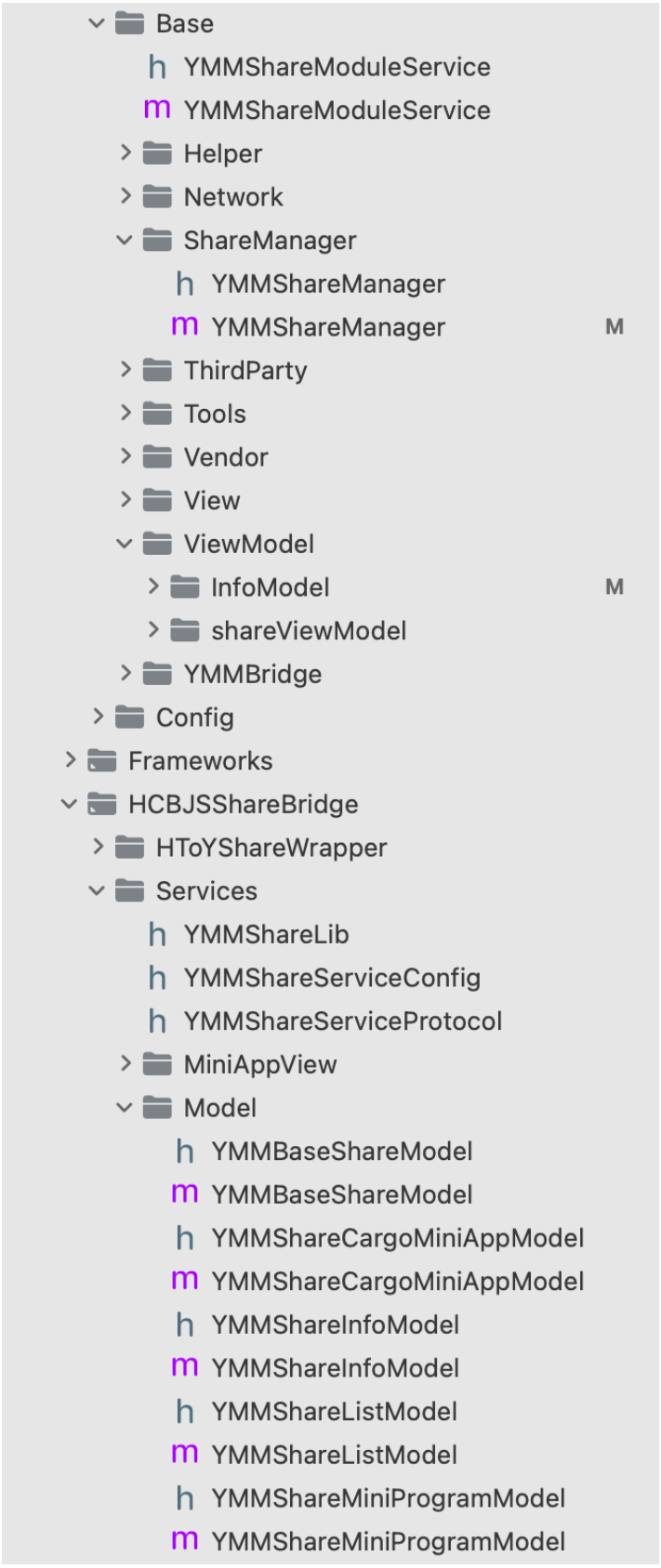
分享库重构方案

分享库现状 & 存在的问题

1. 文件目录结构冗杂，可读性差



2. YMMShareManager 这个类职责复杂，代码冗余

YMMShareManager作为分享库核心类，几乎所有分享逻辑都由其进行处理，包括但不限于判定分享平台是否安装，返回分享弹窗高度，检查相册权限等等。由于其囊括职责繁杂，导致其代码数量越来越多，代码已达2000多行，不易长期维护。

3. 分享逻辑混杂

- **直接分享和弹出分享面板后分享两种方式混杂在一起**
  - 目前分享库直接分享行为逻辑如下：

`直接分享 = 弹出分享弹窗（实际并未弹出）+ 手动点击目标分享平台`

  - 代码如下：

```
- (void)ymm_ShareStart:(YMMShareListModel *)model
              CurrentVC:(UIViewController *)currentVC
  BlockWithReturnBlock:(ReturnShareValueBlock)returnBlock
        WithErrorBlock:(ErrorShareCodeBlock)errorBlock {
    self.currentVC = currentVC;
    self.returnBlock = returnBlock;
    self.errorBlock = errorBlock;
    self.shareModel = model;

    if (self.shareModel.titles.count == 1 && (!self.shareModel.kuaishou &&
!self.shareModel.douyin)) {
        [self shareView:nil buttonTitle:[self.shareModel.titles lastObject]];
    }
    else {
        [self showView];
    }
}
```

分享库现在并未提供**直接分享给目标渠道**的调用方法。如果想直接分享给特定渠道，现在的方式是根据传入model的渠道title数量判断是否为直接分享，如果为直接分享，则手动调用点击分享渠道的回调方法`- (BOOL )shareView:(YMMShareView *)shareView buttonTitle:(NSString *)buttonTitle;`触发分享行为。

- **分享弹窗UI设置和分享内容设置行为混杂在一起**

使用者想要开启分享，需要传入一个全能型的`YMMShareListModel`。这个模型类作为传入分享数据的主要数据模型，结构复杂，包含各个渠道的分享内容的同时还囊括分享弹窗UI方面的配置信息，使用者需要一次性封装好这个model之后传入分享方法进行分享。

`YMMShareListMode`l数据结构如下图所示：

```objc
@interface YMMShareListModel : NSObject

@property (nonatomic, strong) NSNumber *result;
@property (nonatomic, copy)   NSString *errorMsg;
@property (nonatomic, copy)   NSString *requestId;

@property (nonatomic, strong) YMMShareInfoModel *wechat;          // 微信好友
@property (nonatomic, strong) YMMShareInfoModel *wechatFriend;    // 微信朋友圈
@property (nonatomic, strong) YMMShareInfoModel *sms;             // 短信
@property (nonatomic, strong) YMMShareInfoModel *qq;              // qq好友
@property (nonatomic, strong) YMMShareInfoModel *phone;           // 电话通知
@property (nonatomic, strong) YMMShareCargoMiniAppModel *miniApp DEPRECATED_ATTRIBUTE;// 货源小程序,!!废
@property (nonatomic, strong) YMMShareMiniProgramModel *miniProgram;  // 小程序
@property (nonatomic, strong) YMMShareInfoModel *qZone;           // qq空间
@property (nonatomic, strong) YMMShareInfoModel *saveImgModel;    // 保存图片
@property (nonatomic, strong) YMMShareInfoModel *motorcade;       // 车队成员
@property (nonatomic, strong) YMMShareInfoModel *saveVideo;       // 下载视频
@property (nonatomic, strong) YMMShareInfoModel *kuaishou;        // 快手分享
@property (nonatomic, strong) YMMShareInfoModel *douyin;          // 抖音分享

// 分享渠道弹框 标题、描述、可点击链接内容
@property (nonatomic, strong) YMMShareHomeModel *home;

// 预览内容数据
@property (nonatomic, strong) UIImage *previewImage;              // 预览图片
@property (nonatomic, copy) NSString *preImageUrl;                // 预览图片网络url
@property (nonatomic, strong) UIView *headerView;                 // 头部卡片视图
@property (nonatomic, strong) UIView *bottomView;                 // 底部卡片视图

/*
 额外参数传递
 1. shareMode: String 0: 普通分享 2: 截图分享
 场景: new_driver_cargos 分享组件点击/曝光埋点增加shareMode参数
 */
@property (nonatomic, strong) NSDictionary *otherParams;

- (NSArray *)titles;
- (NSArray *)images;

@end
```

4. 存在功能相似的类, 代码复用率低

比如`YMMShareInfoModel`和`YMMShareInfo`，两个数据模型类都是表示分享具体内容，其内部属性设置相似，功能也相似。

`YMMShareInfoModel`是调用者构建传入给分享方法，由`YMMShareManager`作为中转站，拆解后重新构建成`YMMShareInfo`传出第三方分享平台。后续可优化这部分逻辑，考虑将两个类型合并，精简代码。

5. 分享库新需求：希望能够外部注入分享渠道和信息, 通过注册添加分享平台

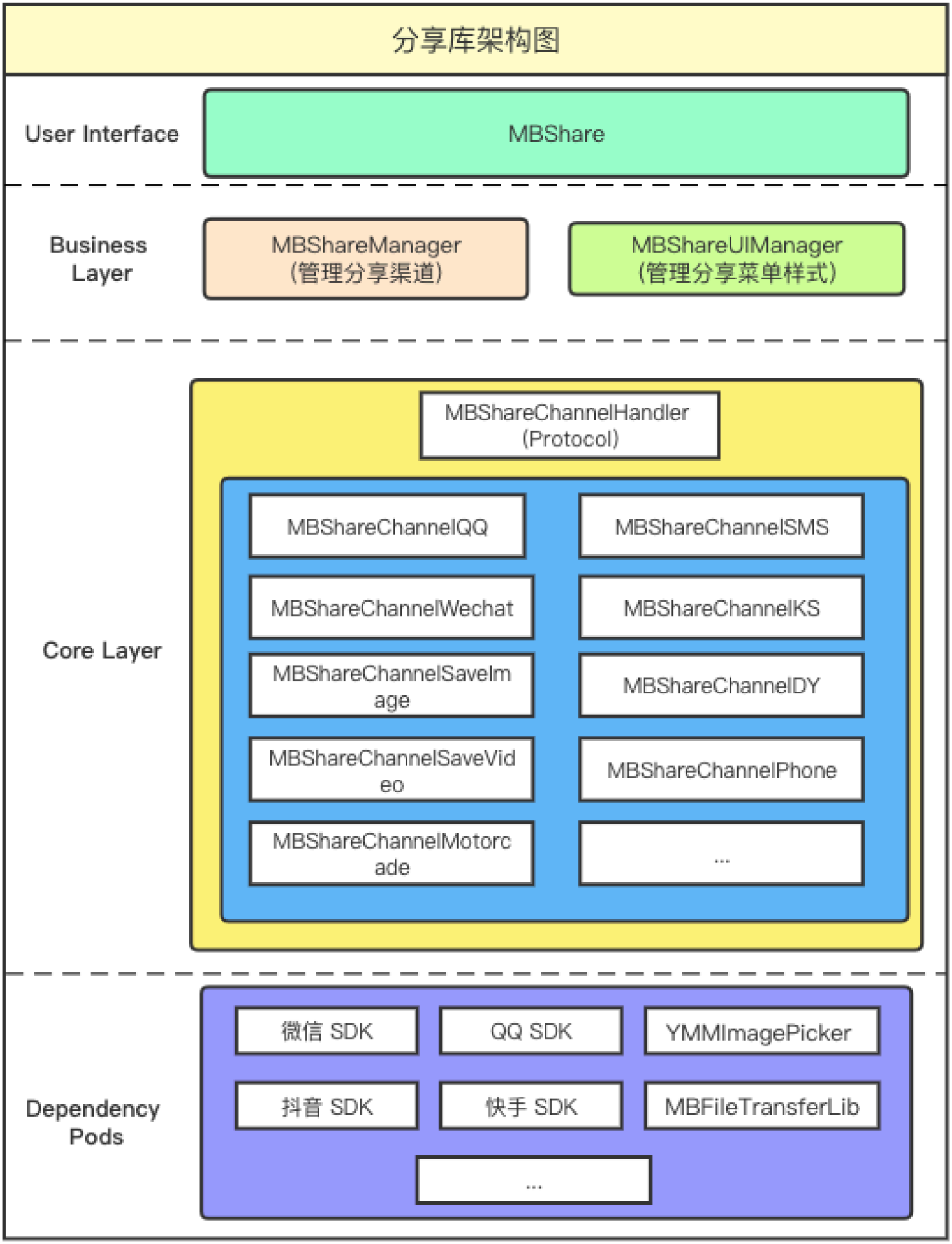
总结：

YMMShareLib 作为满帮分享的核心组件，经过长时间的内容扩展，面临着上层结构冗杂，核心类代码量庞大，代码复用率低，逻辑混杂，部分实现功能方法已废弃等一系列问题，造成了维护不便且接入新需求难度大等现象。故现阶段需要对分享库代码进行重构和优化，精简代码结构，增强代码可维护性，拓展开放性以及使用者易用性。

重构方案

1. 重构思路：

- 针对分享库上层结构杂乱的问题，重构分享库框架结构，架构图设计如下：



- 针对分享库目前代码复用率低，出现相似功能分享数据模型类问题，创建统一的分享数据模型，提高代码复用率，减少

不必要的模型拆解和重建过程；

- 针对分享库核心类`YMMShareManager`代码冗长，新建分享渠道MBShareChannelHandler协议，具体分享渠道通过实现协议自行管理自身的分享行为；
- 针对直接分享和多渠道分享逻辑混杂，弹窗视图配置和分享数据内容配置混杂的问题，将分享菜单UI配置和分享核心业务逻辑实现分离，创建MBShareUIManager类掌管分享弹窗呈现， MBShareManager类只管理核心分享行为，优化代码结构，满足单一职责原则。

2. 相关API设计

- **重构分享内容数据模型类**
  - 创建分享数据model基类：MBShareMessageModel

```
@interface MBShareMessageModel : NSObject

@property (nonatomic, copy) NSString *shareTitle;

@property (nonatomic, copy) NSString *shareContent;

@property (nonatomic, copy) NSString *description;

@property (nonatomic, strong) id thumbImage;

@end
```

  - 根据不同分享内容样式创建分享model子类：包括图片，视频，小程序，网页链接等，支持后续拓展新的样式，符合开闭原则

```
// 以图片为例
@interface MBShareImageModel : MBShareMessageModel

@property (nonatomic, retain) id shareImage;

@property (nonatomic, copy) NSArray *shareImageArray;

@end
```

> **兼容旧版本的问题：**
>
> 目前分享库存在调用者在调用分享方法时并不知道分享内容具体类型，或者具体分享内容需要从服务端请求获取，无法提前判别分享数据model类型。
>
> 解决办法：
>
> 针对此类分享场景，可以提供一个特殊子类MBShareAutotypeModel，调用者传入一个未知类型的分享内容model，分享库内部转换成具体的分享model类型。

- **创建分享渠道MBShareChannelHandler协议，各个渠道具体的分享行为由之前的ShareManager类统一管理，变为由具体的分享渠道类内部自行管理**

  所有分享渠道需要遵循MBShareChannelHandler协议，生成相应的分享渠道类，并由MBShareManager统一管理。

  各个分享平台通过实现MBShareChannelHandler协议，掌管各自的分享逻辑和平台信息，核心接口如下：

```
@protocol MBShareChannelHandler <NSObject>

@property(nonatomic, assign) MBShareChannelType shareChannelType;

@property(nonatomic, strong) MBShareChannelConfig *shareChannelConfig;

// 配置分享渠道信息
-(void)setUpWithConfig: (MBShareChannelConfig*)shareChannelConfig;

// 实现分享方法
-(void)shareWithModel:(MBShareMessageModel *)model
    withViewController:(UIViewController*)viewController
 withCompletionHandler:(MBShareCompletionHandler)completionHandler;

@optional
// 从第三方平台回调到本app的回调
- (BOOL)mbshare_handleOpenURL:(NSURL *)url;
- (BOOL)mbshare_handleOpenURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication
annotation:(id)annotation;
- (BOOL)mbshare_handleOpenURL:(NSURL *)url options:(NSDictionary*)options;
- (BOOL)mbshare_handleUniversalLink:(id)userActivity options:(NSDictionary*)options;

// 平台是否被安装
-(BOOL)isInstalled;

@end


// 分享渠道配置信息类
@interface MBShareChannelConfig: NSObject

@property(nonatomic, strong) UIImage *shareChannelIcon; //显示在分享菜单上的图标

@property(nonatomic, strong) NSString *shareChannelName; //显示在分享菜单上的名称

@property (nonatomic, copy) NSString *appID;

@property (nonatomic, copy) NSString *appSecret;

@property (nonatomic, copy) NSString *universalLink;
```

```
@end
```

- **创建MBShareManager 作为一个全局的掌控/调度者类，核心接口如下：**
  - 管理分享渠道：注册分享渠道/获取渠道类/添加自定义渠道/移除指定渠道接口：

```
// 返回当前所有分享渠道
@property(nonatomic,readonly,strong) NSArray * channelTypeArray;

//注册分享平台信息
- (id<MBShareChannelHandler>)registerChannel:(MBShareChannelType)shareChannelType
withConfiguration:(MBShareChannelConfig*)config;

// 获得对应的分享渠道类型的id<MBShareChannelHandler>渠道类
-(id<MBShareChannelHandler>)platformProviderWithChannelType:
(MBShareChannelType)shareChannelType ;

// 删除指定分享渠道
-(void)removeShareChannelWithChannelType:(MBShareChannelType)shareChannelType;
```

  - 核心分享方法接口：

```
// 分享给指定平台
- (void)shareToPlatform:(MBShareChannelType)shareChannelType
                  model:(MBShareMessageModel *)shareModel
  currentViewController:(UIViewController *)currentViewController
             completion:(MBShareCompletionHandler)completion;

- (void)shareToPlatform:(MBShareChannelType)shareChannelType
                  model:(MBShareMessageModel *)shareModel
             completion:(MBShareCompletionHandler)completion;

typedef void (^MBShareCompletionHandler)(id shareResponse,NSError *error);
```

  - 查询平台是否安装接口：

```
-(BOOL)isInstalled:(MBSharePlatformType)platformType;
```

  - 统一处理回调方法接口：

```
-(BOOL)handleOpenURL:url sourceApplication:sourceApplication annotation:annotation;

-(BOOL)continueUserActivity:(NSUserActivity*)userActivity restorationHandler:(void(^)
(NSArray* __nullable restorableObjects))restorationHandler;

-(BOOL)handleOpenURL:(NSURL *)url;
```

- **将分享弹窗视图呈现和核心分享业务逻辑分离**

  创建MBShareUIManager类，掌管分享弹窗呈现

  - 配置分享菜单样式：通过设置MBShareUIConfig的属性自定义分享弹窗的样式

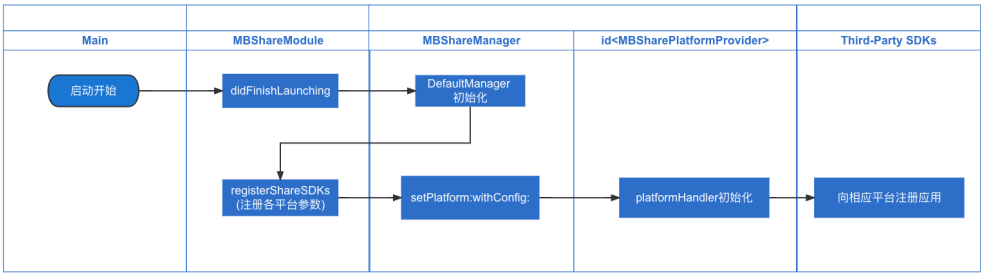
```
@interface MBShareUIConfig: NSObject
@property (nonatomic, strong) UIView *headerView;                    // 头部卡片视图
@property (nonatomic, strong) UIView *bottomView;                    // 底部卡片视图
...
@end
```

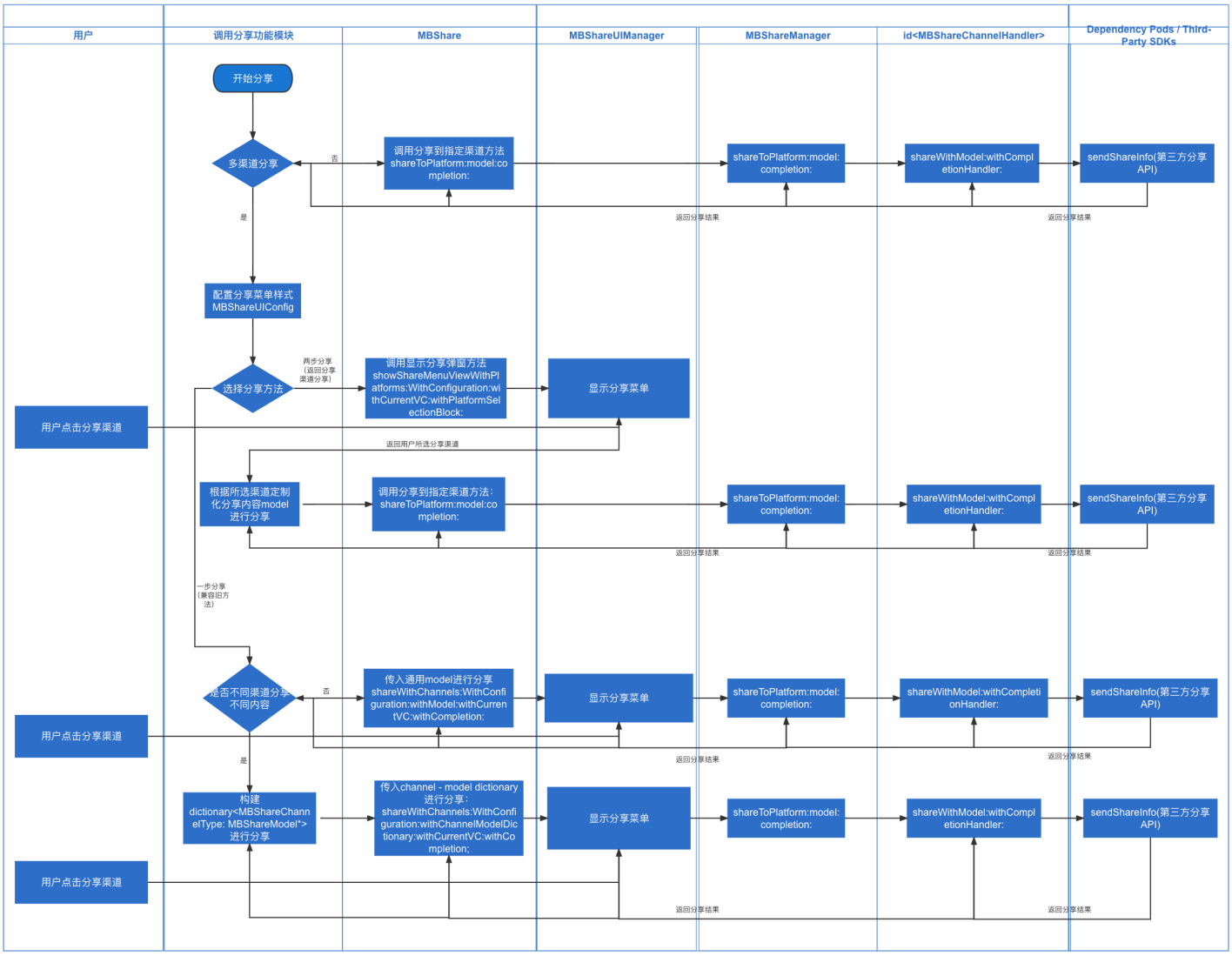
  - 弹起分享菜单窗口

```
// 添加新方法：弹出分享弹窗，并将选择的分享渠道作为参数返回，使用者可以根据分享渠道调用MBShareManager的直接分享方法进行分享；
+ (void)showShareMenuViewWithPredefinedChannels:(NSArray*)channels WithConfiguration:(MBShareUIConfig *)config  withChannelSelectionBlock:^(MBShareChannelType shareChannelType);

// 兼容老方法,实现一步分享，不同平台分享不同内容
+ (void)shareWithPredefinedChannels:(NSArray*)preDefinedPlatforms
                  WithConfiguration:(MBShareUIConfig *)config
         withChannelModelDictionary: (NSDictionary*)PlatformModelDict
                     withcompletion:(MBShareCompletionHandler)completion;

// 兼容老方法，实现一步分享，不同平台分享同样内容
+ (void)shareWithPredefinedChannels:(NSArray*)preDefinedPlatforms
                  WithConfiguration:(MBShareUIConfig *)config
                          withModel:(MBShareMessageModel *)messageObject
                     withcompletion:(MBShareCompletionHandler)completion;
```

```
// 调用新方法分享代码示例：
[MBShareUIManager showShareMenuViewWithPredefinedChannels:(NSArray*)channels
                                          WithConfiguration:(MBShareUIConfig *)config
                                withChannelSelectionBlock:^(MBShareChannelType shareChannelType){

   [MBShareManager defaultMananger] shareToPlatform:(MBShareChannelType)shareChannelType
                                                  model:(MBShareMessageModel *)shareModel
                                      currentViewController:(UIViewController *)currentViewController
                                                 completion:(MBShareCompletionHandler)completion;

}];
```

3. 分享流程图设计


Main
MBShareModule
MBShareManager
id<MBSharePlatformProvider>
Third-Party SDKs
启动开始
didFinishLaunching
DefaultManager 初始化
registerShareSDKs (注册各平台参数)
setPlatform:withConfig:
platformHandler初始化
向相应平台注册应用


用户
调用分享功能模块
MBShare
MBShareUIManager
MBShareManager
id<MBShareChannelHandler>
Dependency Pods / Third-Party SDKs
开始分享
多渠道分享
否
是
调用分享到指定渠道方法 shareToPlatform:model:completion:
shareToPlatform:model:completion:
shareWithModel:withCompletionHandler:
sendShareInfo(第三方分享API)
返回分享结果
返回分享结果
配置分享菜单样式 MBShareUIConfig
选择分享方法
两步分享 (返回分享渠道分享)
调用显示分享弹窗方法 showShareMenuViewWithPlatforms:WithConfiguration:withCurrentVC:withPlatformSelectionBlock:
显示分享菜单
用户点击分享渠道
返回用户所选分享渠道
根据所选渠道定制化分享内容model进行分享
调用分享到指定渠道方法：shareToPlatform:model:completion:
shareToPlatform:model:completion:
shareWithModel:withCompletionHandler:
sendShareInfo(第三方分享API)
返回分享结果
返回分享结果
一步分享 (兼容旧方法)
是否不同渠道分享不同内容
否
传入通用model进行分享 shareWithChannels:WithConfiguration:withModel:withCurrentVC:withCompletion:
显示分享菜单
shareToPlatform:model:completion:
shareWithModel:withCompletionHandler:
sendShareInfo(第三方分享API)
用户点击分享渠道
返回分享结果
返回分享结果
是
构建 dictionary<MBShareChannelType: MBShareModel*> 进行分享
传入channel - model dictionary 进行分享：shareWithChannels:WithConfiguration:withChannelModelDictionary:withCurrentVC:withCompletion;
显示分享菜单
shareToPlatform:model:completion:
shareWithModel:withCompletionHandler:
sendShareInfo(第三方分享API)
用户点击分享渠道
返回分享结果
返回分享结果