意见：

1、不建议使用的框架：

1.1、建议不要用 YYKit，理由是作者长时间没有维护,随着 iOS 系统的不断更迭这明显是不合适的，里面不少方法接近废弃；

2、建议不要过余的使用 runtime 方法交换、增加阅读量和学习成本,给新来者带来困惑。需要特别强调或者特别的处理的除外；

2、对非持久化数据的管理

2.1、NSArray、NSDictionary、NSSet 及其子类的业务逻辑封装．加解密

2.2、对度量衡的统一处理

2.2.1、尺寸类：size、frame、origin 的宏定义、以及相关转换,详见文件 MacroDef_Size.h 、UIView+Measure.h

2.2.2、时间类：NSDate <=> NSString、年月日的处理、星期的处理、时分秒的处理、给定时间戳转换为特有的字符串或者数字等逻辑规则；详见文件 TimeModel 和 NSObject+Time

2.2.3、数据金额的处理、金额的处理 (保留位数精度、货币符号)

2.2.4、对颜色的定义，详见文件 MacroDef_Cor.h

2.2.5、对 App 里面的一些功能性的定义，详见文件 MacroDef_Func.h

2.2.6、对系统层面上一些功能的二次封装,详见文件 MacroDef_Sys.h

3、对持久化数据的管理

3.1、数据库使用达到统一，定死一个数据库，讨论适合和必要的使用场景和模式；

3.2、本地 NSBundle 文件的管理

3.3、本地 JSON 文件的管理

3.4、对用户 NSUserDefaults 的管理，用一个继承自 NSObject 的工具类进行管理，而不是用 NSUserDefaults 的分类的形式，将这个工具类完全赋予新的增删改查功能

3.5、读取黄色和蓝色文件夹的数据的管理

4、一些常规性的封装：

4.1、对 block 进行封装定义到最顶层直接调用；详见文件 #import "AABlock.h”

4.2、通知封装在顶层直接调用；详见文件#import "NotificationManager.h"

4.3、对全局枚举的统一管理；详见文件 DefineStructure.h

4.4、URL 的 Api 用 NSObject 的分类进行管理（方便点语法），而不是宏定义（加快编译速度）

4.5、数据解析全部统一用 MJExtension、YYModel(久不更) 和 MJExtension 一致，反而 JsonModel 需要继承解析速度慢

4.6、数据的处理在数据层处理完毕，外界直接用终值，不要在用的时候再去拼装

4.7、对于弹出框的封装；

例 1：https://github.com/SPStore/SPAlertController ，优势是满足

所有你所想，我自己在上面二次封装，有 demo，直接复制即可；

例 2：https://github.com/shmxybfq/TFPopup 不耦合 view 代码,可以为已创建过 未创建过的 view 添加弹出方式;只是一种弹出方式;

4.8、对悬浮的封装，UIView+SuspendView ==> #import "SuspendBtn.h”（悬浮按钮）、#import "SuspendLab.h”（悬浮标签）、#import "SuspendView.h”（悬浮视图）

4.9、对长链接的封装 pod 'Socket.IO-Client-Swift' 、详见文件：#import "SocketIOTools.h”（目前还没有对 MQTT 的封装，因为我们之前没涉及）

4.10、对网络监控的封装，一般用于开发和测试阶段 JobsBitsMonitor，例：https://github.com/295060456/JobsBitsMonitor

4.11、定位于文件 UICollectionView+RegisterClass。对全局的所有 UICollectionViewCell 及其子类进行统一管理，理论依据：仅仅注册是不开辟内存，只有用字符串取 cell 才会开辟

5、在以上的基础上，对一些常用的，重点业务逻辑、业务逻辑变动较为频繁的逻辑处理进行 2 次封装、比如：登录、注销、切换源、点赞、评论...

6、关于网络请求

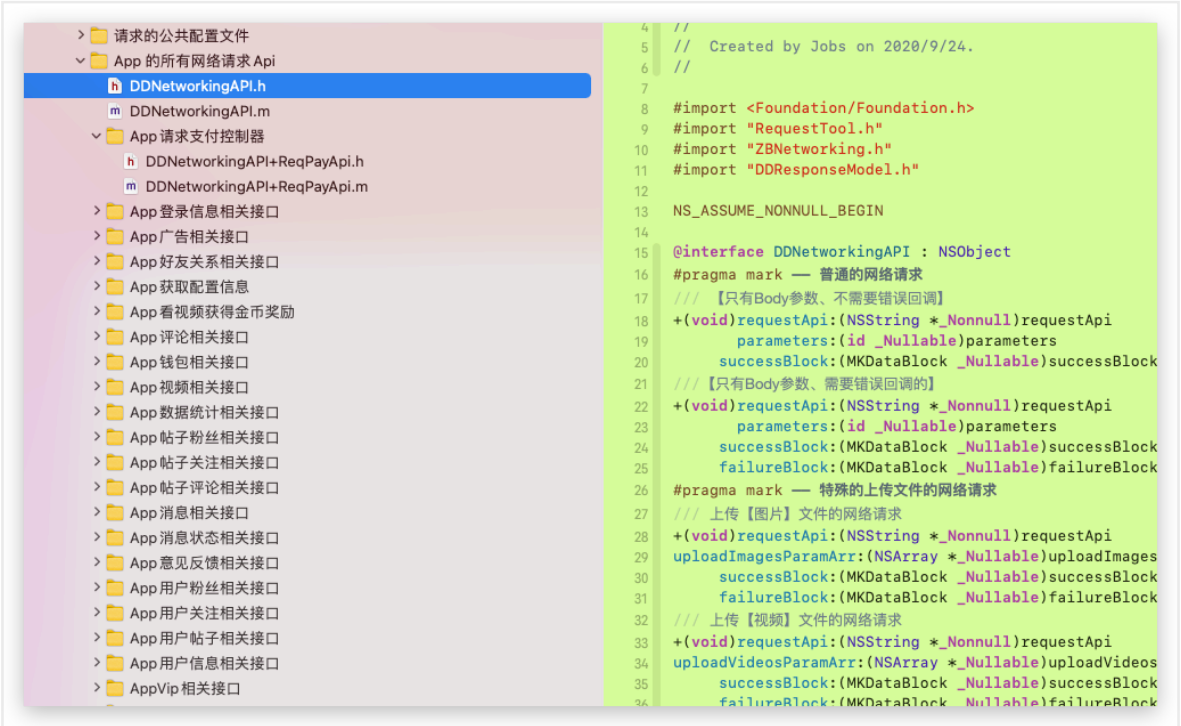
6.1、满足基本的 Get、Post

6.2、最好满足一些不常用的 PUT、Delete 以应对后台有时候需要我们用 put 上传大文件的需求

6.3、暴露修改请求头的接口

6.4、预处理模式（预处理请求、预处理回应）例：https://github.com/Suzhibin/ZBNetworking

6.5、网络请求用一个基类，其他的接口按照模块和功能的划分以分类的形式进行书写。因为调用的时候没有办法检索到分类的讯息，那么我这里用映射进行，意思就是方法名都是有规律的，不规律无法调用



关于调用也是：

7、注意几个重点模块：

7.1 登录模块，涉及到token的取法、是否过期问题

7.2、用户信息模块，用单例管理，看需求是否必要存数据库

7.3、计时器；https://github.com/295060456/JobsTimer，最好用GCD，NSTimer存在精度问题，这个例子里面我有封装，直接用，有Demo

7.4、时间模块：时间转换模块、时间戳转换模块；

7.5、对某一页面的所有/指定内容进行缓存的管理；

7.6、对于一些重型功能模块的二次封装的处理，比如播放器，最好在外层让程序员最方便调用而不是每次都要进行相同的配置处理

7.7、鉴权模块：

7.7.1、本机的相关权限模块，相机、定位、通讯录...定位文件ECPrivacyCheckTools有很好的封装

7.7.2、对特殊权限的处理：白名单处理、网络被墙切换的处理

7.8、对于VC的销毁。每个类必须写dealloc且卸载最上面、懒加载写在文件最后、中间写正常的生命周期各项按顺序之上而下排序。推VC只有2种

push 和 present 我对此进行了封装，因为 push 必须依赖于导航栏，没有导航栏推不了，我这里可以 push 不了直接 present。具体定位于 UIViewController+BaseVC

7.9、本机文件写入 pod 'TXFileOperation' # 文件夹操作 https://github.com/xtzPioneer/TXFileOperation 二次封装文件定位于：FileFolderHandleTool

8、 其他

8.1、全局用纯代码构建、因为纯代码不能直观的在不运行代码的情况下整个页面的情况，可以将需求 UI 截图，拖入 Xcode，只是不包含到工程里面即可，这样打包不会进行编译

8.2、全局图片用 Assets.xcassets，Xcode 对机型做了适配，图片有压缩，这个关乎包的大小以及启动速度

8.3、模块化开发、对不同环境设置不同的 icon 图标。相关文档位于：https://github.com/JobsKit/JobsKitDoc

8.4、对环境的切换：

8.4.1、在 DefineStructure.h 文件里面定义环境的名称；

8.4.2、定义 NetworkingEnvir networkingEnvir = DevEnviron_China_Mainland;/// 中国大陆开发环境

...

详见 https://github.com/295060456/JobsComment

8.5、一些系统类的封装 JobsBaseCustomizeUIKitCore，特别指出，分类和继承相互相成使用；

8.6、每一个单独的文件尽量保证 500~800 行，行数多了减少效率，一个文件作为一个大类来解决一些通用性问题，其子类解决问题的细分，形成一个簇状结构

8.7、协议 + 分类的形式可以将功能模块独立出来，否则直接怼在一个文件里，对于较重的业务模块不是很友好；

8.8、强烈建议广泛的用懒加载，因为这样可以顺着生命周期排查问题。在 UI 层我的习惯是创建控件和控件的约束一起写在懒加载里面，用的时候用 x.alpha = 1 来进行调用，其他属性以此类推

9、一些行业普遍共用的功能模块：（我们好好合计下，一些不强壮的、不活跃的、功能有重叠的应该被舍弃。以下库都是我踩过坑的，在多个项目运用，性能比较稳定）

pod 'Masonry' # https://github.com/SnapKit/Masonry NO_SMP

pod 'AFNetworking' # https://github.com/AFNetworking/AFNetworking YES_SMP

pod 'Reachability' # https://github.com/tonymillion/Reachability 检查联网情况 NO_SMP

pod 'ReactiveObjC' # https://github.com/ReactiveCocoa/ReactiveObjC NO_SMP

pod 'MJRefresh' # https://github.com/CoderMJLee/MJRefresh NO_SMP

pod 'MJExtension' # https://github.com/CoderMJLee/MJExtension NO_SMP

pod 'SDWebImage' # https://github.com/SDWebImage/SDWebImage

YES_SMP
pod 'YQImageTool'
pod 'OpenUDID' # https://github.com/ylechelle/OpenUDID Open source initiative for a universal and persistent UDID solution for iOS
pod 'TABAnimated' # https://github.com/tigerAndBull/TABAnimated // 骨架屏
pod 'GKNavigationBar' # https://github.com/QuintGao/GKNavigationBar NO_SMP
pod 'GKPhotoBrowser' # https://github.com/QuintGao/GKPhotoBrowser iOS仿微信、今日头条等图片浏览器
pod 'JXCategoryView' # https://github.com/pujiaxin33/JXCategoryView NO_SMP
pod 'JXPagingView/Pager' # https://github.com/pujiaxin33/JXPagingView NO_SMP
pod 'PPBadgeView' # https://github.com/jkpang/PPBadgeView iOS自定义Badge组件, 支持UIView, UITabBarItem, UIBarButtonItem以及子类 NO_SMP
pod 'BRPickerView' # https://github.com/91renb/BRPickerView 该组件封装的是iOS中常用的选择器组件，主要包括：日期选择器、时间选择器（DatePickerView）、地址选择器（AddressPickerView）、自定义字符串选择器（StringPickerView）。支持自定义主题样式，适配深色模式，支持将选择器组件添加到指定容器视图。

包括pod的写法，最好用最新的特性，将外源作为单独的工程引入，提升App速度，因为启动速度有一条就是加载pod image的时间