1. 设计说明
   1. 日记：
      1. 功能设计

一个能够写日记的模块，能够实现日记的增删改查，并且实现了备份和恢复功能；

* + 1. 数据库设计

Diary(id, date, title, tag, content)

id:日记的主键id

date：日记的时间

title：日记的标题

tag：删除日记时候的标记

content：日记的内容

* + 1. 界面设计

基于Martial Design的设计，按钮使用FloatingActionButton控件

日记主界面左侧有时间轴，并且能够高亮显示当日的日记内容

点击日记出现编辑选项，点击选项进入编辑页面

* 1. 新闻：
     1. 功能设计

提供浏览新闻的功能

* + 1. 数据库设计

无

* + 1. 界面设计

基于顶部导航栏的新闻浏览界面



新闻内容页则在Android提供的WebView控件上展示，具体页面为网上提供的新闻页



* 1. 跑步
     1. 功能设计

基于高德地图的跑步功能，能够记录历史并且能够备份和恢复

* + 1. 数据库设计

Run(id, date, distance, time)

id:日记的主键id

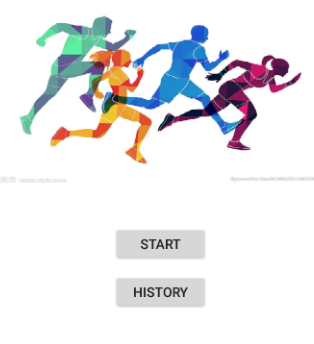
date：跑步的时间

distance：跑步的距离

time：跑步所花时间

* + 1. 界面设计

主页



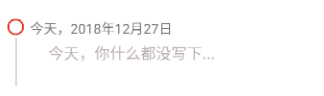
历史记录



跑步主界面



1. 实现说明
   1. 日记
      1. 界面实现
         1. 左侧时间轴的实现



这里用到了layer-list

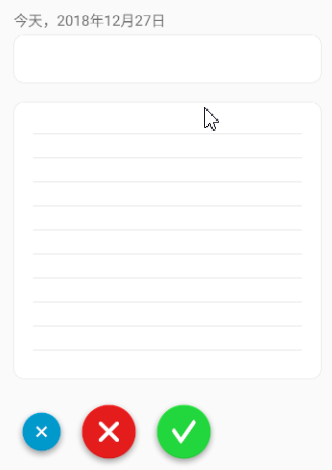
layer-list 的大致原理类似 RelativeLayout（或者FrameLayout），创建出来的是drawable资源，可以通过设置边框宽度来构造竖线

* + - 1. 悬浮框实现



悬浮框的实现用到了FloatingActionButton控件，这个控件是从github上下载下来的，在layout布局文件中设置FloatingActionsMenu和FloatingActionsButton即可

* + - 1. 日记书写主体区域设计



中间书写主体的横线效果是用网上现成的代码来实现的，文件存放在widget目录下的LinedExited中，

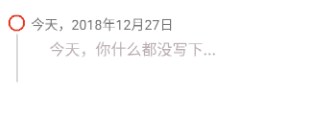
同时需要在layout布局文件中引用这个标签<cn.edu.zucc.bigapp.diary.widget.LinedEditText/>

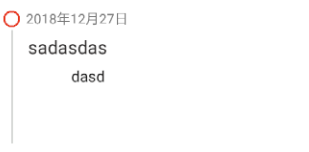
书写主体的圆角效果通过设置xml文件，然后在LinedEdited标签中设置background属性并且引用即可

* + 1. 功能实现
       1. 日记列表的实现

recycleView实现，具体不再赘述

当还未书写日记内容时和书写日记后的界面变化





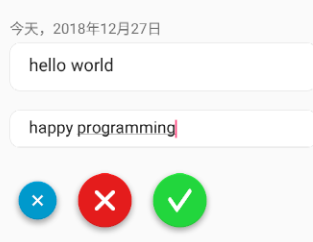
在layout布局文件中内置一串文字，当书写日记后把内置的内容删除，再把新写入的内容添加到数据库然后显示到界面上

* + - 1. 时间轴高亮显示

每个日记模块都有书写时间，把这个书写时间和当天时间进行比较，如果不相等，则把圆圈的颜色变灰，如果相等，这把圆圈颜色变红；每个圆圈即是一个存放在drawable目录中的图片文件，颜色变化即是图片变化

* + - 1. 添加日记

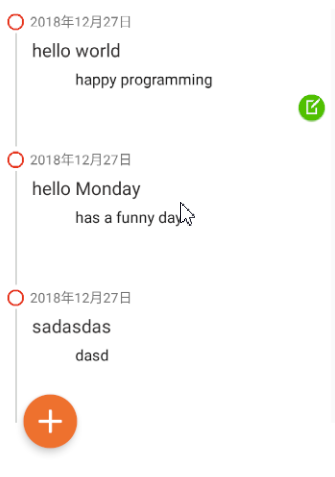
在相应位置填入具体的内容，填写好之后点击浮动按钮选择取消或者确定，然后写入数据库中，操作好之后会退到主界面



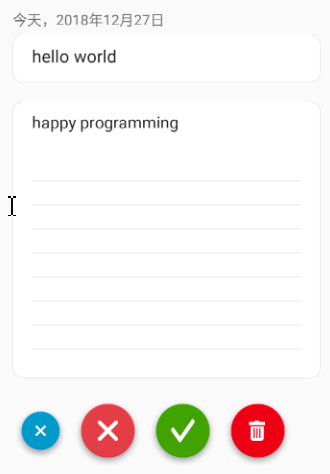


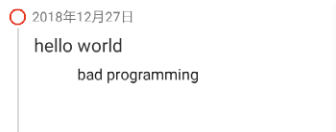
* + - 1. 编辑日记

在日记列表页点击具体的日记项，会出现编辑按钮（在布局文件中内置编辑按钮，通过设置visibility来控制显示与否，当点击列表项时设置为可见，否则设置为不可见）

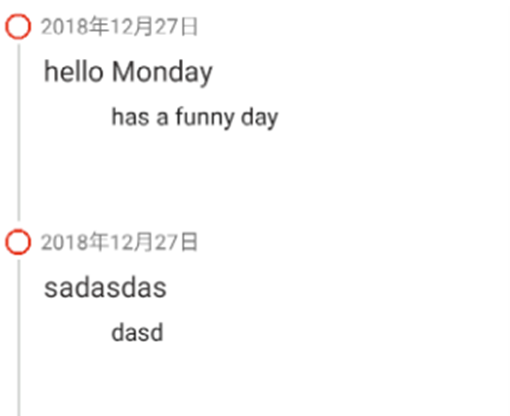


进入编辑区域，按照流程操作，浮动按钮设有取消，确定，删除，回退功能，相应功能由具体的数据库操作实现，删除时会弹框提示是否删除









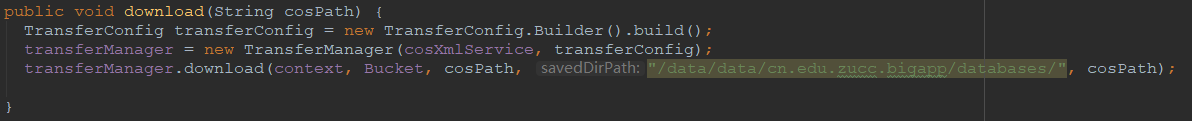
* + - 1. 数据备份与恢复

数据备份在腾讯云上的OOS上，依据OOS的官方文档来通过相应的配置来把日记存储到OOS上，恢复也是同样的道理

OOS地址：



建立本地数据库与OOS的联系



备份上传

yun.upload("/data/data/cn.edu.zucc.bigapp/databases/Diary.db", "Diary.db");

恢复下载

yun.download("Diary.db");

* + - 1. 数据库操作

数据库配置：

public static final String *CREATE\_DIARY* = "create table Diary("  
 + "id integer primary key autoincrement, "  
 + "date text, "  
 + "title text, "  
 + "tag text, "  
 + "content text)";

增加日记：

SQLiteDatabase db = mHelper.getWritableDatabase();  
ContentValues values = new ContentValues();  
values.put("date", dateBack);  
values.put("title", titleBack);  
values.put("content", contentBack);  
values.put("tag", tag);  
db.insert("Diary", null, values);  
values.clear();

删除日记：

String tag = mTvTag.getText().toString();  
SQLiteDatabase dbDelete = mHelper.getWritableDatabase();  
dbDelete.delete("Diary", "tag = ?", new String[]{tag});

修改日记：

SQLiteDatabase dbUpdate = mHelper.getWritableDatabase();  
ContentValues valuesUpdate = new ContentValues();  
String title = mUpdateDiaryEtTitle.getText().toString();  
String content = mUpdateDiaryEtContent.getText().toString();  
String tag = mTvTag.getText().toString();  
valuesUpdate.put("title", title);  
valuesUpdate.put("content", content);  
dbUpdate.update("Diary", valuesUpdate, "tag = ?", new String[]{tag});

查询日记：

do {  
 String date = cursor.getString(cursor.getColumnIndex("date"));  
 String title = cursor.getString(cursor.getColumnIndex("title"));  
 String content = cursor.getString(cursor.getColumnIndex("content"));  
 String tag = cursor.getString(cursor.getColumnIndex("tag"));  
 mDiaryBeanList.add(new DiaryBean(date, title, content, tag));  
} while (cursor.moveToNext());

* + - 1. EventBus实现组件间的通信

当我们进行项目开发的时候，往往是需要应用程序的各组件、组件与后台线程间进行通信，比如在子线程中进行请求数据，当数据请求完毕后通过Handler或者是广播通知UI，而两个Fragment之家可以通过Listener进行通信等等。当我们的项目越来越复杂，使用Intent、Handler、Broadcast进行模块间通信、模块与后台线程进行通信时，代码量大，而且高度耦合。

自定义事件类StartUpdateDiaryEvent

DiaryActivity中注册事件

EventBus.*getDefault*().register(this);

DiaryActivity中解除注册

@Override  
protected void onDestroy() {  
 super.onDestroy();  
 EventBus.*getDefault*().unregister(this);  
}

DiaryAdapter中触发事件

holder.mIvEdit.setOnClickListener(v -> EventBus.*getDefault*().post(new StartUpdateDiaryEvent(position)));

DiaryActivity监听事件

@Subscribe  
public void startUpdateDiaryActivity(StartUpdateDiaryEvent event) {  
 String title = mDiaryBeanList.get(event.getPosition()).getTitle();  
 String content = mDiaryBeanList.get(event.getPosition()).getContent();  
 String tag = mDiaryBeanList.get(event.getPosition()).getTag();  
 UpdateDiaryActivity.*startActivity*(this, title, content, tag);  
}

* + - 1. SwipeRefreshLayout实现页面的刷新

1. 定义<android.support.v4.widget.SwipeRefreshLayout>标签

2. 在DiaryActivity中设置刷新监听

srlDiary.setOnRefreshListener(() -> {  
 mDiaryBeanList = getDiaryBeanList();  
 srlDiary.setRefreshing(false);  
 mMainRvShowDiary.setAdapter(new DiaryAdapter(mContext, mDiaryBeanList));  
});

* 1. 新闻
     1. 界面实现

主页基于ViewPager和Fragment实现导航栏切换

<android.support.design.widget.TabLayout/>  
<android.support.v4.view.ViewPager/>

新闻内容页基于Android提供的WebView显示网页

* + 1. 功能实现

新闻模块尝试采用MVP模式来构建

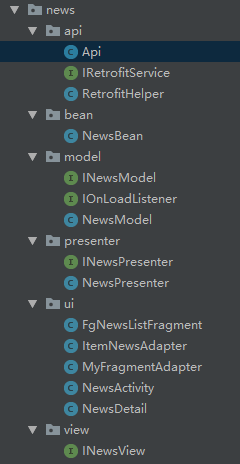
Model用来处理信息，比如获取API数据

Presenter用来处理逻辑

View用来处理页面，比如处理数据加载时的刷新控件

网络框架采用Retrofit+OkHttp

程序整体的架构



1. API：用来设置比如API提供商的APPKEY，ApiUrl等参数

2. IRetrofitService，RetrofitHelper：Retrofit,Okhttp获取网络数据的基本设置

3. NewsBean：新闻对象

4. INewsModel：Model模块提供的接口，用来加载新闻的种类数据

5. IOnLoadListener：类似于回调函数，用来实现数据获取成功或者失败后执行的函数

6. NewsModel：实现了INewsModel接口

7. INewsPresenter：Presenter提供的接口，用来加载新闻数据

8. NewsPresenter：实现了INewsPresenter接口

9. FgNewsListFragment：新闻列表页的fragment

10. ItemNewsAdapter：存放新闻RecycleView的Adapter

11. MyFragmentAdapter：处理fragment的Adapter

12. NewsActivity：新闻主Activity

13. NewsDetail：新闻详情页

14. INewsView：View模块提供的接口，用来实现数据加载时的view操作

* + - 1. ViewPager和Fragment实现导航栏切换

private void setViewPager() {  
 fragments.add(FgNewsListFragment.*newInstance*(*NEWS\_TYPE\_SOCIAL*));  
 fragments.add(FgNewsListFragment.*newInstance*(*NEWS\_TYPE\_SPORT*));  
 fragments.add(FgNewsListFragment.*newInstance*(*NEWS\_TYPE\_TECH*));  
 fragmentTitles.add("社会");  
 fragmentTitles.add("体育");  
 fragmentTitles.add("科技");  
  
 MyFragmentAdapter adapter = new MyFragmentAdapter(getSupportFragmentManager(), fragments, fragmentTitles);  
  
 vpNews.setAdapter(adapter);  
}

* + - 1. SwipeRefreshLayout实现页面刷新

页面自动刷新

srlNews.post(() -> {  
 showDialog();  
 presenter.loadNews(type, 0);  
});

页面手动刷新

srlNews.setOnRefreshListener(() -> presenter.loadNews(type, 0));

* + - 1. 实现INewsModel接口提供的显示新闻方法

@Override  
public void showNews(NewsBean newsBean) {  
 getActivity().runOnUiThread(new TimerTask() {  
 @Override  
 public void run() {  
 newsBeanList = newsBean.getNewslist();  
 adapter.setData(newsBeanList);  
 rvNews.setLayoutManager(new LinearLayoutManager(getActivity(), LinearLayoutManager.*VERTICAL*, false));  
 rvNews.setAdapter(adapter);  
 tvNewsList.setVisibility(View.*GONE*);  
 }  
 });  
}

* + - 1. 实现INewsPresenter接口提供的获取数据方法

@Override  
public void loadNews(int type, int startPage) {  
 iNewsView.showDialog();  
 switch (type) {  
 case NewsActivity.*NEWS\_TYPE\_SOCIAL*:  
 iNewsModel.loadNews(Api.*SOCIAL\_ID*, this);  
 break;  
 case NewsActivity.*NEWS\_TYPE\_SPORT*:  
 iNewsModel.loadNews(Api.*SPORT\_ID*, this);  
 break;  
 case NewsActivity.*NEWS\_TYPE\_TECH*:  
 iNewsModel.loadNews(Api.*TECH\_ID*, this);  
 break;  
 }  
}

而实现自INewsPresenter的类NewsPresenter会去获取数据

@Override  
public void loadNews(final String type,  
 final IOnLoadListener iOnLoadListener) {  
 RetrofitHelper retrofitHelper = new RetrofitHelper(Api.*NEWS\_HOST*);  
 retrofitHelper.getNews(type).enqueue(new Callback<NewsBean>() {  
 @Override  
 public void onResponse(Call<NewsBean> call, Response<NewsBean> response) {  
 if (response.isSuccessful()) {  
 iOnLoadListener.success(response.body());  
 } else {  
 iOnLoadListener.fail("");  
 }  
 }  
  
 @Override  
 public void onFailure(Call<NewsBean> call, Throwable t) {  
 iOnLoadListener.fail(t.toString());  
 }  
 });  
}

* + - 1. 新闻详情页的显示

就是最基本的在Webview上显示网页的方法

* 1. 跑步
     1. 界面实现

基于高德地图的界面显示

<com.amap.api.maps.MapView/>

基于RecycleView的列表显示

基于FloatingActionButton的按钮控件

* + 1. 功能实现

跑步功能全部按照高德地图官方文档提供的方法来实现

数据备份与恢复参照日记模块实现

计时器功能选用了Android提供的Chronometer控件

查看和插入数据库基本操作