交接文档——陈世超

主要工作内容

- 闪屏页
    1. 工信部要求修改；
    2. 闪屏页切换卡顿及图片拉伸变形优化；
    3. Android P 以上刘海屏适配；
    4. 闪屏页跳转按钮动画。
- 引导页重构
- 首页TabActivity重构为**ViewPager2+Fragment**
- 笔记及视频详情页转场动画
- 视频详情页UI调整
- 点评详情页新增字段
- 混淆新增ViewBinding支持
- 新增搜索结果页视频Tab
- 新增科室主页视频Tab
- 笔记列表毛玻璃效果
- 钱包第三方账号绑定及封装
- 新增收藏聚合页面
- 新增关注聚合页面
- 机构介绍页
- 自定义控件：
    1. **CHomeBottomTabBar**
    2. **TopBar**
    3. **NyTextButton**
    4. **NyImageButton**
    5. **NyConstraintLayout**

6. **NyRelativeLayout**
7. **NyLinearLayout**
8. **NyFrameLayout**
9. **NyExpandTextView**
10. **AbstractDialog**
11. **ConfirmDialog**

- 工具类及扩展类：

1. **NyLog**
2. **ViewUtil**
3. **DensityUtil**
4. **ActivityExt**
5. **FragmentExt**
6. **DensityExt**
7. **ViewExt**
8. **CEventCenter**

可能存在的问题

- 笔记列表毛玻璃效果性能问题

  目前毛玻璃效果是利用Glide框架对图片进行高斯模糊，经测试，每张普通图片处理大致耗时10ms内，耗时和图片质量成正比，图片质量越高，越耗时。如果是单张图片则不存在卡顿感知，但笔记列表是多张图片的情况，如果用户快速滑动，卡顿会很明显，同时内存占用过高，频繁分配内存及gc回收也会出现内存抖动的情况。当前较少用户连续发布私密笔记，导致测试无法发现此Bug。建议采用服务器图片处理解决，例如Ali OSS对象存储的图片处理，根据图片地址+高斯模糊参数即可拿到毛玻璃效果的图片，客户端不存在性能问题。

- 内存泄露

  开发中，发现多处代码会导致内存泄露。例如患者端**MainActivity**，不应该直接创建Activity的static instance，这会导致引用无法回收从而出现内存泄露的问题。

- NyTextButton/NyImageButton/NyConstraintLayout/NyRelativeLayout/NyLinearLayout/NyFrameLayout

  不支持动态设置**backgroundColor**/**cornorsRadius**/**gradientColors**等。由于Ny系列为多state控件，所以目前无法动态设置以上属性。

- 友盟Resource FileNotFoundException相关

友盟上频繁crash的问题，是因为某些特定机型无法使用res/color下的selector/shape等功能，此Bug涉及范围过广，云测试机暂时未复现该Bug。

建议

- 网络请求

  目前网络请求库使用稍繁琐，客户端请求接口大致流程为：1. 声明**ApiService**；2. 创建**Model**/**Repository**；3. **Presenter**/**ViewModel**发起请求。同时，由于当前网络请求库封装局限，在接口请求失败时，无法获取errorCode，导致客户端业务异常处理有很大的局限。结合以上情况，封装了一个网络请求库：Shine-Kotlin，在ApiService层抽象**GET**/**POST**/**PUT**/**DELETE**等接口，请求流程为：1. 创建**Model**/**Repository**；2. **Presenter**/**ViewModel**发起请求，当请求新接口或旧接口参数发生改变时，无需修改**ApiService**层。同时，提供动态**baseUrl**设置及**IParser**抽象，当服务端**baseUrl**或**ReponseModel**不统一时，通过动态设置**baseUrl**或自定义**Parser**即可兼容，同时提供异步/同步请求能力。当前仅为**Kotlin**版本，可以参考并统一公司的网络库。

- 页面下拉刷新及加载更多复用

  目前项目多个列表页面具有下拉刷新/加载更多的功能，通过这段时间的了解，几乎全都是每个页面在xml布局引入PullLayout，然后在代码中实现及维护页码、刷新状态等。考虑到下拉刷新/加载更多应为每个项目常用的功能，建议在NySupport库新增NyBaseRefreshActivity/NyBaseRefreshFragment的封装，业务层无需关心刷新及加载的逻辑，也无需关心页码及刷新状态，当某个新功能页面需要具备下拉刷新/加载更多功能时，通过继承并重写某些抽象方法传参即可，提升开发效率。否则，当需要替换下拉刷新/加载更多框架及实现时，修改成本非常高。同理，Glide之类等通用框架也应如此，应使用类似策略模式的方式进行封装，避免后续替换实现框架时付出巨大的开发成本。