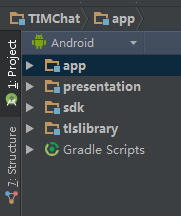
# 工程结构



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：**

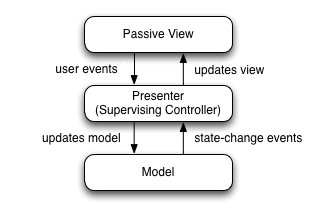
集成tls登录模块，属于工程的依赖项



## 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。使用方式：直接在工程中引用presentation和sdk模块。
* 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，可以自由拷贝和使用这些代码。

# 关键模块和典型应用方式

## 好友关系链

Demo使用了SDK中本地关系链缓存。主要使用的功能：

1. 同步获取好友关系链数据
2. 注册关系链变更的监听

使用方式

1. 在IMSDK初始化之前设置好友关系链缓存参数和监听器，并开启缓存开关，详见FriendshipEvent.init
2. 监听关系链变更，并将变更的数据通知到其他地方
3. 同步获取SDK中关系链数据，详见FriendshipInfo

## 群关系链

Demo使用了SDK中本地群关系缓存。主要使用的功能：

1. 同步获取群相关数据
2. 注册群关系变更的监听

使用方式

1. 在IMSDK初始化之前设置群关系链缓存参数和监听器，并开启缓存开关，详见GroupEvent.init
2. 监听关系链变更，并将变更的数据通知到其他地方
3. 同步获取SDK中群关系数据，详见GroupInfo

## 消息模块

消息模块包括：在线消息监听，发消息，消息内容解析，消息触发的界面刷新。

消息监听详见：MessageEvent。

发消息详见：Message及各个继承类，实质是生成本地的消息然后发送，SDK会自己修改消息的状态（发送中，已发送，发送失败等）。

消息内容解析详见：MessageFactory。

消息触发界面刷新：聊天界面收到消息通知后根据Message的继承类显示界面，会话界面收到消息通知更新最后一条消息。