Android软件设计课程实验

Android软件设计课程实验报告

学号 SA20225506

学号 SA20225483

学号 SA20225490

姓名 谢雪松

姓名 吴俊锋

姓名 伍开秋

日期 2020.10.28

|  |  |
| --- | --- |
| **实验名称** | 实验三：采集手机中安装的应用程序信息并进行持久化存储 |
| **实验内容** | 内容：  采集手机中安装的应用程序信息；将采集好的信息进行持久化存储  （要求采用四种数据库之一进行存储，请选择一种最合适的存储方式）  要求：  1. 至少要收集<APP Name（APP名称），FirstRunningTime（APP第一次启动时间），LastRunningTime（APP最后一次启动时间），TotalRunningTime（App总运行时间）>四种数据；  2. 采用合适的方式对采集的数据进行持久化存储（选择一种数据库）；  评分：  1. 要求的四种数据收集成功，并完成持久化存储（14分）；  2. 上述功能封装成Service后台运行（+2）；  3. 四种数据之外可以扩充收集的数据种类，种类越多分数越高  （前提信息具有完整性并且具有合理性，此加分项为最重要的一项！！）； |
| **实验完成情况**  （包括完成的实验内容及每个实验的完成程度。注意要贴出每个实验的  核心代码） | 1. 成功收集到了应用名称、应用第一次启动时间、应用最后一次启动时间、应用总使用时间、应用总使用次数、应用包名等信息。 1）获取应用包名：   public void onAccessibilityEvent(AccessibilityEvent event) {  …  String pkgname = event.getPackageName() == null ? "" : event.getPackageName().toString();  2）获取应用名：  String appLabel = getAppLabel(pkgname);  3）获取第一次使用时间:  在应用第一次打开时，获得系统时间截  System.currentTimeMillis(); 4）应用总使用时间： 累加每次使用时间，就可以得到总使用时间  app.addTotalRuningTime(spend);  5）应用总使用次数 每次使用应用，就让次数加1  app.addUseCount();  2、使用了基于Sqlite3数据库开源框架room对收集到的数据进行持久化存储。 @Entity(indices = {@Index(value = {"pkgName"},  unique = true)}) public class App {  @PrimaryKey (autoGenerate = true)  long appId;  *//应用名* String appName;  *//包名* String pkgName;  *//应用图标  //应用第一次运行时间* long firstRunningTime;  *//应用最后一次运行时间* long lastRuningTime;  *//应用总使用时间* long totalRuningTime;  *//使用次数* long useCount;  …  }  3、将收集服务集成到了一个AccessibilityMonitorService服务里，该服务继承自AccessibilityService，通过辅助功能来触发记录程序执行。  public class AccessibilityMonitorService extends AccessibilityService {  @Override public void onAccessibilityEvent(AccessibilityEvent event) {  myRecorder2.record(event); }  …  }   1. 还记录了在使用某个应用的启动时间、持续时间、使用前后电量、使用前的网络状态和充电状态。并把每次使用的记录存放在另一张表里。 oneUse.setPkgName(pkgname); oneUse.setAppName(appName); oneUse.setStartTimestamp(l); DBUtils.*storeBatteryInfo*(context, oneUse, false); DBUtils.*storeNetworkInfo*(context, oneUse); |
| **实验中的问题**  （包括在实验中遇到的问题，以及解决问题的方法） | 1. 在记录应用时，是否每个应用都要记录？ 首先，每次退出应用后都会回到系统桌面，再打开其他应用，因此桌面应用通常不会成为用户的目的应用，我们不给予记录，类似的一些类名以andoroid开头的系统应用，也不进行记录。 2. 如何使用AccessibilityService？ 在应用清单manifest.xml文件里注册服务   <service  android:name="com.rom471.service.AccessibilityMonitorService"  android:enabled="true"  android:exported="true"  android:permission="android.permission.BIND\_ACCESSIBILITY\_SERVICE">  <intent-filter>  <action android:name="android.accessibilityservice.AccessibilityService" />  </intent-filter>  <meta-data  android:name="android.accessibilityservice"  android:resource="@xml/accessibility" /> </service>  由于辅助功能不会默认打开，我们需要设计一个按钮，可以跳转到系统的辅助功能设置界面，让用户打开我们的辅助服务。   1. Room框架中，如何添加多张表？ 给多个JavaBean添加注解， @Entity public class OneUse {  @PrimaryKey(autoGenerate = true)  long id;  *//应用名* String appName;  *//包名* String pkgName; ……………. } 另一张表 @Entity(indices = {@Index(value = {"pkgName"},  unique = true)}) public class App {  @PrimaryKey (autoGenerate = true)  long appId;  *//应用名* String appName;  *//包名* String pkgName;  …………………….. } 然后在Database类中指定这些类 @Database(entities = {App.class,OneUse.class,Event.class},version = 1,exportSchema = false) public abstract class AppDataBase extends RoomDatabase {  ………………….. } 2. 如何获得电池、网络等信息？   BatteryManager manager = (BatteryManager)context.getSystemService(BATTERY\_SERVICE); 从系统服务获得BatteryManager，然后可以获得当前电量和是否在充电。 ConnectivityManager manager= (ConnectivityManager)context.getSystemService(Context.*CONNECTIVITY\_SERVICE*);  从系统获得ConnectivityManager，然后可以获得网络信息。  5、每个应用是怎么收集的  public void onAccessibilityEvent(AccessibilityEvent event) {  myRecorder2.record(event);  }  通过event可以获得应用的包名，进而获得应用名，同时记录当前的时间截、电池、网络等信息。 |

|  |  |
| --- | --- |
| **实验结果**  **及成果展示**  （包括实验完成后的源码和打包文件的说明） | 本应用由一个Activity组成，分为三个Fragment，分别展示查找界面、主界面和设置界面。这些界面可通过底部的导航按钮切换。由于引入了ViewPager，所以也可以通过左右滑动来切换不同的界面。  启动应用后，默认打开的是主界面，主界面由3个RecyclerView组成，每个都是横向布局，分别展示如下内容：   1. 最近使用的应用图标、名称和最近使用时间； 2. 应用图标、名称和总使用时间，按总使用时间排序； 3. 应用图标、名称和总使用次数，按总使用次数排序；     查找界面：  查找界面主要提供了对每次使用记录（OneUse）的查找功能，可以分别按名字查找、按使用日期查找和按时间查找应用。    其中每次使用记录包含了如下信息：每次应用在前台打开的时长、上次使用的时间、使用期间电量的变化、是否充电、使用的数据是移动网络还是WIFI。    按名字查找：在输入框输入应用名字可查询显示带关键字的应用使用记录。不区分大小写。    按日期查找：在这个fragment可以根据开始日期、结束日期查找符合日期要求的应用使用记录。开始查找的button旁显示查找到的应用使用记录数码      按时间查找：这个fragment可以根据开始时间和结束时间查找符合时间要求的应用使用记录。        第三个界面是设置界面：  这里包括三个button，第一个是清除数据，点击可以清除数据库内的应用使用数据。        本应用将收集服务集成到了一个AccessibilityMonitorService服务里，该服务继承自AccessibilityService，第二个button可以引导用户前往系统设置，打开应用预测的辅助功能，通过辅助功能来触发记录程序执行。            点击第三个button可以将数据库中当天的该时段的应用使用记录导出到手机本地。如下图为导出后的数据库文件。      打包文件说明  ─Test 工程目录  │ build.gradle  │ gradle.properties  │ gradlew  │ gradlew.bat  │ local.properties  │ settings.gradle  │  ├─.gradle  │  ├─.idea  │  ├─gradle  │  └─recorder 模块目录  │ build.gradle 模块编译配置，含有依赖信息  │ proguard-rules.pro  │  ├─libs  ├─release 模块打包发布目录  │ output-metadata.json  │ recorder-release.apk 打包并签名后的apk  │  └─src 源文件目录  └─main  │ AndroidManifest.xml 应用清单  │  ├─java  │ └─com  │ └─rom471  │ ├─adapter 一些列表的适配器  │ │ AppsAdapter.java  │ │ AppsLastUseAdapter.java  │ │ AppsTotalCountAdapter.java  │ │ AppsTotalTimeAdapter.java  │ │ MyFragmentPagerAdapter.java  │ │ OneUseAdapter.java  │ │  │ ├─db2 跟数据库操作有关的目录  │ │ App.java  │ │ AppDao.java  │ │ AppDataBase.java  │ │ AppRecordsRepository.java  │ │ Event.java  │ │ OneUse.java  │ │  │ ├─service 跟记录服务有关的目录  │ │ AccessibilityMonitorService.java 辅助功能服务  │ │ MyRecorder2.java 记录收集类  │ │  │ ├─ui  │ │ │ MainActivity.java 主界面  │ │ │  │ │ ├─fragments  │ │ │ HomeFragment2.java 主Fragment  │ │ │ RcordFragment.java 记录查找的Fragment  │ │ │ SettingsFragment.java 设置fragment  │ │ │  │ │ └─fragments2 查找Fragment里嵌套的Fragment  │ │ FindByDateFragment.java 按日期查找  │ │ FindByNameFragment.java 按名字查找  │ │ FindByTimeFragment.java 按时间查找  │ │ OneUseFindFragment.java 查找类的基类  │ │  │ └─utils 一些用到的工具类  │ DBUtils.java 主要跟数据存取相关  │  │  └─res  ├─drawable 图片和背景选择器  │ home.png  │ home\_fill.png  │ home\_selector.xml  │ ic\_launcher\_background.xml  │ list.png  │ list\_fill.png  │ list\_selector.xml  │ radio\_button\_selector.xml  │ settings.png  │ settings\_fill.png  │ settings\_selector.xml  │ userlist.png  │ userlist\_fill.png  │  ├─layout 布局文件目录  │ app\_list\_layout.xml  │ fragment\_record\_buttom.xml  │ main.xml  │ main\_buttom.xml  │ main\_fragment\_home.xml  │ main\_fragment\_record.xml  │ main\_fragment\_record\_find\_by\_date.xml  │ main\_fragment\_record\_find\_by\_name.xml  │ main\_fragment\_record\_find\_by\_time.xml  │ main\_fragment\_settings.xml  │ oneuse\_list\_item.xml  │ record\_list\_item.xml  │ record\_list\_layout.xml  │ recycle\_app\_item.xml  │  ├─mipmap-hdpi 启动图标目录  │ ic\_launcher.png  │ ic\_launcher\_round.png  │  ├─values 一些常量值目录  │ colors.xml  │ strings.xml  │ styles.xml  │ themes.xml  │  └─xml  accessibility.xml 辅助功能配置文件  实验完成后的源码  1、主界面MainActivity.java  public class MainActivity extends AppCompatActivity implements RadioGroup.OnCheckedChangeListener, ViewPager.OnPageChangeListener {  public static final String *TAG* = "cedar";  RadioGroup mRadioGroup;  RadioButton rb1, rb2, rb3;  ViewPager vp;  private MyFragmentPagerAdapter mAdapter;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*main*);  mAdapter = new MyFragmentPagerAdapter(getSupportFragmentManager(), FragmentPagerAdapter.*BEHAVIOR\_RESUME\_ONLY\_CURRENT\_FRAGMENT*);  bindView();*//绑定资源* vp.setAdapter(mAdapter);  vp.setCurrentItem(1);  vp.addOnPageChangeListener(this);  mRadioGroup.setOnCheckedChangeListener(this);  }  private void bindView() {  mRadioGroup = findViewById(R.id.*radioGroup1*);  rb1 = findViewById(R.id.*radio1*);  rb2 = findViewById(R.id.*radio2*);  rb3 = findViewById(R.id.*radio3*);  vp = findViewById(R.id.*vpager*);  }  @Override  public void onCheckedChanged(RadioGroup group, int checkedId) {  switch (checkedId) {  case R.id.*radio1*:  vp.setCurrentItem(0);  break;  case R.id.*radio2*:  vp.setCurrentItem(1);  break;  case R.id.*radio3*:  vp.setCurrentItem(2);  break;  }  }  @Override  public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {  }  @Override  public void onPageSelected(int position) {  }  @Override  public void onPageScrollStateChanged(int state) {  if (state == 2) {  switch (vp.getCurrentItem()) {  case 0:  rb1.setChecked(true);  break;  case 1:  rb2.setChecked(true);  break;  case 2:  rb3.setChecked(true);  break;  }  }  } }  HomeFragment2.java  public class HomeFragment2 extends Fragment {  RecyclerView last\_list\_view;  RecyclerView total\_list\_view;  RecyclerView count\_list\_view;  Context context;  AppRecordsRepository appRecordsRepository;  AppsLastUseAdapter lastUseAdapter;  AppsTotalTimeAdapter totalTimeAdapter;  AppsTotalCountAdapter totalCountAdapter;  @Nullable  @Override  public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  return inflater.inflate(R.layout.*main\_fragment\_home*, container, false);  }  @RequiresApi(api = Build.VERSION\_CODES.*M*)  @Override  public void onActivityCreated(@Nullable Bundle savedInstanceState) {  super.onActivityCreated(savedInstanceState);  context = getContext();  last\_list\_view = getActivity().findViewById(R.id.*app\_last\_use\_list*);  total\_list\_view = getActivity().findViewById(R.id.*app\_total\_use\_list*);  count\_list\_view = getActivity().findViewById(R.id.*app\_total\_count\_list*);  appRecordsRepository = new AppRecordsRepository(getActivity().getApplication());  totalTimeAdapter = new AppsTotalTimeAdapter(this, appRecordsRepository);  lastUseAdapter = new AppsLastUseAdapter(this, appRecordsRepository);  totalCountAdapter = new AppsTotalCountAdapter(this, appRecordsRepository);  LinearLayoutManager layoutManager1 = new LinearLayoutManager(context);  layoutManager1.setOrientation(LinearLayoutManager.*HORIZONTAL*);  last\_list\_view.setLayoutManager(layoutManager1);  last\_list\_view.setAdapter(lastUseAdapter);  LinearLayoutManager layoutManager = new LinearLayoutManager(context);  layoutManager.setOrientation(LinearLayoutManager.*HORIZONTAL*);  total\_list\_view.setLayoutManager(layoutManager);  total\_list\_view.setAdapter(totalTimeAdapter);  LinearLayoutManager layoutManager3 = new LinearLayoutManager(context);  layoutManager3.setOrientation(LinearLayoutManager.*HORIZONTAL*);  count\_list\_view.setLayoutManager(layoutManager3);  count\_list\_view.setAdapter(totalCountAdapter);  }  @Override  public void onResume() {  super.onResume();  } }  SettingsFragment.java  public class SettingsFragment extends Fragment implements View.OnClickListener {  *//先定义* private static final int *REQUEST\_EXTERNAL\_STORAGE* = 1;  private static final String[] *PERMISSIONS\_STORAGE* = {  "android.permission.READ\_EXTERNAL\_STORAGE",  "android.permission.WRITE\_EXTERNAL\_STORAGE"};  Button accessibility\_btn;  Button clearRecord\_btn;  Button normal\_service\_btn;  Button ouput\_db\_btn;  AppRecordsRepository appRecordsRepository;  Context context;  private static boolean isAccessibilitySettingsOn(Context context) {  String service = "com.rom471.recorder2/com.rom471.service.AccessibilityMonitorService";  TextUtils.SimpleStringSplitter mStringColonSplitter = new TextUtils.SimpleStringSplitter(':');  String settingValue = Settings.Secure.*getString*(  context.getApplicationContext().getContentResolver(),  Settings.Secure.*ENABLED\_ACCESSIBILITY\_SERVICES*);  Log.*d*("cedar", "isAccessibilitySettingsOn: " + settingValue);  if (settingValue != null) {  mStringColonSplitter.setString(settingValue); *//各个服务由分号分割* while (mStringColonSplitter.hasNext()) {  String accessibilityService = mStringColonSplitter.next();  if (accessibilityService.equalsIgnoreCase(service)) {  return true;  }  }  }  return false;  }  *//然后通过一个函数来申请* public static boolean verifyStoragePermissions(Activity activity) {  try {  *//检测是否有写的权限* int permission = ActivityCompat.*checkSelfPermission*(activity,  "android.permission.WRITE\_EXTERNAL\_STORAGE");  if (permission != PackageManager.*PERMISSION\_GRANTED*) {  *// 没有写的权限，去申请写的权限，会弹出对话框* ActivityCompat.*requestPermissions*(activity, *PERMISSIONS\_STORAGE*, *REQUEST\_EXTERNAL\_STORAGE*);  }  return true;  } catch (Exception e) {  e.printStackTrace();  return false;  }  }  @Nullable  @Override  public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  return inflater.inflate(R.layout.*main\_fragment\_settings*, container, false);  }  @RequiresApi(api = Build.VERSION\_CODES.*M*)  @Override  public void onActivityCreated(@Nullable Bundle savedInstanceState) {  super.onActivityCreated(savedInstanceState);  context = getContext();  appRecordsRepository = new AppRecordsRepository(getActivity().getApplication());  accessibility\_btn = getActivity().findViewById(R.id.*open\_accessibility\_btn*);  normal\_service\_btn = getActivity().findViewById(R.id.*start\_service\_btn*);  accessibility\_btn.setOnClickListener(this);  clearRecord\_btn = getActivity().findViewById(R.id.*clear\_records\_btn*);  ouput\_db\_btn = getActivity().findViewById(R.id.*output\_db\_btn*);  ouput\_db\_btn.setOnClickListener(this);  clearRecord\_btn.setOnClickListener(this);  normal\_service\_btn.setOnClickListener(this);  }  @RequiresApi(api = Build.VERSION\_CODES.*N*)  @Override  public void onClick(View v) {  switch (v.getId()) {  case R.id.*clear\_records\_btn*:  confirmClearRecordsDialog();  break;  case R.id.*open\_accessibility\_btn*:  getAccessibilityPermission(getContext());  break;  case R.id.*output\_db\_btn*:  if (*verifyStoragePermissions*(getActivity())) {  export\_db();  } else {  }  break;  }  }  @Override  public void onResume() {  super.onResume();  boolean accessibilitySettingsOn = *isAccessibilitySettingsOn*(context);  if (accessibilitySettingsOn) {  accessibility\_btn.setText("辅助功能已经打开");  } else {  accessibility\_btn.setText("点击打开辅助功能");  }  }  @RequiresApi(api = Build.VERSION\_CODES.*N*)  public void export\_db() {  File sd = Environment.*getExternalStorageDirectory*();  File data = Environment.*getDataDirectory*();  FileChannel source = null;  FileChannel destination = null;  String currentDBPath = "/data/" + "com.rom471.recorder2" + "/databases/" + "apps.db";  String backupPath = sd + "/rom471/";  File backupPathFile = new File(backupPath);  if (!backupPathFile.exists()) { *//检查目录是否存在* try {  *//按照指定的路径创建文件夹* backupPathFile.mkdirs();  } catch (Exception e) {  }  }  String dbname = DBUtils.*getCurrentDBString*();  String backupDBName = dbname + ".db";  File currentDB = new File(data, currentDBPath);  File backupDB = new File(backupPath, backupDBName);  try {  source = new FileInputStream(currentDB).getChannel();  destination = new FileOutputStream(backupDB).getChannel();  destination.transferFrom(source, 0, source.size());  source.close();  destination.close();  Toast.*makeText*(context, "DB Exported!", Toast.*LENGTH\_LONG*).show();  } catch (IOException e) {  e.printStackTrace();  }  }  *//确认清除数据的弹出对话框* public void confirmClearRecordsDialog() {  final AlertDialog.Builder normalDialog =  new AlertDialog.Builder(context);  normalDialog.setIcon(R.drawable.*ic\_launcher\_background*);  normalDialog.setTitle("警告！");  normalDialog.setMessage("确认要删除已有的数据吗？");  normalDialog.setPositiveButton("确定",  new DialogInterface.OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {  appRecordsRepository.deleteAll();  }  });  normalDialog.setNegativeButton("关闭",  new DialogInterface.OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {  *//...To-do* }  });  normalDialog.show();  }  *//跳转到打开辅助功能界面* public void getAccessibilityPermission(Context context) {  final String mAction = Settings.*ACTION\_ACCESSIBILITY\_SETTINGS*;*//辅助功能* Intent intent = new Intent(mAction);  context.startActivity(intent);  }  public void toast(Context context, String text) {  Toast toast = Toast.*makeText*(context, text, Toast.*LENGTH\_SHORT*);  toast.setGravity(Gravity.*CENTER*, 0, 0);  toast.show();  } } |
|  | RcordFragment.java  public class RcordFragment extends Fragment implements RadioGroup.OnCheckedChangeListener {  RadioGroup mRadioGroup;  RadioButton rb1, rb2, rb3;  Context context;  private final List<Fragment> fragments = new ArrayList<>();  private Fragment currentfragment;  private FragmentManager fm;  @Nullable  @Override  public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  return inflater.inflate(R.layout.*main\_fragment\_record*, container, false);  }  public void onActivityCreated(@Nullable Bundle savedInstanceState) {  super.onActivityCreated(savedInstanceState);  context = getActivity();  bindView();  initFragments();  mRadioGroup.setOnCheckedChangeListener(this);  showFragmnet(fragments.get(0));  rb1.setChecked(true);  }  @Override  public void onStart() {  super.onStart();  }  private void initFragments() {  fm = getChildFragmentManager();  fragments.add(new FindByNameFragment());  fragments.add(new FindByDateFragment());  fragments.add(new FindByTimeFragment());  }  private void showFragmnet(Fragment fragment) {  FragmentTransaction transaction = fm.beginTransaction();  if (currentfragment == null) {*// 当还没有fragment时* currentfragment = fragment;  transaction.add(R.id.*record\_fragment*, currentfragment).show(currentfragment).commit();  return;  }  if (currentfragment != fragment) {  transaction.hide(currentfragment);  currentfragment = fragment;  if (!fragment.isAdded()) {  transaction.add(R.id.*record\_fragment*, fragment).show(fragment).commit();  } else {  transaction.show(fragment).commit();  }  }  }  private void bindView() {  mRadioGroup = getActivity().findViewById(R.id.*radioGroup1*);  rb1 = getActivity().findViewById(R.id.*record\_findbyname*);  rb2 = getActivity().findViewById(R.id.*radio2*);  rb3 = getActivity().findViewById(R.id.*radio3*);  }  @Override  public void onCheckedChanged(RadioGroup group, int checkedId) {  switch (checkedId) {  case R.id.*record\_findbyname*:  showFragmnet(fragments.get(0));  break;  case R.id.*record\_find\_by\_date*:  showFragmnet(fragments.get(1));  break;  case R.id.*record\_find\_by\_time*:  showFragmnet(fragments.get(2));  break;  }  } } |
|  | FindByDateFragment.java  public class FindByDateFragment extends OneUseFindFragment {  Button start\_date\_btn;  Button end\_date\_btn;  Button start\_btn;  TextView start\_date\_tv;  TextView end\_date\_tv;  TextView result\_tv;  long start\_timestamp;  long end\_timestamp;  public FindByDateFragment() {  super(R.layout.*main\_fragment\_record\_find\_by\_date*, R.id.*record\_list\_by\_date*);  }  private void registRecords() {  AppRecordsRepository appRecordsRepository;  appRecordsRepository = new AppRecordsRepository(getActivity().getApplication());  appRecordsRepository.getAllOneUses().observe(this, new Observer<List<OneUse>>() {  @Override  public void onChanged(List<OneUse> records) {  DBUtils.*setOneUseIcon*(context, records);  mAdapter.setOneUses(records);  list\_view.setAdapter(mAdapter);  }  });  }  public void bindView() {  list\_view = getActivity().findViewById(R.id.*record\_list\_by\_date*);  start\_date\_btn = getActivity().findViewById(R.id.*record\_date\_start\_btn*);  end\_date\_btn = getActivity().findViewById(R.id.*record\_date\_end\_btn*);  start\_btn = getActivity().findViewById(R.id.*date\_start\_search*);  start\_btn.setOnClickListener(this);  start\_date\_btn.setOnClickListener(this);  end\_date\_btn.setOnClickListener(this);  start\_date\_tv = getActivity().findViewById(R.id.*record\_date\_start\_tv*);  end\_date\_tv = getActivity().findViewById(R.id.*record\_date\_end\_tv*);  result\_tv = getActivity().findViewById(R.id.*date\_search\_result*);  }  *//过滤日期* private List<OneUse> filterByDate(List<OneUse> old\_list, long start, long end) {  if (start\_timestamp == 0 && end\_timestamp == 0) *//未指定时间则返回所有记录* return old\_list;  List<OneUse> new\_list = new ArrayList<>();  for (OneUse r : old\_list  ) {  long timestamp = r.getStartTimestamp();  if (start\_timestamp != 0 && timestamp < start)  continue;  if (end\_timestamp != 0 && timestamp > end)  continue;  new\_list.add(r);  }  return new\_list;  }  @RequiresApi(api = Build.VERSION\_CODES.*N*)  @Override  public void onClick(View v) {  switch (v.getId()) {  case R.id.*record\_date\_start\_btn*:  showDatePickerDialog(context, true);  break;  case R.id.*record\_date\_end\_btn*:  showDatePickerDialog(context, false);  break;  case R.id.*date\_start\_search*:  List<OneUse> records = filterByDate(mOneUses, start\_timestamp, end\_timestamp);  result\_tv.setText("查到记录：" + records.size() + "条");  mAdapter.setOneUses(records);  list\_view.setAdapter(mAdapter);  break;  }  }  @RequiresApi(api = Build.VERSION\_CODES.*N*)  public void showDatePickerDialog(Context activity, boolean start) {  Calendar calendar = Calendar.*getInstance*();  DatePickerDialog datePickerDialog = new DatePickerDialog(activity,  new DatePickerDialog.OnDateSetListener() {  @Override  public void onDateSet(DatePicker view, int year,  int monthOfYear, int dayOfMonth) {  *// 此处得到选择的时间，可以进行你想要的操作* long timestamp = new GregorianCalendar(year, monthOfYear, dayOfMonth).getTimeInMillis();  if (start) {  start\_timestamp = timestamp;  start\_date\_tv.setText("" + year + "年" + (monthOfYear + 1) + "月" + dayOfMonth + "日");  } else {  end\_timestamp = timestamp;  end\_date\_tv.setText("" + year + "年" + (monthOfYear + 1) + "月" + dayOfMonth + "日");  }  }  }, calendar.get(Calendar.*YEAR*),  calendar.get(Calendar.*MONTH*),  calendar.get(Calendar.*DAY\_OF\_MONTH*));  datePickerDialog.show();  } } |
|  | FindByNameFragment.java  public class FindByNameFragment extends OneUseFindFragment {  Button record\_search\_btn;  EditText record\_search\_et;  TextView record\_result\_tv;  public FindByNameFragment() {  super(R.layout.*main\_fragment\_record\_find\_by\_name*, R.id.*record\_list\_by\_name*);  }  @Override  void bindView() {  record\_search\_btn = getActivity().findViewById(R.id.*record\_search\_btn*);  record\_search\_et = getActivity().findViewById(R.id.*record\_search\_et*);  record\_result\_tv = getActivity().findViewById(R.id.*record\_serch\_result*);  record\_search\_btn.setOnClickListener(this);  }  @Override  public void onClick(View v) {  switch (v.getId()) {  case R.id.*record\_search\_btn*:  String appname = record\_search\_et.getText().toString();  List<OneUse> records = filterByName(appname, mOneUses);  updateWithFoundRecords(records);  record\_result\_tv.setText("查到记录：" + records.size() + "条");  break;  }  }  private List<OneUse> filterByName(String appname, List<OneUse> old\_list) {  List<OneUse> new\_list = new ArrayList<>();  for (OneUse r : old\_list  ) {  if (r.getAppName().toLowerCase().contains(appname.toLowerCase())) {  new\_list.add(r);  }  }  return new\_list;  } } |
|  | FindByTimeFragment.java  public class FindByTimeFragment extends OneUseFindFragment {  Button start\_time\_btn;  Button end\_time\_btn;  Button start\_btn;  TextView start\_time\_tv;  TextView end\_time\_tv;  TextView result\_tv;  long start\_timestamp;  long end\_timestamp;  Context context;  public FindByTimeFragment() {  super(R.layout.*main\_fragment\_record\_find\_by\_time*, R.id.*record\_list\_by\_time*);  }  public void bindView() {  list\_view = getActivity().findViewById(R.id.*record\_list\_by\_time*);  start\_time\_btn = getActivity().findViewById(R.id.*record\_time\_start\_btn*);  end\_time\_btn = getActivity().findViewById(R.id.*record\_time\_end\_btn*);  start\_btn = getActivity().findViewById(R.id.*time\_start\_search*);  start\_time\_tv = getActivity().findViewById(R.id.*record\_time\_start\_tv*);  end\_time\_tv = getActivity().findViewById(R.id.*record\_time\_end\_tv*);  result\_tv = getActivity().findViewById(R.id.*time\_search\_result*);  start\_btn.setOnClickListener(this);  start\_time\_btn.setOnClickListener(this);  end\_time\_btn.setOnClickListener(this);  context = getContext();  }  *//过滤时间* private List<OneUse> filterByTime(List<OneUse> old\_list, long start, long end) {  Calendar cal = Calendar.*getInstance*(Locale.*getDefault*());  if (start\_timestamp == 0 && end\_timestamp == 0) *//未指定时间则返回所有记录* return old\_list;  List<OneUse> new\_list = new ArrayList<>();  for (OneUse r : old\_list  ) {  long timestamp = r.getStartTimestamp();  cal.setTimeInMillis(timestamp);  int hours = cal.get(Calendar.*HOUR\_OF\_DAY*);  int minutes = cal.get(Calendar.*MINUTE*);  *//int ms\_in\_day=(hours\*60+minutes)\*60\*1000;* int ms\_in\_day = cal.get(Calendar.*MILLISECONDS\_IN\_DAY*);  if (start != 0 && ms\_in\_day < start)  continue;  if (end != 0 && ms\_in\_day > end)  continue;  new\_list.add(r);  }  return new\_list;  }  @RequiresApi(api = Build.VERSION\_CODES.*N*)  @Override  public void onClick(View v) {  Log.*d*("TAG", "onClick:按下了 ");  switch (v.getId()) {  case R.id.*record\_time\_start\_btn*:  showTimePickerDialog(context, true);  break;  case R.id.*record\_time\_end\_btn*:  showTimePickerDialog(context, false);  break;  case R.id.*time\_start\_search*:  List<OneUse> records = filterByTime(mOneUses, start\_timestamp, end\_timestamp);  result\_tv.setText("查到记录：" + records.size() + "条");  mAdapter.setOneUses(records);  list\_view.setAdapter(mAdapter);  break;  }  }  @RequiresApi(api = Build.VERSION\_CODES.*N*)  private void showTimePickerDialog(Context context, boolean start) {  Calendar cal = Calendar.*getInstance*();  Dialog timeDialog = new TimePickerDialog(context, new TimePickerDialog.OnTimeSetListener() {  @Override  public void onTimeSet(TimePicker view, int hourOfDay, int minute) {  SimpleDateFormat format = new SimpleDateFormat(" HH:mm ");  *// TODO Auto-generated method stub* cal.set(Calendar.*HOUR\_OF\_DAY*, hourOfDay);  cal.set(Calendar.*MINUTE*, minute);  long timestamp = cal.get(Calendar.*MILLISECONDS\_IN\_DAY*);  if (start) {  start\_time\_tv.setText(format.format(cal.getTime()));  start\_timestamp = timestamp;  } else {  end\_time\_tv.setText(format.format(cal.getTime()));  end\_timestamp = timestamp;  }  }  }, cal.get(Calendar.*HOUR\_OF\_DAY*), cal.get(Calendar.*MINUTE*), true);  timeDialog.setTitle("请选择时间");  timeDialog.show();  } } |
|  | OneUseFindFragment.java  public abstract class OneUseFindFragment extends Fragment implements View.OnClickListener {  RecyclerView list\_view;  *//RecordDBHelper db;* int laytoutResource;  int listviewResource;  AppRecordsRepository appOneUsesRepository;  OneUseAdapter mAdapter;  List<OneUse> mOneUses;  Context context;  public OneUseFindFragment(@LayoutRes int layoutResource, @IdRes int listviewResouce) {  this.laytoutResource = layoutResource;  this.listviewResource = listviewResouce;  }  public OneUseFindFragment() {  }  public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  return inflater.inflate(laytoutResource, container, false);  }  public void onActivityCreated(@Nullable Bundle savedInstanceState) {  super.onActivityCreated(savedInstanceState);  bindView();  context = getContext();  list\_view = getActivity().findViewById(listviewResource);  registRecords();  mAdapter = new OneUseAdapter(mOneUses);  LinearLayoutManager layoutManager = new LinearLayoutManager(context);  layoutManager.setOrientation(LinearLayoutManager.*VERTICAL*);  list\_view.setLayoutManager(layoutManager);  list\_view.setAdapter(mAdapter);  *//设置动画* DefaultItemAnimator defaultItemAnimator = new DefaultItemAnimator();  defaultItemAnimator.setRemoveDuration(600);  list\_view.setItemAnimator(defaultItemAnimator);  registerForContextMenu(list\_view);  }  abstract void bindView();  private void registRecords() {  appOneUsesRepository = new AppRecordsRepository(getActivity().getApplication());  appOneUsesRepository.getAllOneUses().observe(this, new Observer<List<OneUse>>() {  @Override  public void onChanged(List<OneUse> records) {  mOneUses = records;  DBUtils.*setOneUseIcon*(context, mOneUses);  mAdapter.setOneUses(mOneUses);  *//list\_view.setAdapter(mAdapter);* mAdapter.notifyDataSetChanged();  }  });  }  void updateWithFoundRecords(List<OneUse> records) {  DBUtils.*setOneUseIcon*(context, records);  mAdapter.setOneUses(records);  list\_view.setAdapter(mAdapter);  }  @Override  public abstract void onClick(View v);  @Override  public boolean onContextItemSelected(@NonNull MenuItem item) {  int position = mAdapter.getPosition();  OneUse record = mOneUses.get(position);  switch (item.getItemId()) {  case 0:  appOneUsesRepository.delete(record);  Toast.*makeText*(context, "记录已经删除", Toast.*LENGTH\_SHORT*).show();  break;  }  return super.onContextItemSelected(item);  } } |
|  | AccessibilityMonitorService.java  public class AccessibilityMonitorService extends AccessibilityService {  MyRecorder2 myRecorder2;  public AccessibilityMonitorService() {  }  @Override  protected void onServiceConnected() {  super.onServiceConnected();  myRecorder2 = new MyRecorder2(getApplication());  }  @Override  public void onAccessibilityEvent(AccessibilityEvent event) {  myRecorder2.record(event);  }  @Override  public void onInterrupt() {  myRecorder2.close();  } *///将会迁移的函数* } |
|  | public class MyRecorder2 {  private static final String *TAG* = "cedar";  AppDao appDao;  *// RecordDAO dao;* App app;  OneUse oneUse;  Context context;  PackageManager pm;  List<String> stopApps;  boolean filter\_exclude = false;  boolean filter\_skip = false;  boolean app\_first = false;*//当前app是否是第一次插入* boolean record\_events = false;  AppDataBase appDB;  private String lastPkgName = "";  public MyRecorder2(Context context) {  this.context = context;  appDB = AppDataBase.*getInstance*(context);  appDao = appDB.getAppDao();  pm = context.getPackageManager();*//初始化包管理器* stopApps = getSkip\_names();  filter\_exclude = true;*//过滤开关* filter\_skip = true;*//过滤开关* record\_events = true;  }  public List<String> getSkip\_names() {  List<String> apps = new ArrayList<>();  apps.add("系统 UI");  apps.add("应用预测");  apps.add("系统桌面");  apps.add("设置");  apps.add("权限管理服务");  apps.add(("万象息屏"));  return apps;  }  *//获得应用名称* private String getAppLabel(String pkgname) {  String appLabel = "";  try {  appLabel = pm.getPackageInfo(pkgname, PackageManager.*GET\_ACTIVITIES*).applicationInfo.loadLabel(pm).toString();  } catch (PackageManager.NameNotFoundException e) {  e.printStackTrace();  }  return appLabel;  }  *//开始记录* private void record\_start(String pkgname) {  String appname = getAppLabel(pkgname);  long l = System.*currentTimeMillis*();  app\_first = false;  app = appDao.getAppByName(appname);  if (app == null) {  app = new App();  app.setFirstRunningTime(l);  app.setPkgName(pkgname);  app.setAppName(appname);  app\_first = true;  }  oneUse = new OneUse();  oneUse.setPkgName(pkgname);  oneUse.setAppName(appname);  oneUse.setStartTimestamp(l);  DBUtils.*storeBatteryInfo*(context, oneUse, true);  }  *//结束记录* private void record\_finish(String lastPkgName) {  if (oneUse == null)  return;  if (!oneUse.getPkgName().equals(lastPkgName)) {  return;  }  long l = System.*currentTimeMillis*();  long spend = l - oneUse.getStartTimestamp();  *//不记录0.5秒内的* if (spend < 500) return;  oneUse.setSpendTime(spend);  app.setLastRuningTime(l);  app.addTotalRuningTime(spend);*//增加统计表的时间* app.addUseCount();*//增加使用次数* DBUtils.*storeBatteryInfo*(context, oneUse, false);  DBUtils.*storeNetworkInfo*(context, oneUse);  if (app\_first)  appDao.insert(app);  else  appDao.updateApp(app);  appDao.insert(oneUse);  oneUse = null;  }  public void record(AccessibilityEvent event) {  int type = event.getEventType();  if (type != 32) return;  String pkgname = event.getPackageName() == null ? "" : event.getPackageName().toString();  String appName = getAppLabel(pkgname);  *//获取类名* String name = event.getClassName() == null ? "" : event.getClassName().toString();  if (!name.startsWith("android.")) {  Log.*d*(*TAG*, "" + appName + " " + name);  *//如果回到桌面就停止记录* if (stopApps.contains(appName)) {  record\_finish(lastPkgName);  }  *//不是桌面程序就开始记录* else {  record\_start(pkgname);  lastPkgName = pkgname;  }  }  }  public void close() {  appDB.close();  } } |
|  | @Entity(indices = {@Index(value = {"pkgName"},  unique = true)}) public class App {  @PrimaryKey(autoGenerate = true)  long appId;  *//应用名* String appName;  *//包名* String pkgName;  *//应用图标  //应用第一次运行时间* long firstRunningTime;  *//应用最后一次运行时间* long lastRuningTime;  *//应用总使用时间* long totalRuningTime;  *//使用次数* long useCount;  @Ignore  Drawable icon;  public App() {  }  public long getUseCount() {  return useCount;  }  public void setUseCount(long useCount) {  this.useCount = useCount;  }  public void addUseCount() {  this.useCount++;  }  public long getFirstRunningTime() {  return firstRunningTime;  }  public void setFirstRunningTime(long firstRunningTime) {  this.firstRunningTime = firstRunningTime;  }  public long getLastRuningTime() {  return lastRuningTime;  }  public void setLastRuningTime(long lastRuningTime) {  this.lastRuningTime = lastRuningTime;  }  public long getTotalRuningTime() {  return totalRuningTime;  }  public void setTotalRuningTime(long totalRuningTime) {  this.totalRuningTime = totalRuningTime;  }  public void addTotalRuningTime(long timespend) {  this.totalRuningTime += timespend;  }  public long getAppId() {  return appId;  }  public void setAppId(long appId) {  this.appId = appId;  }  public String getAppName() {  return appName;  }  public void setAppName(String appName) {  this.appName = appName;  }  public String getPkgName() {  return pkgName;  }  public void setPkgName(String pkgName) {  this.pkgName = pkgName;  }  public Drawable getIcon() {  return icon;  }  public void setIcon(Drawable icon) {  this.icon = icon;  }  @Override  public String toString() {  return "App{" +  "appName='" + appName + '\'' +  ", lastRuningTime=" + lastRuningTime +  ", totalRuningTime=" + totalRuningTime +  '}';  } } |
|  | @Dao public abstract class AppDao {  @Insert  abstract void insertOneUse(OneUse... oneUses);  @Insert  abstract void insertAll(List<OneUse> records);  @Query("select \* from App order by lastRuningTime desc limit :limit")  public abstract LiveData<List<App>> getLastUsedApp(int limit);  @Query("select \* from App order by totalRuningTime desc limit :limit")  public abstract LiveData<List<App>> getMostUsedApps(int limit);  @Query("select \* from App order by useCount desc limit :limit")  public abstract LiveData<List<App>> getMostCountsApps(int limit);  @Update  public abstract void updateApp(App app);  @Insert(onConflict = OnConflictStrategy.*IGNORE*)  public abstract long insertApp(App app);  @Query("select \* from App where appName=:appname limit 1")  public abstract App getAppByName(String appname);  @Query("select \* from OneUse order by id desc")  public abstract LiveData<List<OneUse>> getAllOneUse();  @Insert  public abstract void insert(App app);  @Insert  public abstract void insert(OneUse oneUse);  @Insert  public abstract void insert(Event app);  @Query("select \* from Event order by id desc")  public abstract LiveData<List<Event>> getAllEvents();  @Delete  public abstract void delete(OneUse oneUse);  @Query("delete from App")  public abstract void deleteApps();  @Query("delete from OneUse")  public abstract void deleteOneUses(); } |
|  | @Database(entities = {App.class, OneUse.class, Event.class}, version = 1, exportSchema = false) public abstract class AppDataBase extends RoomDatabase {  private static final Object *Lock* = new Object();  private static AppDataBase *INSTANCE*;*//单例模式* public static AppDataBase getInstance(Context context) {  synchronized (*Lock*) {  if (*INSTANCE* == null) {  *INSTANCE* =  Room.*databaseBuilder*(context.getApplicationContext(), AppDataBase.class, "apps.db")  .allowMainThreadQueries()  .build();  }  return *INSTANCE*;  }  }  public abstract AppDao appDao();  public AppDao getAppDao() {  return *INSTANCE*.appDao();  }  @NonNull  @Override  protected SupportSQLiteOpenHelper createOpenHelper(DatabaseConfiguration config) {  return null;  }  @NonNull  @Override  protected InvalidationTracker createInvalidationTracker() {  return null;  }  @Override  public void clearAllTables() {  } } |
|  | public class AppRecordsRepository {  AppDataBase appDataBase;  AppDao appDao;  public AppRecordsRepository(Application application) {  appDataBase = AppDataBase.*getInstance*(application);  appDao = appDataBase.getAppDao();  *//List<App> allApps = appDao.getAllApps();* }  public LiveData<List<App>> getLastApp(int limit) {  return appDao.getLastUsedApp(limit);  }  public LiveData<List<App>> getMostUsedApps(int limit) {  return appDao.getMostUsedApps(limit);  }  public LiveData<List<App>> getMostCountsApps(int limit) {  return appDao.getMostCountsApps(limit);  }  public void insert(OneUse oneUse) {  new AppRecordsRepository.insertOneUseAsyncTask(appDao).execute(oneUse);  }  public LiveData<List<OneUse>> getAllOneUses() {  return appDao.getAllOneUse();  }  public void delete(OneUse oneUse) {  appDao.delete(oneUse);  }  public void deleteAll() {  appDao.deleteApps();  appDao.deleteOneUses();  }  private static class insertAsyncTask extends AsyncTask<App, Void, Void> {  private final AppDao mAsyncTaskDao;  insertAsyncTask(AppDao dao) {  mAsyncTaskDao = dao;  }  @Override  protected Void doInBackground(final App... params) {  try {  mAsyncTaskDao.insertApp(params[0]);  } catch (Exception e) {  e.printStackTrace();  }  return null;  }  }  private static class insertOneUseAsyncTask extends AsyncTask<OneUse, Void, Void> {  private final AppDao mAsyncTaskDao;  insertOneUseAsyncTask(AppDao dao) {  mAsyncTaskDao = dao;  }  @Override  protected Void doInBackground(final OneUse... params) {  try {  mAsyncTaskDao.insertOneUse(params[0]);  } catch (Exception e) {  e.printStackTrace();  }  return null;  }  } } |
|  | @Entity public class Event {  @PrimaryKey(autoGenerate = true)  long id;  long eventType;  String appName;  String pkgName;  long timeStamp;  public Event() {  }  public long getId() {  return id;  }  public void setId(long id) {  this.id = id;  }  public String getAppName() {  return appName;  }  public void setAppName(String appName) {  this.appName = appName;  }  public String getPkgName() {  return pkgName;  }  public void setPkgName(String pkgName) {  this.pkgName = pkgName;  }  public long getTimeStamp() {  return timeStamp;  }  public void setTimeStamp(long timeStamp) {  this.timeStamp = timeStamp;  }  public long getEventType() {  return eventType;  }  public void setEventType(long eventType) {  this.eventType = eventType;  } } |
|  | @Entity public class OneUse {  @PrimaryKey(autoGenerate = true)  long id;  *//应用名* String appName;  *//包名* String pkgName;  @Ignore  Drawable icon;  *//使用开始时间* long startTimestamp;  *//使用时长* long spendTime;  *//开始使用时的电量* private int battery;  *//结束时的电量* private int batteryAfter;  *//开始使用时，是否在充电* private int charging;  *//开始使用时的网络状态* private int net;  public OneUse() {  }  public String getAppName() {  return appName;  }  public void setAppName(String appName) {  this.appName = appName;  }  public int getBatteryAfter() {  return batteryAfter;  }  public void setBatteryAfter(int batteryAfter) {  this.batteryAfter = batteryAfter;  }  public String getPkgName() {  return pkgName;  }  public void setPkgName(String pkgName) {  this.pkgName = pkgName;  }  public Drawable getIcon() {  return icon;  }  public void setIcon(Drawable icon) {  this.icon = icon;  }  public long getId() {  return id;  }  public void setId(long id) {  this.id = id;  }  public int getBattery() {  return battery;  }  public void setBattery(int battery) {  this.battery = battery;  }  public int getCharging() {  return charging;  }  public void setCharging(int charging) {  this.charging = charging;  }  public int getNet() {  return net;  }  public void setNet(int net) {  this.net = net;  }  @Override  public String toString() {  return "OneUse{" +  "appName='" + appName + '\'' +  ", startTimestamp=" + startTimestamp +  ", spendTime=" + spendTime +  '}';  }  public long getStartTimestamp() {  return startTimestamp;  }  public void setStartTimestamp(long startTimestamp) {  this.startTimestamp = startTimestamp;  }  public long getSpendTime() {  return spendTime;  }  public void setSpendTime(long spendTime) {  this.spendTime = spendTime;  }  */////////* public String getTimeSpendString() {  long sec = spendTime / 1000;  if (sec < 60)  return sec + "秒";  long min = sec / 60;  sec = sec % 60;  return min + "分" + sec + "秒";  }  public String getDatatime() {  SimpleDateFormat sdf = new SimpleDateFormat("MM/dd HH:mm:ss", Locale.*getDefault*());  return sdf.format(startTimestamp);  }  public String getNetString() {  if (net == 1) return "移动网络";  else if (net == 2) return "wifi";  else return "无网络";  }  public String getChargingString() {  if (charging == 1) return "充电中";  else if (charging == 2) return "USB充电";  return "未充电";  } } |
|  | public abstract class AppsAdapter extends RecyclerView.Adapter<AppsAdapter.ViewHolder> {  private List<App> mAppsList;  public AppsAdapter() {  }  public void setmAppsList(List<App> mAppsList) {  this.mAppsList = mAppsList;  }  @NonNull  @Override  public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  View view = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*recycle\_app\_item*, parent, false);  ViewHolder holder = new ViewHolder(view);  return holder;  }  @Override  public void onBindViewHolder(@NonNull ViewHolder holder, int position) {  App app = mAppsList.get(position);  holder.appIcon.setBackground(app.getIcon());  holder.appName.setText(app.getAppName());  setAppInfo(holder.appInfo, app);  }  *//设置* public abstract void setAppInfo(TextView appInfo, App app);  @Override  public int getItemCount() {  if (mAppsList != null)  return mAppsList.size();  return 0;  }  static class ViewHolder extends RecyclerView.ViewHolder {  ImageView appIcon;  TextView appName;  TextView appInfo;  public ViewHolder(View view) {  super(view);  appIcon = view.findViewById(R.id.*appicon*);  appName = view.findViewById(R.id.*appname*);  appInfo = view.findViewById(R.id.*app\_info*);  }  } } |
|  | public class AppsLastUseAdapter extends AppsAdapter {  public static final int *APP\_LIST\_SIZE* = 100;  public AppsLastUseAdapter(Fragment context, AppRecordsRepository appRecordsRepository) {  appRecordsRepository.getLastApp(*APP\_LIST\_SIZE*).observe(context, new Observer<List<App>>() {  @Override  public void onChanged(List<App> apps) {  DBUtils.*setAppsIcon*(context.getContext(), apps);  setmAppsList(apps);  notifyDataSetChanged();  }  });  }  @Override  public void setAppInfo(TextView appInfo, App app) {  appInfo.setText(DBUtils.*getSinceTimeString*(app.getLastRuningTime()) + "前");  } } |
|  | public class AppsTotalCountAdapter extends AppsAdapter {  public static final int *APP\_LIST\_SIZE* = 100;  public AppsTotalCountAdapter(Fragment context, AppRecordsRepository appRecordsRepository) {  appRecordsRepository.getMostCountsApps(*APP\_LIST\_SIZE*).observe(context, new Observer<List<App>>() {  @Override  public void onChanged(List<App> apps) {  DBUtils.*setAppsIcon*(context.getContext(), apps);  setmAppsList(apps);  notifyDataSetChanged();  }  });  }  @Override  public void setAppInfo(TextView appInfo, App app) {  appInfo.setText(app.getUseCount() + "次");  } } |
|  | public class AppsTotalTimeAdapter extends AppsAdapter {  public static final int *APP\_LIST\_SIZE* = 100;  public AppsTotalTimeAdapter(Fragment context, AppRecordsRepository appRecordsRepository) {  appRecordsRepository.getMostUsedApps(*APP\_LIST\_SIZE*).observe(context, new Observer<List<App>>() {  @Override  public void onChanged(List<App> apps) {  DBUtils.*setAppsIcon*(context.getContext(), apps);  setmAppsList(apps);  notifyDataSetChanged();  }  });  }  @Override  public void setAppInfo(TextView appInfo, App app) {  appInfo.setText(DBUtils.*getTimeSpendString*(app.getTotalRuningTime()));  } } |
|  | public class MyFragmentPagerAdapter extends FragmentPagerAdapter {  private final int PAGER\_COUNT = 3;  private final List<Fragment> fragments = new ArrayList<>();  public MyFragmentPagerAdapter(@NonNull FragmentManager fm, int behavior) {  super(fm, behavior);  initFragments();  }  private void initFragments() {  fragments.add(new RcordFragment());  fragments.add(new HomeFragment2());  fragments.add(new SettingsFragment());  }  @NonNull  @Override  public Fragment getItem(int position) {  Fragment fragment = null;  switch (position) {  case 0:  fragment = fragments.get(0);  break;  case 1:  fragment = fragments.get(1);  break;  case 2:  fragment = fragments.get(2);  break;  }  return fragment;  }  @Override  public int getCount() {  return PAGER\_COUNT;  } } |
|  | public class OneUseAdapter extends RecyclerView.Adapter<OneUseAdapter.ViewHolder> {  private List<OneUse> oneUses;  private int position;  public OneUseAdapter(List<OneUse> oneUses) {  this.oneUses = oneUses;  }  public int getPosition() {  return position;  }  public void setPosition(int position) {  this.position = position;  }  @NonNull  @Override  public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  View view = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*oneuse\_list\_item*, parent, false);  ViewHolder holder = new ViewHolder(view);  return holder;  }  @RequiresApi(api = Build.VERSION\_CODES.*N*)  @Override  public void onBindViewHolder(@NonNull ViewHolder holder, int position) {  OneUse oneuse = oneUses.get(position);  holder.appIcon.setBackground(oneuse.getIcon());  holder.appName.setText(oneuse.getAppName());  holder.spendTime.setText(oneuse.getTimeSpendString());  holder.timeStamp.setText(oneuse.getDatatime());  holder.battery.setText("" + oneuse.getBattery() + "->" + oneuse.getBatteryAfter());  holder.charging.setText(oneuse.getChargingString());  holder.net.setText(oneuse.getNetString());  holder.listView.setOnLongClickListener(new View.OnLongClickListener() {  @Override  public boolean onLongClick(View v) {  setPosition(position);  Log.*d*("TAG", "onLongClick: " + position);  return false;  }  });  }  @Override  public int getItemCount() {  if (oneUses == null)  return 0;  return oneUses.size();  }  public void setOneUses(List<OneUse> filterByName) {  this.oneUses = filterByName;  }  static class ViewHolder extends RecyclerView.ViewHolder implements View.OnCreateContextMenuListener {  LinearLayout listView;  ImageView appIcon;  TextView id;  TextView appName;  TextView spendTime;  TextView timeStamp;  TextView battery;  TextView charging;  TextView net;  public ViewHolder(View view) {  super(view);  view.setOnCreateContextMenuListener(this);  listView = view.findViewById(R.id.*list\_view*);  appIcon = view.findViewById(R.id.*appicon\_img*);  id = view.findViewById(R.id.*id\_tv*);  appName = view.findViewById(R.id.*appname\_tv*);  spendTime = view.findViewById(R.id.*spendtime\_tv*);  timeStamp = view.findViewById(R.id.*timestamp\_tv*);  battery = view.findViewById(R.id.*battery\_tv*);  charging = view.findViewById(R.id.*charging\_tv*);  net = view.findViewById(R.id.*net\_tv*);  }  @Override  public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {  menu.setHeaderTitle(appName.getText());  menu.add(0, 0, 0, "删除记录");  }  } } |