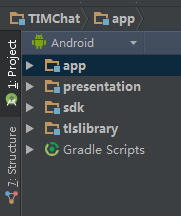
# 工程结构



整个工程包含四个模块，分别是app,presentation,sdk,tlslibrary

**App:**

用户上层界面，界面实现下层接口，用户数据结构，依赖presentation

**Presentation：**

根据demo需求组装模块化的逻辑供上层调用，通过接口回调上层的实现，依赖sdk

**Sdk:**

包含所有的sdk

**Tlslibrary：**

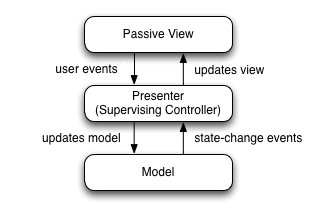
集成tls登录模块，属于工程的依赖项



## MVP

整个工程采用了MVP的设计模式，主要目的是将逻辑和activity等界面层剥离。MVP模式主要分为四个要素：

1. View：负责绘制UI元素、与用户进行交互（主要是activity，在demo的app模块）实现view interface
2. View interface：需要view实现的接口，view通过view interface与Presenter交互，降低耦合（在demo的presentation模块）
3. Model：负责数据存储（demo的app模块和SDK中）
4. Presenter：负责用户交互逻辑（demo的presentation中）



# 使用方式

* TIMDemo本身就是一个完整的APP，可以在com.tencent.qcloud.sdk.Constant中修改demo里的appid和accounttype直接使用。
* TIMDemo在IMSDK的基础上根据上层业务进行了一次简单的封装，业务逻辑相近的功能可以直接使用该层封装。我们假设用户会使用自己的数据结构，所以这一层没有任何自己的数据结构，而是直接将IMSDK的数据通过接口回调给上层。

用户可以将presentation module和sdk module直接引用到工程，调用里面的方法。其中重要的类的功能如下：

1. business.InitBusiness 初始化IMSDK，在使用IMSDK前调用
2. business.LoginBusiness IMSDK登录和登出
3. event.FriendshipEvent 好友关系链通知系统，使用前必须初始化，所有需要好友关系链通知的类可以监听这里
4. event.GroupEvent 群关系链通知系统，使用前必须初始化，所有关系群关系变化的类可以监听这里
5. event.MessageEvent 消息通知系统
6. presenter.ChatPresenter 聊天界面逻辑，包括发送消息，监听新消息。和viewfeatures.ChatView配合使用
7. presenter.ConversationPresenter 会话列表逻辑，包括获取最近会话列表，监听新消息。和viewfuatures.ConversationView配合使用
8. presenter.FriendshipManagerPresenter 好友关系链管理和用户资料相关逻辑。和FriendshipManageView,FriendshipMessageView,FriendInfoView配合使用
9. presenter.GroupManagerPresenter 群关系链和资料相关逻辑。和GroupInfoView,GroupManageMessageView,Group,GroupManageView配合使用

* TIMDemo演示了大量IMSDK的接口使用方式，另外提供了一些通用的界面控件，用户可以从代码中自取所需。

1. Customview.ChatInput 聊天界面输入控件
2. Customview.CameraPrview 视频预览控件
3. Model.Message 消息数据抽象。TextMessage,ImageMessage,VoiceMessage,VideoMessage均继承自该类，可以使用MessageFactory解析
4. Model.Conversation会话类的数据抽象，NomalConversation（群和好友会话），FriendshipConversation（好友关系链变更会话），GroupManageConversation（群关系链变更会话）均继承自此

# 关键模块组成和典型应用方式

## 好友关系链

TODO 这里介绍IMSDK本地缓存的好友关系链使用（包括数据存储，获取，写入，通知处理，用户自己数据同步，界面更新）

## 群关系链

TODO 这里介绍IMSDK本地缓存的群管理使用（包括数据存储，获取，写入，通知处理，用户自己数据同步，界面更新）

## 消息模块

TODO 这里介绍消息系统使用（包括消息收发，消息通知，界面更新）