**EVERNOTE API 调查：**

沙盒：沙盒是一种单独的印象笔记服务实例，是一个安全的测试环境，与正式服务器环境不同。一开始获得的api key 只能在沙盒中使用，当app开发完毕时，提交审核，则会可以访问正式的服务系统。

API的调用频率：api如果超出了一个小时内规定的调用频率，则会报错（[EDAMSystemException](https://documentation/reference/Errors.html" \l "Struct_EDAMSystemException)），需要自己捕获处理。如果是sandbox环境则等待15秒就可以重新请求，而生产环境下，会一个小时内不能请求。

API Key 的权限：1.基本权限，2.完整权限

两种认证方式：1.token，2.OAuth,建议使用OAuth.

撤销和过期：

初始化：es = new EvernoteSession.Builder(this)  
 .setEvernoteService(EvernoteSession.EvernoteService.*SANDBOX*)  
 .setForceAuthenticationInThirdPartyApp(true)  
 .setSupportAppLinkedNotebooks(true)  
 .setLocale(Locale.*CHINESE*)  
// .buildForSingleUser(ConstantUtils.TOKEN, ConstantUtils.SELF\_URL)  
 .build(ConstantUtils.*CONSUMER\_KEY*, ConstantUtils.*CONSUMER\_SECRET*)  
 .asSingleton();

创建一个note:

final EvernoteNoteStoreClient noteStoreClient = EvernoteSession.*getInstance*().getEvernoteClientFactory().getNoteStoreClient();  
  
Note note = new Note();  
note.setTitle("app created");  
//必须设置为EML  
note.setContent(EvernoteUtil.*NOTE\_PREFIX* + "this is a content from app " + EvernoteUtil.*NOTE\_SUFFIX*);  
  
noteStoreClient.createNoteAsync(note, new EvernoteCallback<Note>() {  
 @Override  
 public void onSuccess(Note result) {  
 Toast.*makeText*(getApplicationContext(), result.getTitle() + " has been created", Toast.*LENGTH\_LONG*).show();  
 }  
  
 @Override  
 public void onException(Exception exception) {  
  
 }  
});

获取所有的notebooks:

es.getEvernoteClientFactory().getNoteStoreClient().listNotebooksAsync(new EvernoteCallback<List<Notebook>>() {  
 @Override  
 public void onSuccess(List<Notebook> result) {  
 List<String> namesList = new ArrayList<String>(result.size());  
 for (Notebook notebook : result) {  
 namesList.add(notebook.getName());  
 }  
  
 String notebookNames = TextUtils.*join*(",", namesList);  
 Toast.*makeText*(MainActivity.this, notebookNames + " notebooks have been retrieved", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 @Override  
 public void onException(Exception exception) {  
 Log.*e*("123", "Error retrieving notebooks", exception);  
 }  
});

获取默认的notebooks:

*/\*\*  
 \* note  
 \*/*es.getEvernoteClientFactory().getNoteStoreClient().getDefaultNotebookAsync(new EvernoteCallback<Notebook>() {  
 @Override  
 public void onSuccess(final Notebook result) {  
  
 Toast.*makeText*(MainActivity.this, "" + result.getGuid() + "\r\n" + result.getName(), Toast.*LENGTH\_SHORT*).show();  
 }  
  
 @Override  
 public void onException(Exception exception) {  
  
 }  
});

获取note：异步获取

过滤：通过NoteFilter类来设置setWord过滤查找的类型，MIME

通过setOrder来进行排序的设置。

NoteFilter noteFilter = new NoteFilter();  
noteFilter.setOrder(NoteSortOrder.*CREATED*.getValue());  
noteStoreClient.findNotesAsync(noteFilter, 0, 50, new EvernoteCallback<NoteList>() {  
 @Override  
 public void onSuccess(NoteList result) {  
 List<Note> notes = result.getNotes();  
 noteStoreClient.getNoteAsync(notes.get(2).getGuid(), false, false, false, false, new EvernoteCallback<Note>() {  
 @Override  
 public void onSuccess(Note result) {  
  
 Toast.*makeText*(getApplicationContext(), result.getTitle() + "..." + result.getContent(), Toast.*LENGTH\_SHORT*).show();  
 }  
  
 @Override  
 public void onException(Exception exception) {  
  
 }  
 });  
  
 }  
  
 @Override  
 public void onException(Exception exception) {  
  
 }  
});

注意：上述获取的不包含content内容，

1. 通过note的guid(global unique identifier),设置获取content，来获取content。
2. 并且获取的content为html格式的文字，需要使用html类来进行格式化。

noteStoreClient.getNoteAsync(notes.get(2).getGuid(), true, false, false, false, new EvernoteCallback<Note>() {  
 @Override  
 public void onSuccess(Note result) {  
  
 Toast.*makeText*(getApplicationContext(), result.getTitle() + "..." + result.getContent(), Toast.*LENGTH\_SHORT*).show();  
 Log.*i*("123", "onSuccess: "+result.getTitle() + "..." + Html.*fromHtml*(result.getContent()) );

1. 通过 EvernoteHtmlHelper类出入note的guid来获取解析响应获取数据。

final EvernoteHtmlHelper ehp = EvernoteSession.*getInstance*().getEvernoteClientFactory().getHtmlHelperDefault();  
  
 try {  
 ehp.downloadNoteAsync(notes.get(2).getGuid(), new EvernoteCallback<Response>() {  
 @Override  
 public void onSuccess(Response result) {  
  
 try {  
 String s= ehp.parseBody(result);  
 Toast.*makeText*(getApplicationContext(), s, Toast.*LENGTH\_SHORT*).show();  
 Log.*i*("123", "path :"+ Html.*fromHtml*(s));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }

开发UI：

1. 抽屉使用：DrawerLayout 与 NavigationView
2. Fab:浮动action button的使用，自定义。
3. Toolbar的使用
4. recyclerView列表使用。

功能可行性：

1. 登录，抽屉log out ，OK