## 开发技术选择

APP采用JAVA语言进行设计开发。JAVA是一门面向对象编程语言，具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多线程、动态性等特点。JAVA目前已经是IT行业内被最广泛应用的编程语言之一。JAVA可以编写桌面应用程序、Web应用程序、分布式系统和嵌入式系统应用程序等，只要安装了JVM（JAVA虚拟机），就可将项目完整的移植与运行。而且它可以用面向对象的方法来描述用户的每一个动作，所以在程序除错、修改、升级和增加新功能等方面，依靠其面向对象特性，这些维护也变得非常容易。另外在数据的安全方面，JAVA提出了一个防止恶意代码攻击的安全机制，JAVA本身具有的特性，如：强类型机制、垃圾回收器、异常处理和安全检查机制，也使得用JAVA语言编写的程序具有很好的健壮性。

## 开发环境与性能优化选择

针对本软件的需求，将程序依托于Android移动设备平台。

后台采用LeanCloud后端云实现后台数据的存储，移动设备（客户端）采用XML可扩展置标语言与SQLite数据库实现本地数据的加载与缓存。

现在人们用到的国产智能手机大多是将Android系统作为为操作系统。 Android作为一种手机操作系统，在手机使用过程中要频繁地对存储的数据进行读取。它的数据读取方式一般有XML可拓展置标语言、配置文件( Preference)、文件(File)、 SQLite数据库、网络和Content Provider组件等。数据的存取方式与数据的存取次数、存取内容数量、存取效率等有关。因此，在程序设计阶段，依据程序要完成的功能和程序运行时的实际需求去选择合适的数据存取方式。

## 数据管理功能实现

从用户获取数据且对数据进行管理是APP的核心功能之一。在APP的使用过程中，用户的操作不可避免的会产生数据，获取数据，需要数据的哪些参数，参数的本地资源与网络资源的分配，都由APP的数据管理功能进行操作。

APP通过Android自带组件的Listener监控调用回调函数，如TextView.TextCheageListener、EditText.DateChangeListener、Button.ClickListener等。回调函数会将用户的部分操作记录保留。从当前Activity或Fragment的组件获取数据，组成用户的数据，根据数据中信息的重要程度，保存在XML、SQLite或LeanCloud后端云中。

XML由HTML发展而来，与HTML格式相似，但相对于HTML更加严格。XML描述的是结构、内容和语义，它不描述页面元素的格式化。HTML 侧重于如何表现信息，内容描述与显示整合为一体。更加适合于描述数据、存储数据、传输（交换）数据,在数据集成与交互方面具有非常大的优势。在APP不同Activity之间能够实现快速的信息传输与交换，提升用户体验。

SQLite数据库是一种轻量级数据库，具有一个嵌入式数据库引擎，占用资源非常少，而且占用内存空间非常少，这也是 Android移动设备采用 SQLite数据库的重要原因之一。 SQLite是关系型数据库管理系统，目前能够支持大多数主流操作系统，并且能够跟很多程序语言相结合。与其他的数据库管理系统如Mysql、Postgresql等相比较，SQLite的处理速度更快。

LeanCloud提供一站式后端云服务，包括数据模型与存储、云引擎、即时通讯、游戏解决方案、微信小程序等现代应用基础模块。通过LeanCloud云服务，实现数据的保存与查询获取变得十分方便与可靠。LeanCloud官方提供了多种开发语言版本的SDK与API供开发者调用，缩减了开发周期。LeanCloud在多种开发环境中提供全平台SDK支持，帮助各平台开发者快速集成，简化了开发流程。在使用LeanCloud的开发过程中，需要在Android程序中的build.gradle依赖文件中添加'cn.leancloud:realtime-android:7.2.3'，'io.reactivex.rxjava2:rxandroid:2.1.1'，'cn.leancloud:mixpush-android:7.2.3',然后通过sync下载导入三项依赖。

LeanCloud云数据库表格描述：

（1）（用户表图）

用户表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名name | 字段意义mean | 规则Rule | 默认值default |
| ObjectId | 唯一标识字段 | Primary key | / |
| 用户名 | 用户昵称 | Not null | / |
| 邮箱 | 用户邮箱 | / | / |
| 手机号 | / | / | / |
| 更新时间 | 最后一次用户修改信息时间 | Not null | / |



（2）聊天记录表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名name | 字段意义mean | 规则Rule | 默认值default |
| ObjectId | 唯一标识字段 | Primary key | / |
| ACL | 权限 | Not null | Read:true,write:true |
| C | 会话发起人 | Not null | 发起会话用户Id |
| Im | 最后一条聊天信息时间 | Not null | / |
| M | 会话名 | Not null | 会话发起人Id+会话接收人Id |
| Mu | / | / | / |
| Name | 会话接收人Name | Not null | / |

好友关系表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名name | 字段意义mean | 规则Rule | 默认值default |
| User | 用户Id | Not null | / |
| ObjectId | 唯一标识字段 | Primary key | / |
| ACL | 权限 | Not null | Read:true,write:true |
| Queryblankline | 查询空行标志 | Not null | null |
| Followee | 朋友Id | Not null | / |
| FriendStatus | 是否是对方好友 | Not null | false |

好友申请表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名name | 字段意义mean | 规则Rule | 默认值default |
| ObjectId |  |  |  |
| ACL |  |  |  |
| Friend |  |  |  |
| Status |  |  |  |
| User |  |  |  |
| createAt |  |  |  |
| updateAt |  |  |  |

Seek表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名name | 字段意义mean | 规则Rule | 默认值default |
| Publisher |  |  |  |
| objectId |  |  |  |
| ACL |  |  |  |
| Queryblankline |  |  |  |
| Content |  |  |  |
| Subject |  |  |  |
| createdAt |  |  |  |

## 权限与数据信息安全性

APP权限列表：联网权限、创建文件权限、存储文件权限、读取文件权限。

在开发之前，通过调研发现多数安卓APP获取了不需要的权限，于此本项目会杜绝多余权限的获取，以实现对用户隐私的保护。

读取与存储的文件都为用户首次登陆时自动创建的文件，尽量避免了用户个人敏感信息的获取。因为APP的权限不包括获取用户的联系人、通话记录、录音等权限，所以APP在运行过程中获取不到“无需”的用户隐私信息，从根源上杜绝了用户隐私的泄露。

## 数据持久化实现与展示

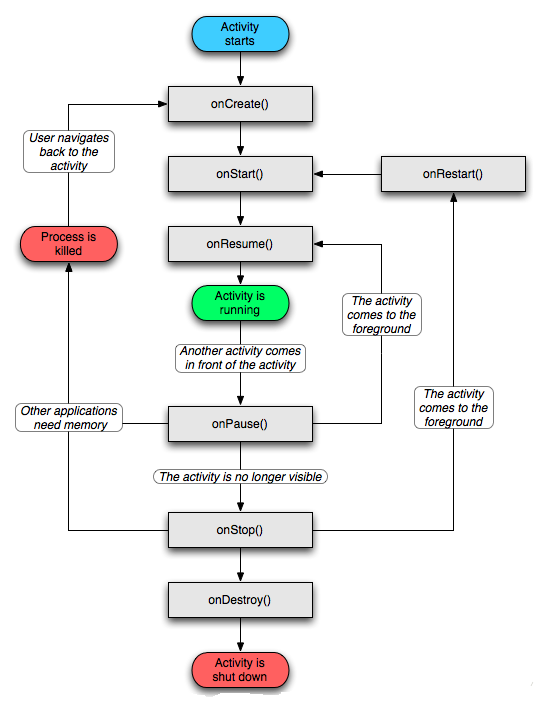
在Activity的生命周期中，如何在APP的后台获取、传递数据并不影响用户的体验是一个非常重要的关键点。Activity生命周期图中可以看出，Activity在onStop()阶段开始部分轻量级的不太耗时的回收工作，在生命周期的最后一个阶段onDestroy()进行回收工作和资源的释放。而在APP的工作中，Activity的创建与销毁是不可避免的，所以如何实现数据持久化是APP开发中较为重要的部分。如图

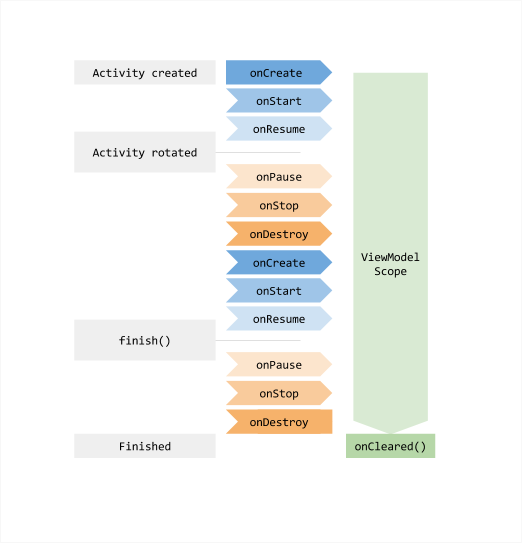
APP采用Fragment、ViewModel、ViewBinding实现数据的持久化操作。

Fragment是Android为了应用适配不同设备的大屏幕、支持更加动态和灵活的UI设计所提供的一个组件。

ViewModel类旨在以生命周期意识的方式存储和管理与UI相关的数据。ViewModel类允许数据在配置更改（例如屏幕旋转）后继续存在，其生命周期包括了Activity的所有生命周期—onCreate()、onStart()、onResume()、onPause()、onStop()、onDestroy()。ViewModel实例仅在Activity的onDestroy()结束之后调用onCleared()函数释放资源，结束生命周期。

ViewBinding是为了替代findViewById(内部实现还是使用findViewById)，在启动视图绑定后，系统会为模块中的每个xml文件生成一个绑定类，绑定类的实例包含对在相应布局中具有Id的所有视图的直接引用。所以因为会创建对视图的直接引用，杜绝了因视图ID无而引发的空指针错误。

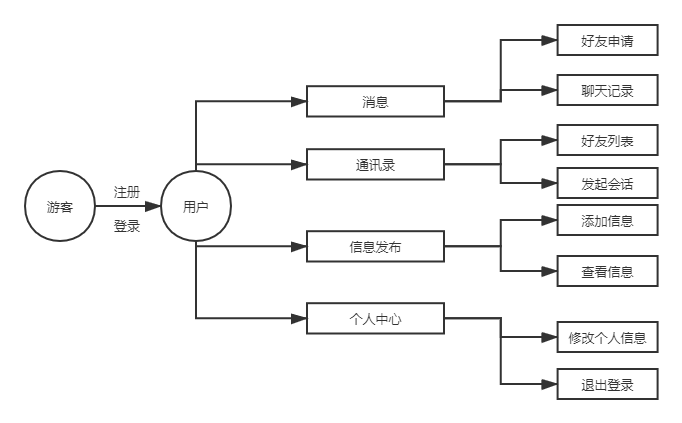


## 系统实现与效果展示：

要完成一个APP，首先需要对用户进行识别，游客通过注册成为用户。在注册的同时通过user.signUpInBackground().subscribe()方法将从EditText获取的用户名、密码、手机号、邮箱提交到LeanCloud云端，根据返回参数调用onNext()提示用户注册成功并且将返回参数AVUser保存到User.xml文件中。（截图）

若使用者本身拥有账号，则直接在登录界面进行登录，输入账号密码点击登录按钮，如果正确则进入MainAcvity，即功能界面。（截图）

APP功能界面通过Fragment实现



功能界面分为四个：消息界面、通讯录界面、信息发布界面、个人中心界面。

1. 消息界面：显示用户的聊天记录，点击可以通过Intent跳转到聊天界面，APP会在Activity的onCreate()函数创建IMClient对象登录LeanCLoud后端云的即时通讯服务器并创建用户与好友的Conversation。，在下方的输入框输入，点击发送，向好友发送消息。好友会在消息界面收到用户发送的消息。
2. 通讯录界面：用户的好友列表，点击可以通过Dialog查看好友的信息。点击发起会话进入聊天界面。也可在此界面添加好友。

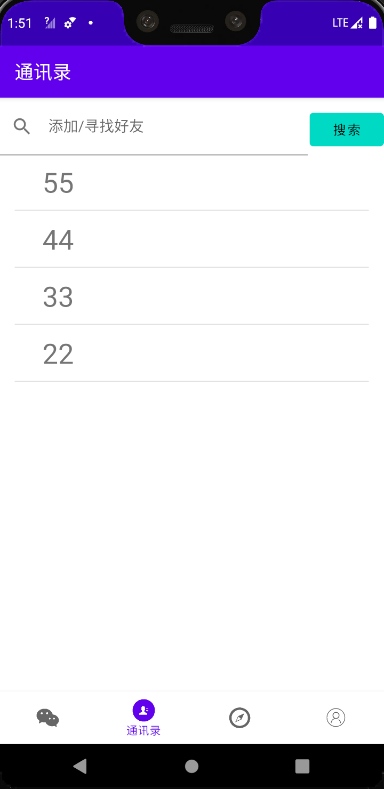
 

Figure 1

1. 信息发布界面：点击右下角floatbutton按钮可以发布信息。信息会提交到LeanCloud后端云，且因为信息表开启LiveQuery服务，所有客户端信息会实时同步到所有的客户端界面。
2. 个人信息界面：可以查看自己的信息，通过User.Logout进行登出操作，User.update修改自己的个人信息。