架构设计

后端

使用`Java`语言进行开发，使用`SprintBoot`进行依赖注入、切面等的管理，使用`Netty`进行`Http`及`Websocket`接口的监听并处理请求，数据库使用`mongodb`进行信息的持久化存储，使用`redis`作会话缓存；向前端提供用户信息、会话信息、联系人信息、群组信息、动态信息等`Http`接口的服务，及实时会话通知的`Websocket`接口服务；使用`TimeLine`模型及扩散写进行消息的同步及存储。

数据库设计

User

> 存储用户信息

id	name	avatar	password	friends	timeline
用户ID	用户名字	用户头像	密码	好友信息	同步timeline

```
friend: {
  nickname:"昵称",
  timeline:"会话ID",
  id:"好友ID"
}
```

Group

> 存储群组信息

id	name	members	timeline
群组ID	群组名字	群组成员id列表	会话ID

Message

id	content	content_type	message_type	time	from	to
消息ID	消息内容	内容类型	消息类型	消息时间	发送者ID	接受者ID

```
# `text`指文本消息 `audio`指音频消息 `video`指视频消息 `location`指地图定位消息
content_type: 'text' | 'audio' | 'video' | 'location'
# `single`指单聊 `group`指群聊 `application`指好友申请 `invite`指邀请加入群聊 `confirm`指同意好友请求
type: 'single' | 'group' | 'application' | 'invite' | 'confirm'
```

TimelineSaved

> 消息存储库，对消息进行持久化存储

id	messages
会话ID	会话消息ID

TimelineSync

> 消息同步库，对每个用户进行消息同步

id	messages
会话ID	会话消息ID

Discover

> 存储用户发布的动态

id	sender	content	images	video	time	likes	replies
动态ID	发送者ID	文字内容	图片列表	视频	时间	点赞用户ID	评论

接口设计

采用`Restful`风格

用户

> 使用Cookie进行鉴权

请求方法	意义	链接
GET	获取单个用户信息	`/user`
POST	注册	`/user/register`
POST	登录	`/user/login`
POST	登出	`/user/logout`
POST	确认添加好友	`/addfriend`
PUT	更新用户信息	`/user`
POST	删除好友	`/deletefriend`

群组

请求方法	意义	链接
POST	创建群组	`/group`
POST	邀请加入群聊	`/group/invite`
POST	退出群聊	`/group/exit`

消息

请求方法	意义	链接
GET	获取同步消息	`/timelinesync/:id`
GET	获取会话漫游消息	`/timelinesaved/:id`
POST	发送消息	`/message`

动态

请求方法	意义	链接
GET	获取动态	`/discover`
POST	发布动态	`/discover`
POST	点赞动态	`/discover/like`
POST	取消点赞动态	`/discover/unlike`
POST	回复动态	`/discover/reply`

文件

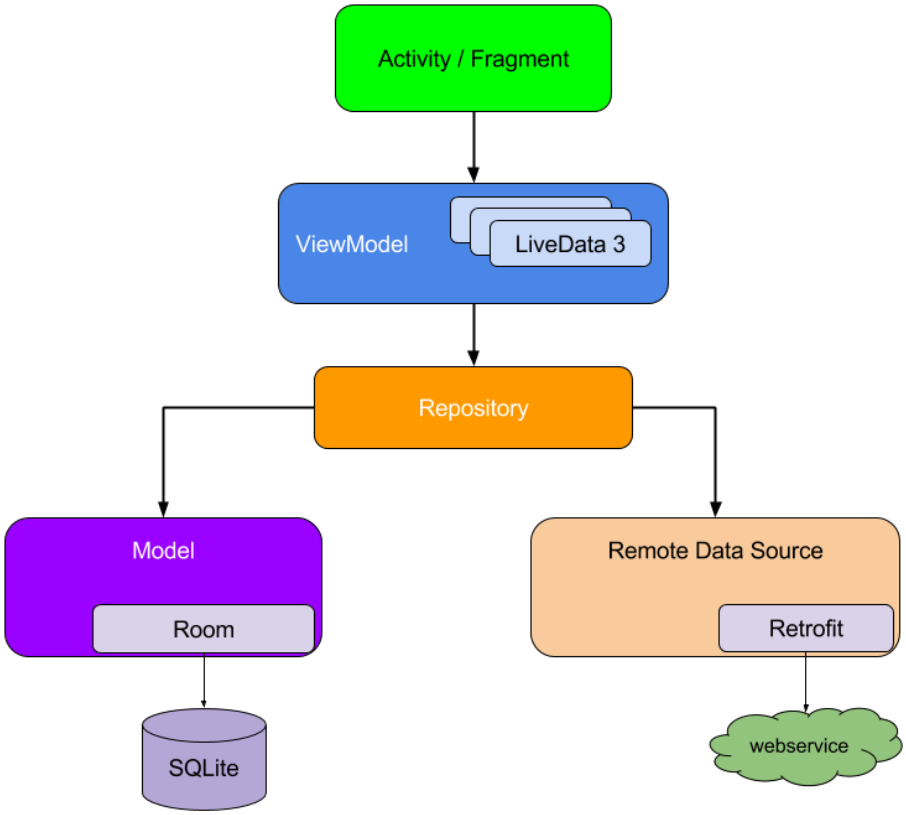
请求方法	意义	链接
GET	获取文件	`/upload/:id`
POST	上传文件	`/upload`

客户端

使用`Java`语言进行开发，主要使用`MVVM`的架构，使用`Activity/Fragment`进行`UI`展示，使用`LiveData`及`ViewModel`进行数据界面绑定及`Room`进行缓存，使用`Retrofit`进行`Http`请求，`Scarlet`进行`Websocket`连接，使用`Hilt`进行依赖注入。

MVVM架构设计

用户、群组、消息、动态等实体均以`ViewModel`形式供`UI`界面包括`Activity/Fragment`进行使用，界面交互事件及请求事件由`ViewModel`派发给`Repository`进行处理，`Repository`进行数据的缓存管理，请求处理等，请求处理使用`NetworkBoundService`进行处理，将结果转为`LiveData`由`UI`进行控制，整体架构如图



请求设计

请求包括`HTTP`请求和`WebSocket`请求，`Http`请求使用`OkHttp`进行管理，同时使用`Adapter`转为`LiveData`进行处理，`WebSocket`请求使用`Scarlet`进行管理，同时使用`rxjava`转为`observer`进行处理

界面设计

整体界面设计与微信类似，下方以`BottomNavigationMenu`进行导航，分别用消息，好友，发现及设置四个`Fragment`来进行`UI`展示和交互，消息`Fragment`可进入与好友或群组的聊天主界面，好友界面可以进行添加好友，创建群组，好友管理等操作，发现界面可以进入朋友圈，进行浏览朋友圈以及发朋友圈的操作，设置界面可以进行账号信息的修改和管理

功能实现

> 详细分工请见项目过程管理文档

后端

接口逻辑实现

用户

- 注册：在`User`中新增用户，对密码进行加密，新建`timeline`，保存在`User`,`TimeLineSync`中【张智】
- 登录：验证微信好、密码，返回加密cookie，并在`Redis`中加入缓存【张智】
- 登出：在`Redis`中清除该用户【张智】
- 确认添加好友：在`User`中双方`friends`中均新增好友信息，新建`timeline`，保存在`friend`，`TimeLineSaved`中【张智】

群组

- 新建群组：根据名字，好友新建`Group`群聊即可【张智】
- 邀请好友入群：更新`Group`即可【王澳】
- 退出群聊：更新`Group`即可【王澳】

消息

- 获取同步消息：若用户在线，则在扩散写时通过`websocket`发送消息，发送成功后，若未在线，则登录后获取同步消息，同步消息使用`time`作为检验，在客户端保存最新同步`time`，每次获取同步消息时，仅获取`time`之后的消息，需要考虑设置`timeline`过期时间后删除【王澳】
- 发送消息：消息类型包括单聊，群聊，好友申请，内容类型包括音频、视频、文本、地图，消息发送成功后，扩散写进入接受者`timeline`，单聊和好友申请直接写入`friends`中即可，群聊需要写入所有群组成员`timeline`中【王澳】
- 消息漫游：跨端时需进行消息漫游，考虑消息会缓存在客户端，每次检查同步`time`，若首次不存在`time`，客户端调用获取消息存储库中所有消息，直接返回所有消息即可【王澳】
- 消息删除：实现软删除，即服务端进行数据保留，客户端进行软删除【王澳】

动态

- 发布动态：在`Discover`中根据信息创建新动态【张智】
- 获取动态：按时间排序获取好友的所有动态【张智】
- 点赞与取消点赞动态：更新动态`Discover`即可【王澳】
- 回复动态：更新`Discover`即可【王澳】

客户端

> 分工：王澳

消息功能实现

进行消息交互时，可以选择发送文本信息，图片信息以及视频，地图定位信息，各种信息均使用文本进行内容的存储，若为图片或视频，则存储`URL`，若为地图定位，则存储经纬度，在显示时只需根据不同消息类型对文本进行不同的渲染即可，地图消息展示使用高德的地图`SDK`，图片及视频消息先上传到服务器后直接发送`URL`信息