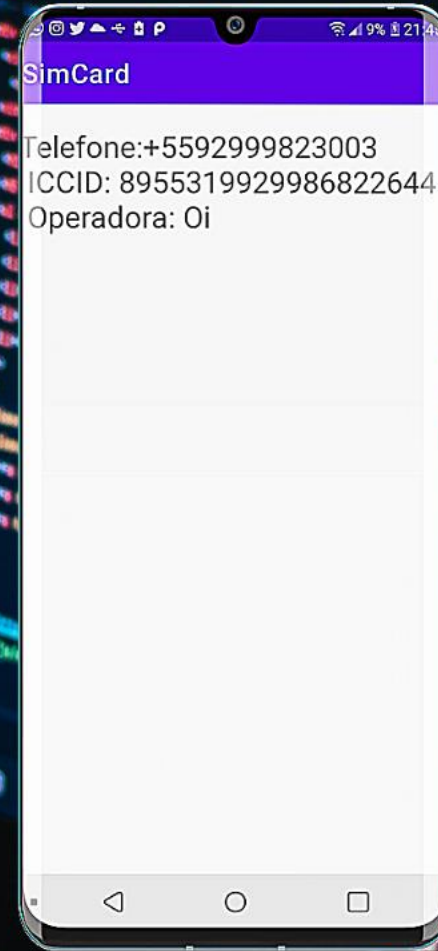
```
@SuppressLint(...value: "MissingPermission")
fun getNumber(): String {
    val telephonyManager = this.getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager
    val number = telephonyManager.line1Number
    Log.i(TAG, msg: "Numero: $number")
    return number
}

@SuppressLint(...value: "MissingPermission")
fun getIccId(): String {
    val telephonyManager = this.getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager
    val iccid = telephonyManager.simSerialNumber
    Log.i(TAG, msg: "ICCID: $iccid")

    return iccid
}

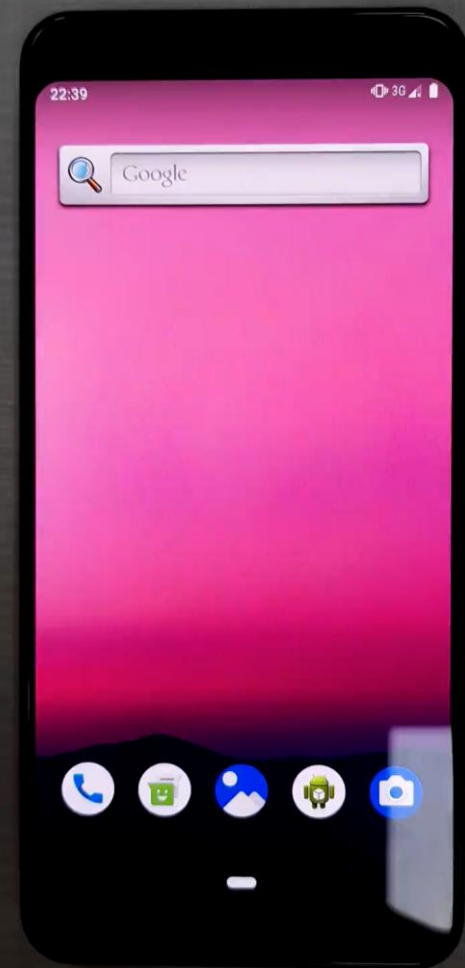
@SuppressLint(...value: "MissingPermission")
fun getOperator(): CharSequence? {
    val telephonyManager = this.getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager
    val operator = telephonyManager.simCarrierIdName
    Log.i(TAG, msg: "Operadora: $operator")

    return operator
}
```

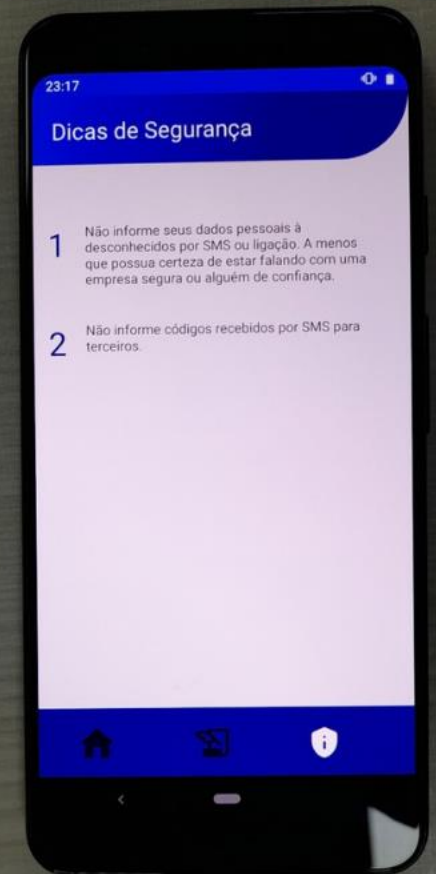
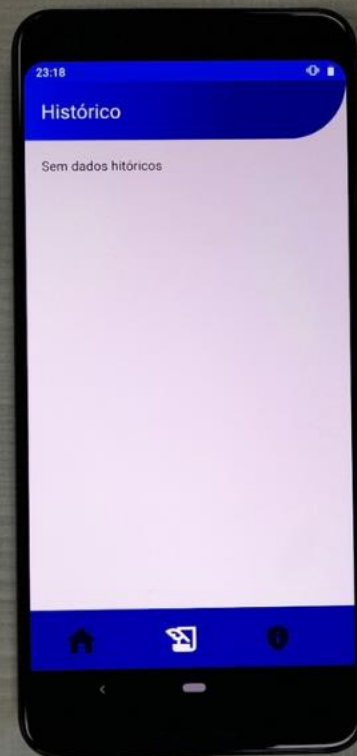


PERMISSÃO APP

```
private fun getPermission(): Boolean {  
    return (ContextCompat.checkSelfPermission(  
        this,  
        Manifest.permission.READ_PHONE_STATE  
    ) == PackageManager.PERMISSION_GRANTED)  
}  
  
private fun setPermission() {  
    val permissionsList = listOf<String>(  
        Manifest.permission.READ_PHONE_STATE  
    )  
    ActivityCompat.requestPermissions(this, permissionsList.toTypedArray(), 1)  
}
```



LAYOUT ACTIVITY'S



LAYOUT ACTIVITY'S

```
override fun onClick(v: View) {  
    when (v.id) {  
        R.id.btHome3 -> {  
            val intent = Intent(applicationContext, com.example.simcard.MainActivity::class.java)  
            startActivity(intent)  
        }  
        R.id.btHistory3 -> {  
            val intent = Intent(applicationContext, com.example.simcard.historico::class.java)  
            startActivity(intent)  
        }  
    }  
}
```



DETECTAR TROCAS

≡ developers 

See also:

`SIM_STATE_UNKNOWN`

`SIM_STATE_ABSENT`

`SIM_STATE_PIN_REQUIRED`

`SIM_STATE_PUK_REQUIRED`

`SIM_STATE_NETWORK_LOCKED`

`SIM_STATE_READY`

`SIM_STATE_NOT_READY`

`SIM_STATE_PERM_DISABLED`

`SIM_STATE_CARD_IO_ERROR`

`SIM_STATE_CARD_RESTRICTED`

TelephonyManager.java

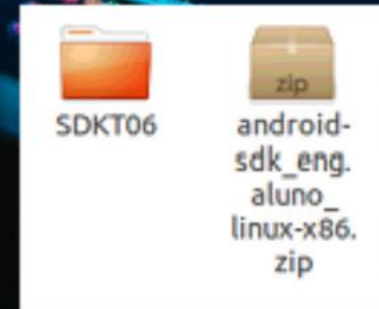
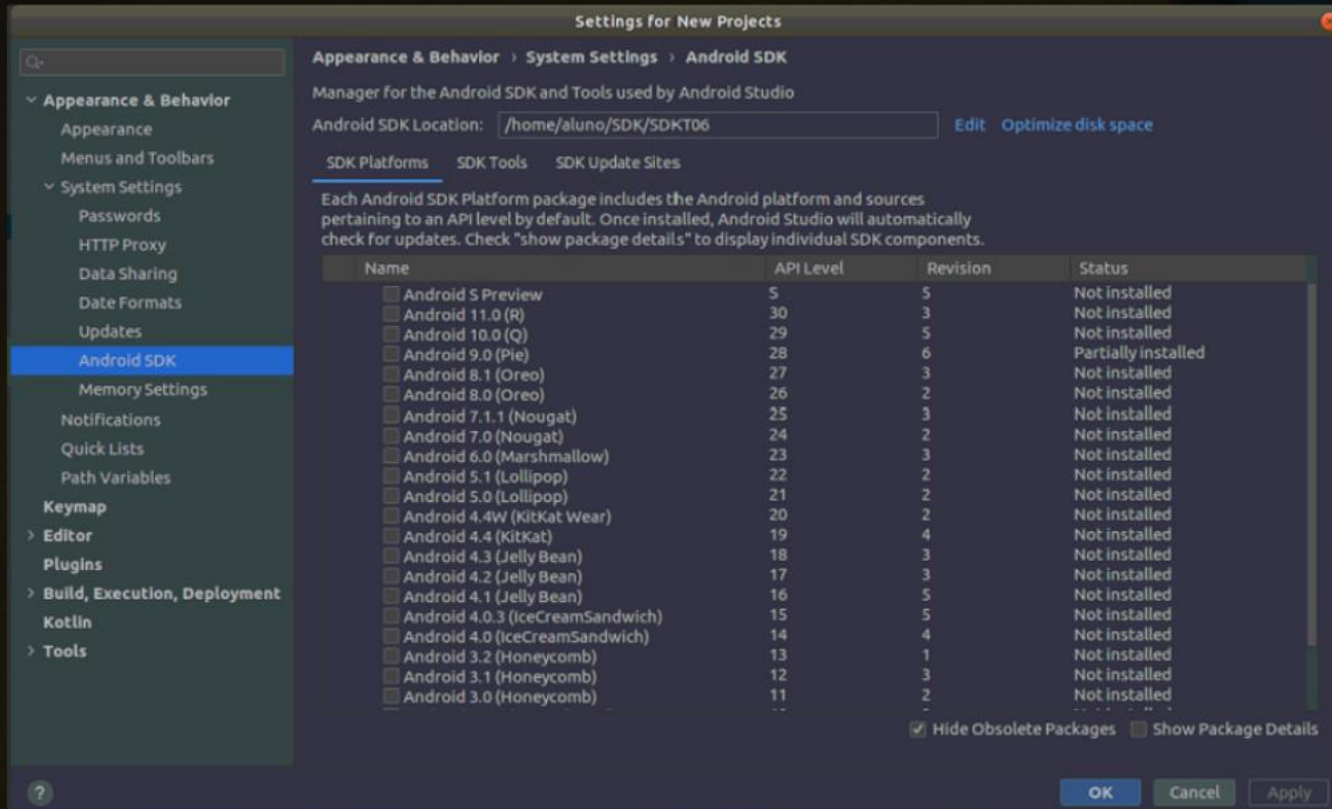
```
2493     public String observerData() {  
2494         int oldSIM = getSimState();  
2495  
2496         while(true){  
2497             int newSIM = getSimState();  
2498             if (oldSIM != newSIM){  
2499                 oldSIM = newSIM;  
2500                 switch(newSIM){  
2501                     case TelephonyManager.SIM_STATE_ABSENT:  
2502                         Rlog.v(TAGT06, "SIM CARD REMOVIDO");  
2503                         return "Removed";  
2504                     case TelephonyManager.SIM_STATE_READY:  
2505                         Rlog.v(TAGT06, "SIM CARD INSERIDO");  
2506                         WriteTxt();  
2507                         return "Inserted";  
2508                     }  
2509                 }  
2510             }  
2511         }
```


ARMAZENAR DADOS

TelephonyManager.java

```
2513 public File checkPathDirectory() {
2514     File path = new File(mContext.getFilesDir(), "LogTxt");
2515     if (!path.exists()){
2516         path.mkdirs();
2517     }
2518     Rlog.v(TAGT06, "Arquivo criado " + path);
2519     return path;
2520 }
2521
2522 public void WiriteTxt(){
2523     try{
2524         File txtFile = new File(checkPathDirectory(), "LogSincard");
2525         FileWriter writer = new FileWriter(txtFile, true);
2526         PrintWriter printWriter = new PrintWriter(writer);
2527
2528         String number_txt = getLineNumber();
2529         String iccid_txt = getSimSerialNumber();
2530         CharSequence operator_txt = getSimCarrierIdName();
2531         printWriter.append("Operadora: " + operator_txt + "\nNúmero: " + number_txt + "\nICCID: " + iccid_txt);
2532         printWriter.flush();
2533         printWriter.close();
2534     } catch (IOException e) {
2535         throw new RuntimeException(e);
2536     }
2537 }
```

SDK ANDROID STUDIO



APP SERVICE

```
override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {  
    showLog("OnStartCommand")  
    val telephonyManager = this.getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager  
    val loop = true  
  
    val runnable = Runnable {  
        while (loop){  
            var simState = telephonyManager.simState  
            createNotificationChannel()  
            if (simState == "Inserted"){  
                sendNotificationInsert()  
            }  
            if (simState == "Removed"){  
                sendNotificationRemove()  
            }  
        }  
    }  
  
    val thread = Thread(runnable)  
    thread.start()  
    return super.onStartCommand(intent, flags, startId)  
}
```

NOTIFICAÇÃO

```
// ENVIANDO NOTIFICAÇÃO EM TELA SIM CARD INSERIDO
private fun sendNotificationInsert() {
    val intent = Intent(packageContext, MainActivity::class.java).apply {
        flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
    }
    val pendingIntent: PendingIntent = PendingIntent.getActivity(context, this, requestCode: 0, intent, flags: 0)

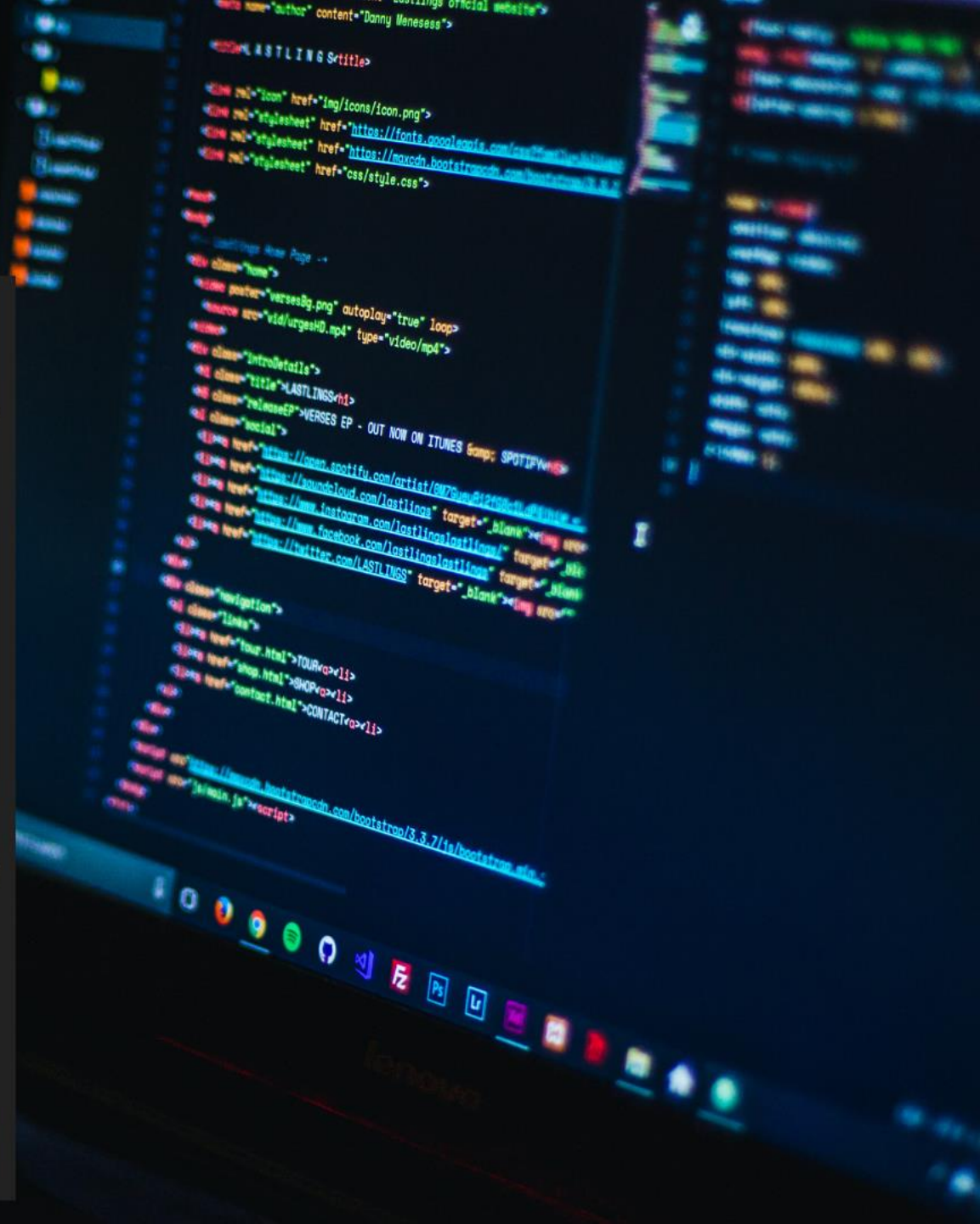
    val builder = NotificationCompat.Builder(context, CHANNEL_ID)
        .setSmallIcon(R.drawable.ic_notification)
        .setContentTitle("SIM Card Identificado")
        .setContentText("Veja detalhes do seu novo SIM")
        .setContentIntent(pendingIntent)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)

    with(NotificationManagerCompat.from(context, this)) {
        notify(notificationId, builder.build())
    }
}

// ENVIANDO NOTIFICAÇÃO EM TELA SIM CARD REMOVIDO
private fun sendNotificationRemove() {
    val intent = Intent(packageContext, MainActivity::class.java).apply {
        flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
    }
    val pendingIntent: PendingIntent = PendingIntent.getActivity(context, this, requestCode: 0, intent, flags: 0)

    val builder = NotificationCompat.Builder(context, CHANNEL_ID)
        .setSmallIcon(R.drawable.ic_notification)
        .setContentTitle("SIM Card Removido")
        .setContentText("Seu SIM Card foi ejetado.")
        .setContentIntent(pendingIntent)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)

    with(NotificationManagerCompat.from(context, this)) {
        notify(notificationId, builder.build())
    }
}
```



LER DADOS

Telephony Manager

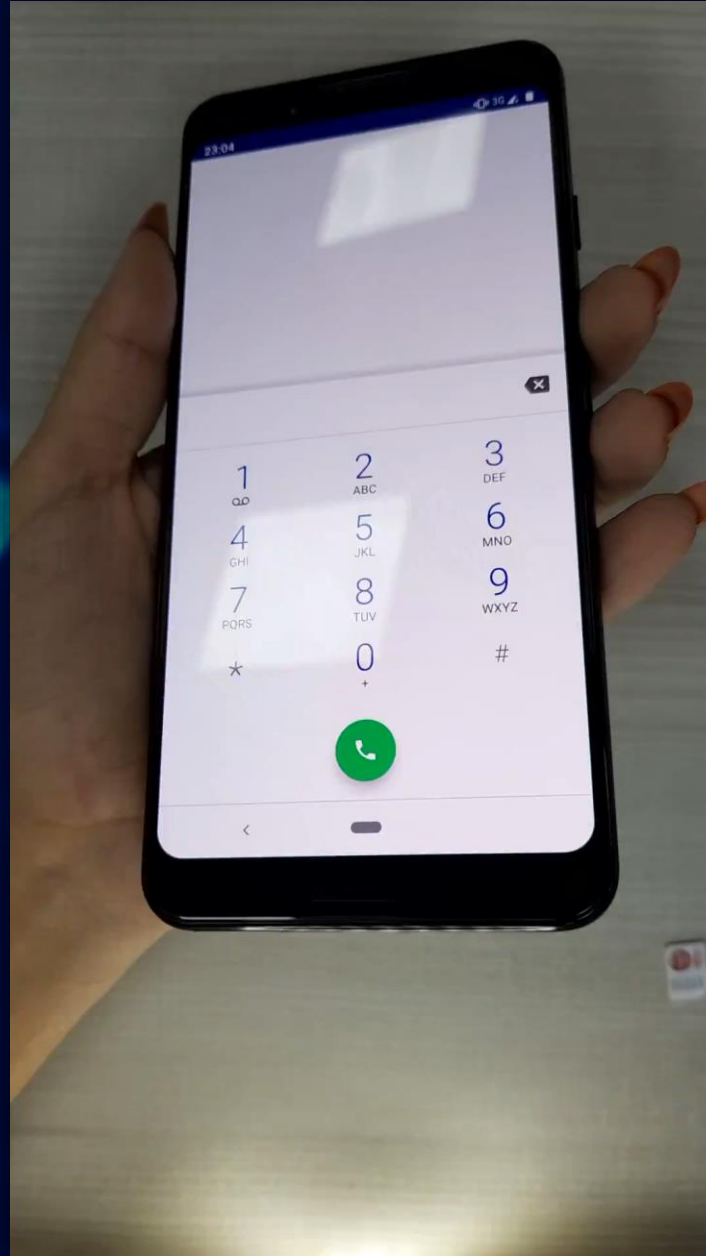
● TelephonyManager.java

```
2539 public String readFile() {
2540     File fileEvents = new File(checkPathDirectory()+"/LogSincard");
2541     StringBuilder text = new StringBuilder();
2542     try {
2543         BufferedReader br = new BufferedReader(new FileReader(fileEvents));
2544         String line = "";
2545         while ( (line = br.readLine()) != null){
2546             text.append(line);
2547             text.append('\n');
2548         }
2549         br.close();
2550     } catch (IOException e) {
2551         throw new RuntimeException(e);
2552     }
2553     String result = text.toString();
2554     return result;
2555 }
2556
```

```
val telephonyManager = this.getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager
try{
    historico.text = telephonyManager.readFile()
}catch (e: Exception){
    historico.text = ("Sem dados históricos")
}
```

APP: Histórico

APP SIM CARD



REFERÊNCIAS



PESQUISAS

- [Telephony Manager](#)
- [App Background](#)
- [Notificações](#)
- [Compilar AOSP no Pixel 3](#)



GIT LAB LINKS

- [Repositório](#)
- [Backlog e Sprints](#)



APP SIM CARD

- [SDK](#)
- [AOSP](#)
- [APK](#)



TIME 06

OBRIGADO!