**《基于Android的即时通讯系统》**

**软件设计说明书**

学号： 201922172014406

姓名： 杨鹏

专业： 软件工程

**《基于Android的即时通讯系统》软件设计说明书**

**一、问题定义实习**

系统主要是提供交流服务，主要的功能包括文字消息、语音消息、语音聊天和视频聊天这四个通讯功能。整个系统拟解决用户登录、注册、退出、忘记密码、添加好友、好友管理、聊天记录管理、聊天消息通知以及四种聊天方式等功能项目。

**二、用例描述**

**注册用例：**

描述：为新用户提供注册账号的服务。

参与者：想要使用系统的新用户。

前置条件：无。

后置条件：用户拥有已经注册的账号。

基本流程：

1、新用户进入注册界面；

2、输入自己的手机号和密码；

3、点击注册按钮；

4、注册成功。

替代流程：

A1、手机号格式不正确：①系统提示“手机号格式不正确”；②用户选择重新输入手机号；③手机号格式还不正确，跳转到①。

A2、密码不符合安全要求：①系统提示“密码安全强度过低，建议数字和大小写字母混合使用”；②用户选择重新输入密码；③密码仍不符合要求，跳转到①。

**登录用例：**

描述：提供用户进入系统的入口。

参与者：使用系统的用户。

前置条件：用户已经注册拥有账号。

后置条件：用户进入系统。

基本流程：

1、用户进入登录界面；

2、输入自己的账号和密码；

3、点击登录按钮；

4、系统验证账号密码；

5、登录成功。

替代流程：

A1、账号不存在：①系统提示“输入的账号不存在”；②用户选择重新输入账号，跳转到基本流程；③用户选择注册，跳转到注册用例。

A2、密码错误：①系统提示“密码错误”；②用户选择重新输入密码，跳转到基本流程；③用户选择忘记密码，跳转到忘记密码用例。

**忘记密码用例：**

描述：用户忘记密码时，为其提供修改密码的服务。

参与者：使用系统的用户。

前置条件：用户忘记自己账号的密码。

后置条件：修改账号密码。

基本流程：

1、用户点击登录界面中的忘记密码按钮；

2、用户输入自己的手机号，点击发送验证码；

3、用户收到验证码短信，并填入验证码；

4、验证码正确，输入新的密码并确认；

5、修改成功。

替代流程：

A1、验证码不正确：①系统提示“验证码不正确”；②用户重新输入验证码；③验证码仍不正确，跳转到①。

A2、密码不符合安全要求：①系统提示“密码安全强度过低，建议数字和大小写字母混合使用”；②用户选择重新输入密码；③密码仍不符合要求，跳转到①。

**添加好友用例：**

描述：为用户提供添加好友的服务。

参与者：使用系统的用户。

前置条件：用户成功登录进入系统。

后置条件：发送添加好友请求成功。

基本流程：

1、用户进入添加好友界面；

2、输入对方的手机号；

3、点击发送添加请求按钮；

4、等待对方验证通过。

替代流程：

A1、用户不存在：①系统提示“该用户不存在”；②用户重新输入手机号；③用户仍不存在，跳转到①。

**好友请求通知用例：**

描述：提醒用户有其他用户发送过来好友请求，请及时处理。

参与者：使用系统的用户。

前置条件：用户成功登录进入系统，并收到其他用户发送的好友请求。

后置条件：处理好友请求完毕，不再显示通知提醒。

基本流程：

1、用户进入好友请求通知界面；

2、查看好友请求的详细信息；

3、点击同意或拒绝按钮；

4、通知提醒消失。

替代流程：无。

**文字、语音消息用例：**

描述：为用户提供发送文字或者语音消息聊天的服务。

参与者：使用系统的用户。

前置条件：用户已经成功登录系统并添加该联系人为好友。

后置条件：发送消息成功。

基本流程：

1、点击联系人，进入联系人界面；

2、选择文字或者语音消息；

3、输入消息内容；

4、点击发送按钮；

5、发送成功。

替代流程：

A1、消息发送失败：①系统提示“消息发送失败，请检查网络”；②用户选择重新输入消息内容，点击发送；③消息发送失败，跳转到①。

A2、该用户不是您的好友：①系统提示“该用户不是您的好友，不能发送消息”；②用户选择重新添加该联系人为好友；③用户选择重新输入并发送，跳转到①。

**语音、视频通话用例：**

描述：为用户提供语音或者视频通话的服务。

参与者：使用系统的用户。

前置条件：用户已经成功登录系统并添加该联系人为好友。

后置条件：发送语音或视频通话请求成功，等待对方接受。

基本流程：

1、点击联系人，进入联系人界面；

2、点击语音通话或视频通话按钮；

3、系统询问是否确定要进行通话，点击确定；

4、发送请求成功，等待对方接受。

替代流程：

A1、网络异常：①系统提示“网络异常，通话中断”；②用户检查网络，并重新发送通话请求；③网络异常，跳转到①。

**删除好友用例：**

描述：为用户提供删除好友的服务。

参与者：使用系统的用户。

前置条件：用户已经成功登录系统并添加该联系人为好友。

后置条件：删除好友成功。

基本流程：

1、点击该联系人，进入联系人界面；

2、点击界面右上角，进入选项；

3、点击删除好友按钮；

4、系统询问是否确定删除该好友；

5、点击确定，删除好友成功。

替代流程：无。

**修改备注用例：**

描述：为用户提供修改好友备注的服务。

参与者：使用系统的用户。

前置条件：用户已经成功登录系统并添加该联系人为好友。

后置条件：修改备注成功。

基本流程：

1、点击该联系人，进入联系人界面；

2、点击界面右上角，进入选项；

3、点击备注按钮；

4、输入备注内容，点击确定；

5、修改备注成功。

替代流程：无。

**删除会话用例：**

描述：为用户提供删除已有的会话窗口以及聊天内容。

参与者：使用系统的用户。

前置条件：用户已经成功登录系统并存在会话窗口。

后置条件：删除会话和聊天记录成功。

基本流程：

1、用户点击进入会话界面；

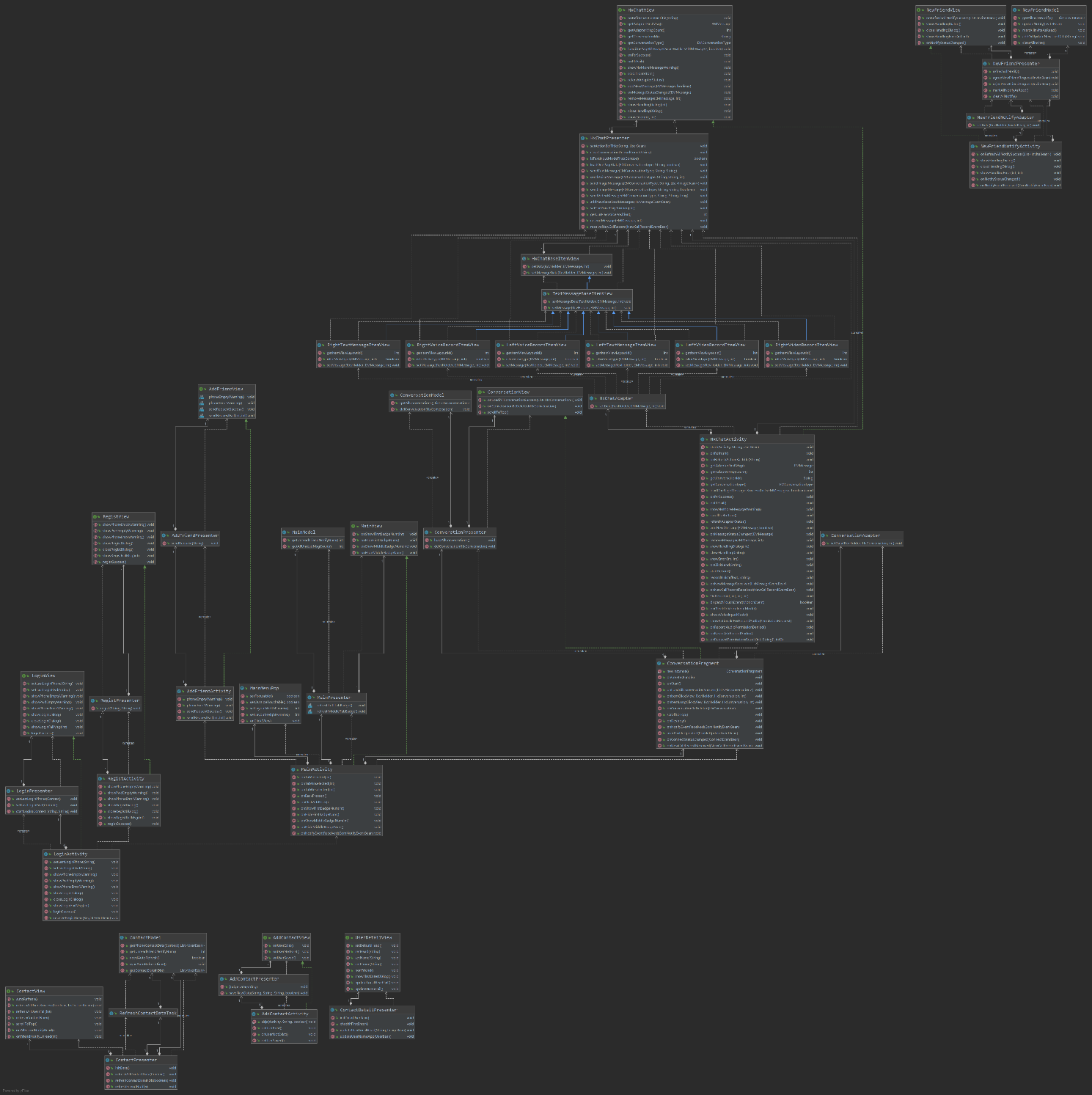
2、长按会话窗口，点击删除；

3、系统询问删除会话时聊天记录也一并删除，是否继续；

4、点击继续，删除成功。

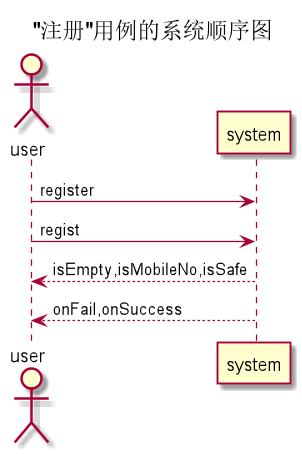
替代流程：无。

**三、领域模型构建**

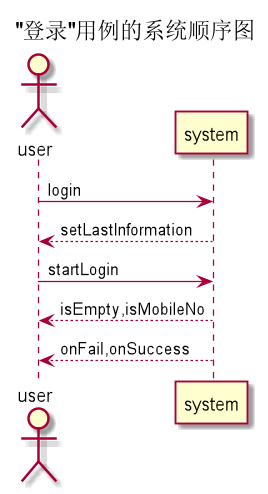


**四、系统顺序图构建**

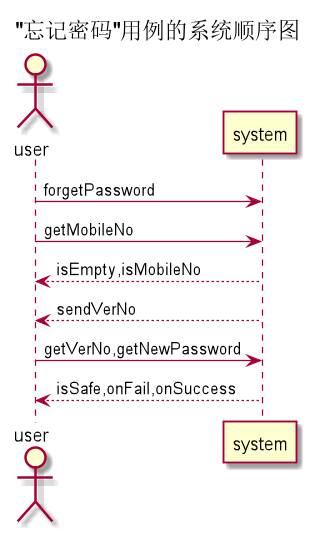
**1、"注册"用例的系统顺序图**



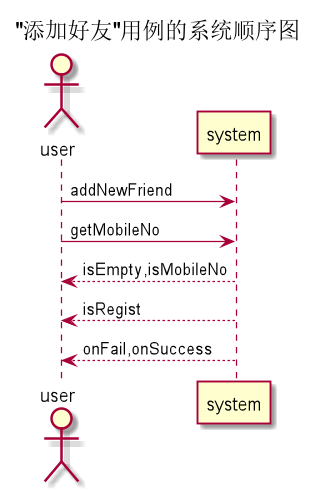
**2、“登录”用例的系统顺序图**



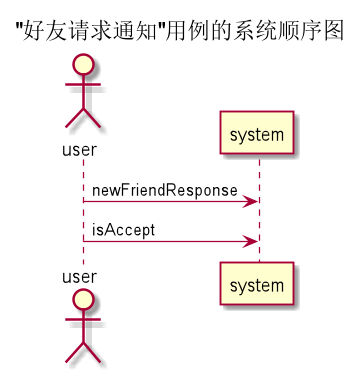
**3、“忘记密码”用例的系统顺序图**



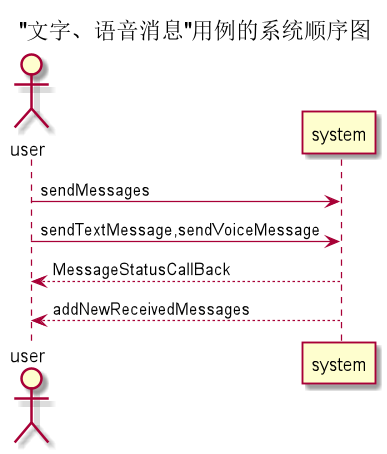
**4、“添加好友”用例的系统顺序图**



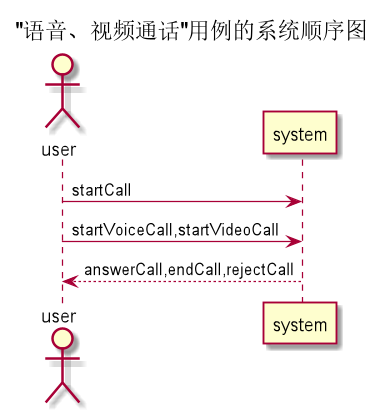
**5、“好友请求通知”用例的系统顺序图**



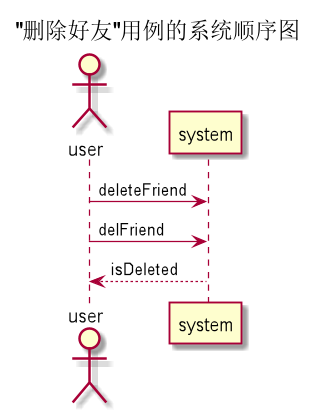
**6、“文字、语音消息”用例的系统顺序图**



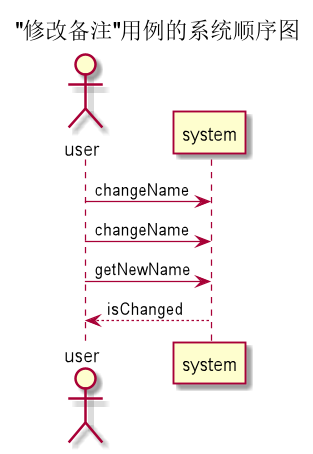
**7、“语音、视频通话”用例的系统顺序图**



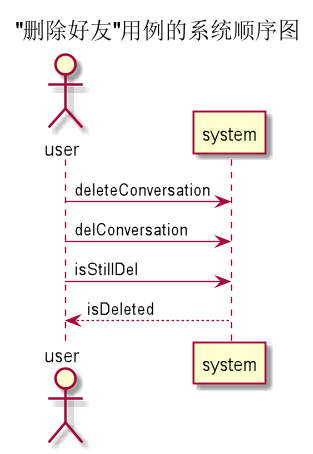
**8、“删除好友”用例的系统顺序图**



**9、“修改备注”用例的系统顺序图**



**10、“删除会话”用例的系统顺序图**



**五、契约构建**

**契约1：register**

操作：regist(String phone, String password)

交叉引用：用例：注册。

前置条件：无。

后置条件：

①创建RegistPresenter的实例registp。

②将registp关联到当前的RegistActivity。

③将registp关联到当前的RegistView。

④registp.phone和registp.password分别被赋值为phone和password。

**契约2：login**

操作：login(String phone, String password)

交叉引用：用例：登录。

前置条件：正在进行中的登录。

后置条件：

①创建LoginPresenter的实例login。

②将login关联到当前的LoginActivity。

③将login关联到当前的LoginView。

④ login.phone和login.password分别被赋值为phone和password。

**契约3：forgetPassword**

操作：forgetPassword (String password)

交叉引用：用例：忘记密码。

前置条件：修改密码找回账号。

后置条件：

①创建User的实例user。

②user.password被赋值为password。

**契约4：addNewFriend**

操作：addNewFriend(String phone)

交叉引用：用例：添加好友。

前置条件：成功登录进入系统，添加联系人。

后置条件：

①创建User实例user。

②user.phone被赋值为phone。

③将user关联到Response类。

**契约5：newFriendResponse**

操作：newFriendResponse()

交叉引用：用例：好友请求通知。

前置条件：接收到好友请求通知。

后置条件：

①创建Response类的实例res。

②res.status被赋值为1。

③将res关联到User类。

**契约6：messages**

操作：sendMessages(MessageType conID, Message context)

交叉引用：用例：文字、语音消息。

前置条件：正在进行中的消息聊天。

后置条件：

①创建ChatPresenter的实例chat。

②chat.messageType和chat.message被赋值为conID和context。

③将chat关联到User、ChatActivity和ChatView。

**契约7：call**

操作：startCall(String phone)

交叉引用：用例：语音、视频通话。

前置条件：正在进行中的通话。

后置条件：

①创建CallPresenter的实例call。

②call.phone被赋值为phone。

③将call关联到User、CallActivity和CallView。

**契约8：deleteFriend**

操作：delFriend(String phone)

交叉引用：用例：删除好友。

前置条件：成功登录进入系统并添加联系人为好友。

后置条件：

①创建一个Friend类的实例friend。

②friend.phone被赋值为phone。

③将friend关联到Contact类。

**契约9：changeName**

操作：changeName(String name)

交叉引用：用例：修改备注。

前置条件：成功登录进入系统并添加联系人为好友。

后置条件：

①创建一个Friend类的实例friend。

②friend.name被赋值为name。

③将friend关联到Contact类。

**契约10：deleteConversation**

操作：delConversation (Conversation conversation)

交叉引用：用例：删除会话。

前置条件：成功登录系统并与好友进行聊天。

后置条件：

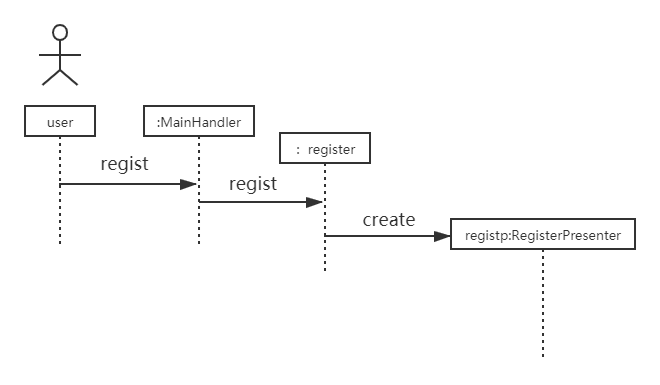
①创建Conversation的实例conversation。

②conversation.status被赋值为1。

③将conversation关联到MainPresenter。

**六、顺序图与设计类图构建和完善**

**顺序图1：**



**类描述：**

1、register:负责系统的注册业务。

属性：

private RegistPresenter mPresenter;

private FlatEditText mEdPhone;

private FlatEditText mEdPwd;

private ProgressDialog mDialog;

方法：

setContentViewId()；

initUI()； 初始化界面

showRegistDialog()；

showPhoneEmptyWarning()； 手机号为空异常警告

showPwdEmptyWarning()； 密码为空异常警告

showPhoneErrorWarning()； 手机号格式不规范警告

2、RegisterPresenter：注册业务中的表示层。

属性：

private RegistView mRegistView;

方法：

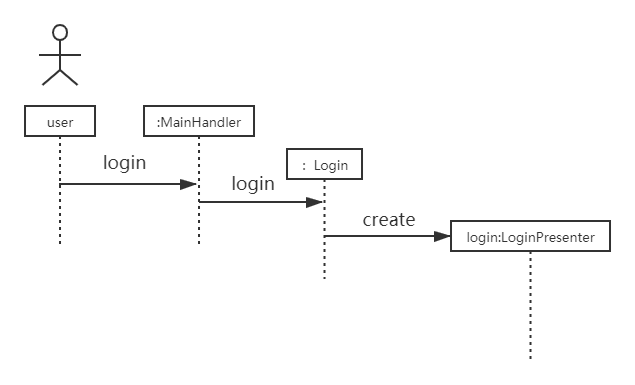
public RegistPresenter(RegistView registView)； 构造方法

public void regist(String phone, String pwd)； 注册方法

public void onFail(int status, int errorMsgResId)；注册失败

public void onSuccess(Object o)； 注册成功

**顺序图2：**



**类描述：**

1、Login：负责系统的登录业务。

属性：

private LoginPresenter mPresenter;  
private FlatEditText mEdPhone;  
private FlatEditText mEdPwd;  
private ProgressDialog mDialog;

方法：

protected int setContentViewId()；

protected void initData()； 初始化数据

public void setLastLoginPhone(String phone)； 初始化上次登录的账号信息  
public void setLastLoginPwd(String pwd)； 初始化上次登录的账号密码  
public void showPhoneEmptyWarning()； 手机号为空警告  
public void showPwdEmptyWarning()； 密码为空警告  
public void showPhoneErrorWarning()； 手机号格式不规范警告  
public void showLoginDialog()； 显示提示信息

2、LoginPresenter：登录业务的表示层。

属性：

private LoginView mLoginView;

方法：

public LoginPresenter(LoginView loginView)； 构造方法

public void setLastLoginPhone(Context context)；

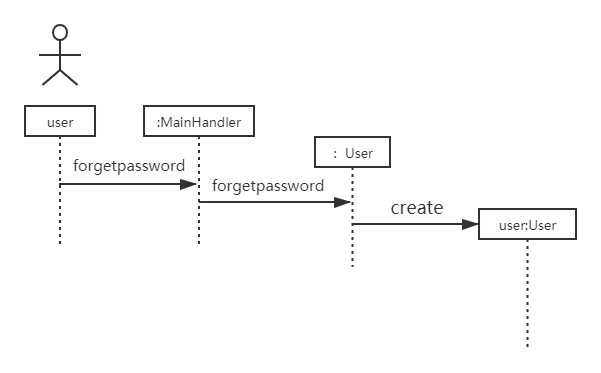
public void setLastLoginPwd(Context context)；

public void startLogin(final Context context, final String phone, final String pwd)； 登录方法

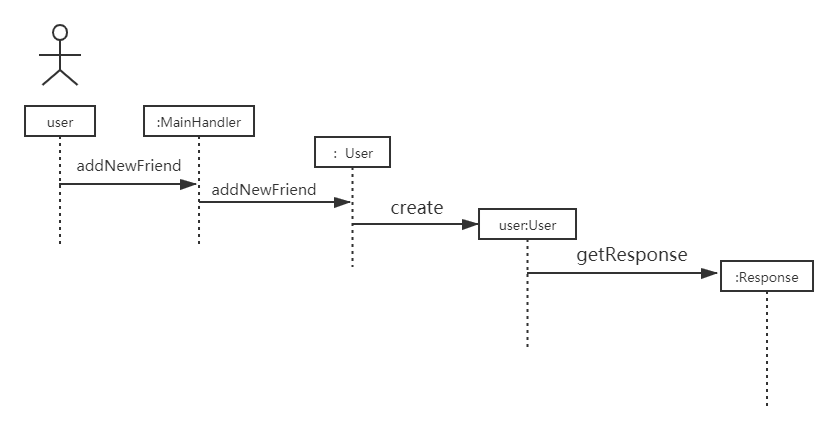
public void onFail(int status, int errorMsgResId)； 登录成功

public void onSuccess(Object o)； 登录失败

**顺序图3：**



**顺序图4：**



**类描述：**

1、User：负责用户相关的业务。

属性：

private AddFriendPresenter mPresenter;  
private FlatEditText mFedPhone;  
private FlatButton mFbtnConfirm;

方法：

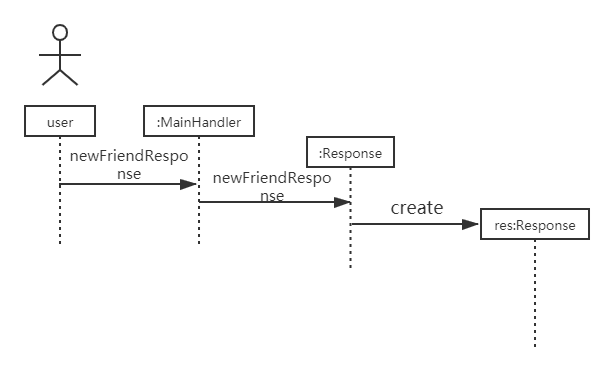
protected int setContentViewId()

protected void initUI() 初始化界面

public void sendRequestSuccess() 发送好友请求成功

public void phoneEmptyWarning() 手机号为空警告

**顺序图5：**



**类描述：**

1、Response：负责请求通知相关业务。

属性：

private NewFriendPresenter mPresenter;  
private RecyclerView mRecyclerView;  
private NewFriendNotifyAdapter mAdapter;  
private ProgressDialog mDialog;

方法：

public void showHandlingDialog()

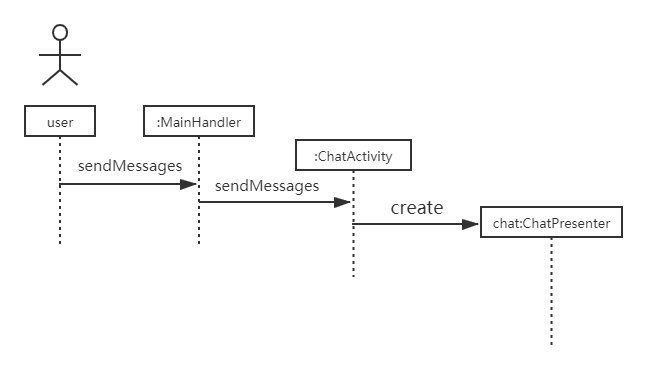
public void closeHandingDialog()

public void showHandlingError(int status, int errResId)

public void onNotifyStatusChanged() 消息状态改变方法

private void clearAllNotify() 清空消息通知

**顺序图6：**



**类描述：**

1、ChatActivity：负责系统的消息聊天业务。

属性：  
private static final String *INTENT\_KEY\_USERBEAN* = "userbean";  
private static final String *INTENT\_KEY\_CONID* = "conId";  
private static final String *INTENT\_KEY\_CONTYPE* = "conType";

private String mConversationId;  
private UserBean mUserBean;  
private CommonActionBar mActionBar;  
private CommonPtrLayout mPtrView;  
private RecyclerView mRecyclerView;  
private LinearLayoutManager mLayoutManager;  
private HxChatAdapter mAdapter;  
private HxChatController mChatController;  
private ResizeLayout mResizeLayout;  
private ProgressDialog mProgressDialog;

方法：

public static void start(Activity activity, String conId, UserBean userBean) 开始聊天

protected void beforeOnCreate(Bundle savedInstanceState)

public void showNoMoreMessageWarning() 没有更多消息通知

public void addNewMessage(EMMessage message, boolean forceScrollToBottom) 接收新消息

public void onMessageStatusChanged(EMMessage message) 改变消息状态

public void showHandlingDialog(int msgResId)

2、ChatPresenter：消息聊天的表示层。

属性：

private final int EACH\_PAGE\_SIZE = 20;  
private HxChatView mViewImpl;  
private Handler mMainHandler;  
private int mCurPlayVoicePosition = -1;

方法：

public HxChatPresenter(HxChatView viewImpl, Handler handler)

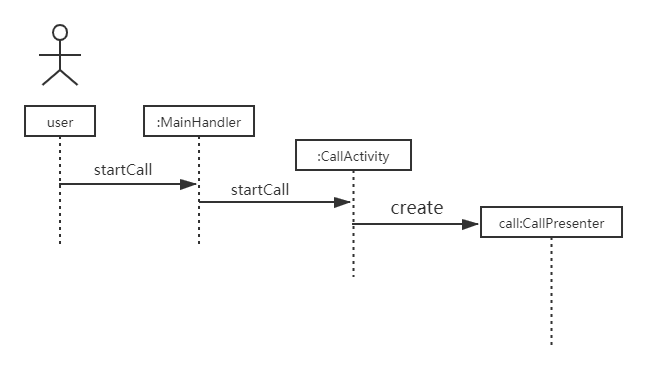
public void clearConversationUnreadCount(final String phone) 将会话未读消息数清零

public boolean isTextInputModeFirst(Context context) 是否优先展示文字输入模式

public void loadOnePageData(EMConversation.EMConversationType conType, String conId, final boolean isFirstLoad) 加载一页消息记录 conId会话id；conType会话类型；isFirstLoad 是否为第一次加载  
public void sendTextMessage(String conId, String message) 发送文本消息

public void sendVoiceMessage(String conId, String filePath, int seconds) 发送语音消息

**顺序图7：**



**类描述：**

1、CallActivity：负责系统的语音和视频通话业务。

属性：

private HxVideoCallPrenter mPresenter;  
private View mViewHeader;  
private EMOppositeSurfaceView mSfvOpposite;  
private EMLocalSurfaceView mSfvLocal;

方法：

public static void start(Context context, String phone, boolean isComingCall) 跳转到该界面的公共方法

protected int setContentViewId()

protected void initUI()

public void startVideoCall() 开始通话，发送通话请求

public void doAnswercall() 接受通话请求

public void doRejectCall() 拒绝通话请求

public void doEndCall() 结束通话

2、CallPresenter：语音视频通话业务的表示层。

属性：

protected HxCallView mViewImpl;  
protected Handler mMainHandler;

方法：

public HxCallPresenter(HxCallView viewImpl, Handler handler)

public void startVoiceCall(String phone) 请求语音通话

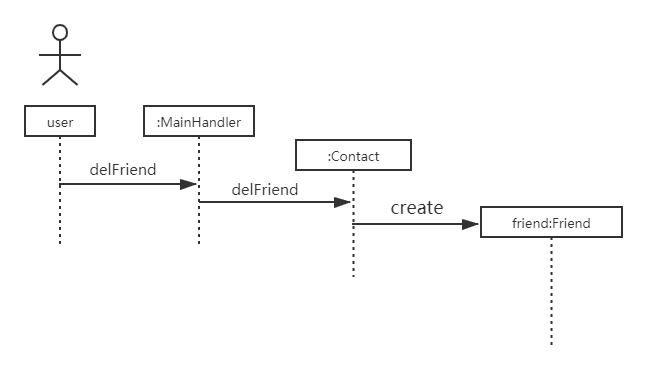
public void startVideoCall(String phone) 请求视频通话

public void answerCall() 接受通话请求

public void endCall() 结束通话

public void rejectCall() 拒绝通话请求

**顺序图8：**



**类描述：**

1、Contact：负责系统中联系人管理业务。

属性：

private ContactView mContactView;  
private ContactModel mModel;  
private RefreshContactDataTask mRefreshContactDataTask;

方法：

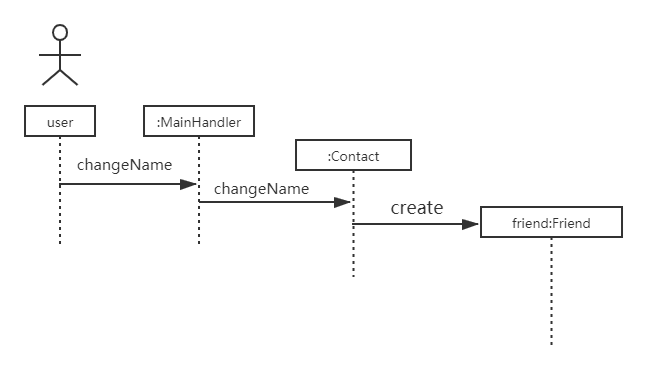
public void initData() 初始化数据

public void refreshAllContactData(Context context) 刷新所有通讯录数据

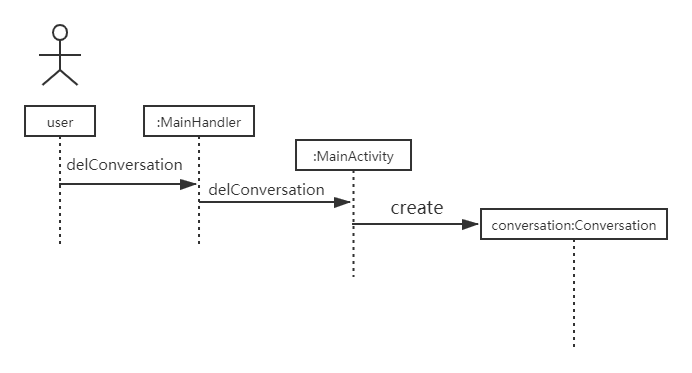
public void refreshContactDataInDb(final boolean isPtrRefresh) 刷新数据库中好友数据

public void refreshFriendNotify() 刷新好友通知

**顺序图9：**



**顺序图10：**



**类描述：**

1、Conversation：负责系统的会话业务。

属性：

private Handler mMainHandler;  
private ConversationView mViewImpl;  
private ConversationModel mModel;

方法：

public Converstion (ConversationView viewImpl, Handler handler) 构造方法

public void loadAllConversations() 加载所有会话片段

public void delConversation(final HxConversation conversation) 删除会话以及聊天记录

protected void initUI() 初始化界面

public void onItemLongClick(View view, RcvHolder holderint position) 长按触发删除会话的方法

public void onConversationBeDeleted(HxConversation conversation) 会话已经删除的提示