**Android端设计**

1. **质量属性：**
2. **性能**

为了提高使用流畅性以及交互性，应该采用Android原生开发的模式，而非B/S结构，有效提高了性能。

因为整体程序的界面较少，因此采用单Acitivity+多Fragment的整体架构进行开发，可以使界面切换更加流畅。

1. **安全性 可靠性：**

通过严格的账号密码权限体系，以及双端分离的方式，不同的用户角色应该展示不同的界面，访问不同的数据，提供不同的功能。

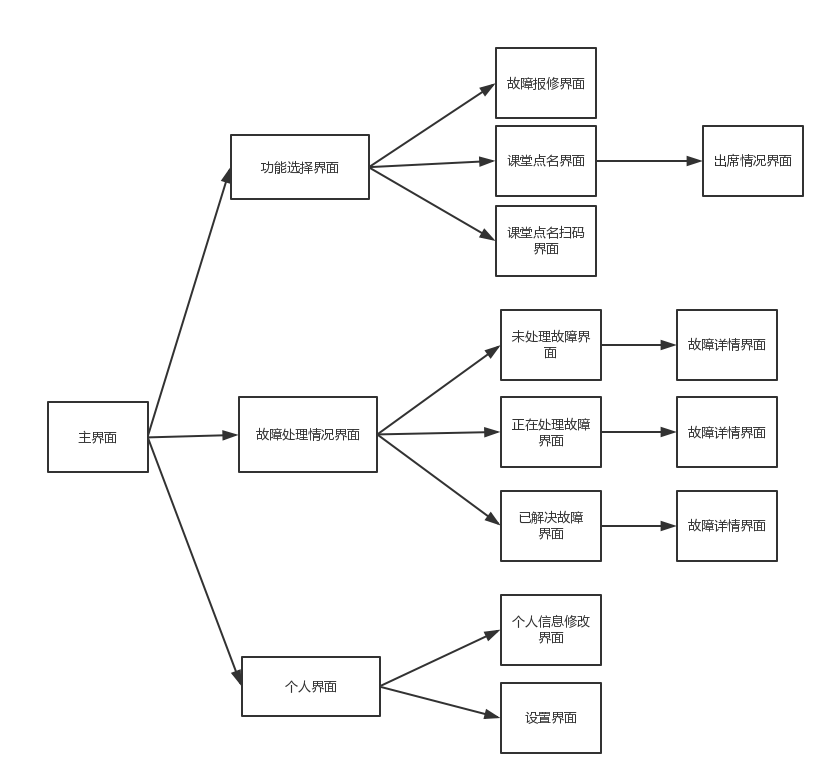
1. Android端只提供学生注册功能，对于教师，维修工人的注册需要通过账户管理平台进行访问。
2. 登录注册的逻辑实现，全部交给服务端处理，Android前端，不会直接访问数据库，增加安全性。
3. **可修改性设计**
4. 前后端分离开发，通过Http接口进行交互，整体系统耦合低。
5. **可维护性设计**
6. 采用MVC架构进行设计，提高整体代码编写的可维护性
7. 对于项目使用的第三方库应该集中管理，尽量使用通用的第三方库。
8. 前后端分离开发，减少整体系统的耦合度提高可维护性
9. **健壮性**
10. **易使用性**
11. 整体Android客户端界面美观友好，交互流畅，并且采用标准化的Android移动端UI设计规范，符合移动端用户的操作习惯。
12. 对于一些表单的填写，能够提供可选项，尽量不需要用户手动输入。比如故障类型的选择，故障地址选择。
13. 文字说明配合图标使用方便用户找到相应功能。

****

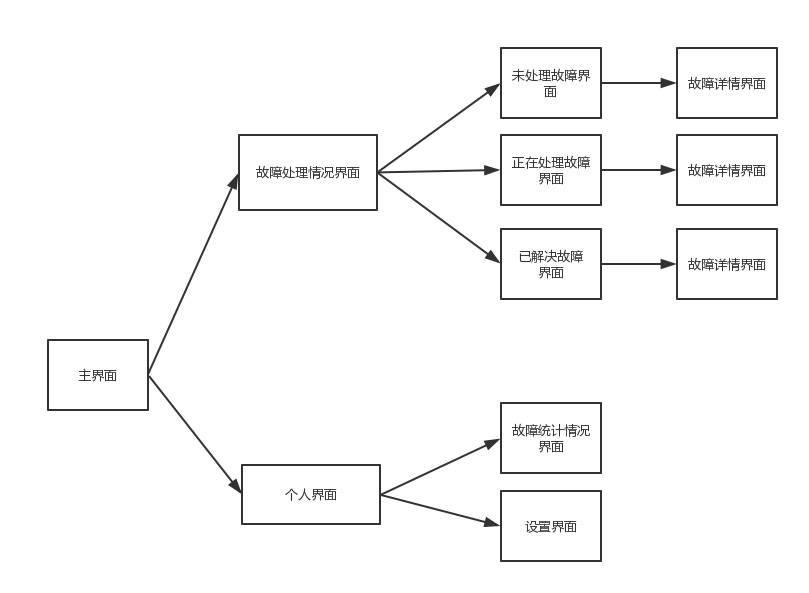
1. **界面设计：**

**{{软件使用说明书}}**

**师生端界面设计：**

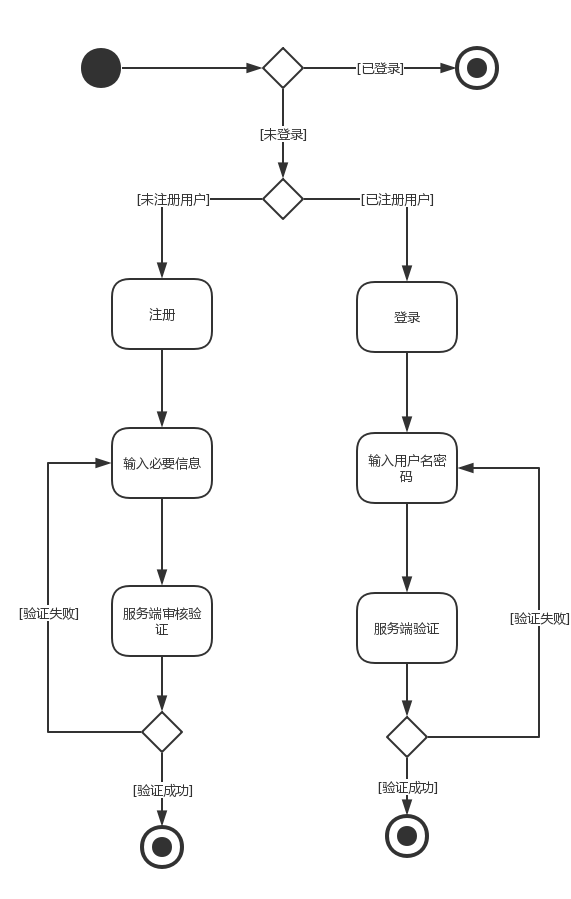
****

**管理端界面设计：**

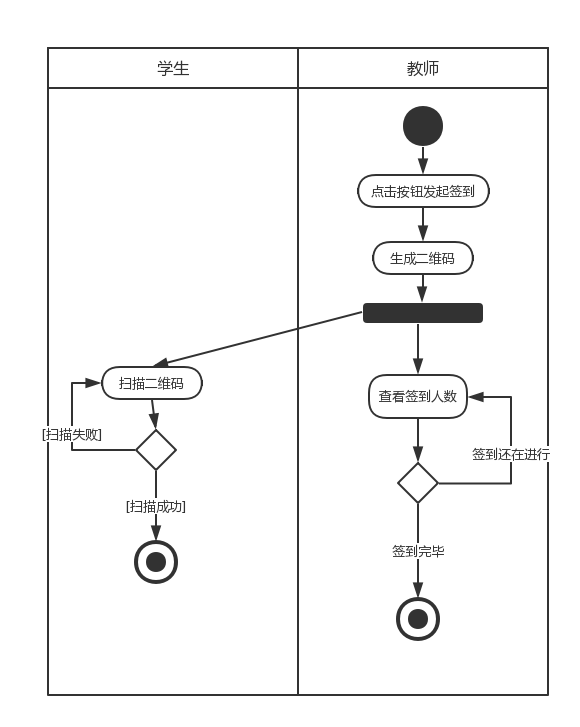
****

1. **功能活动图：**

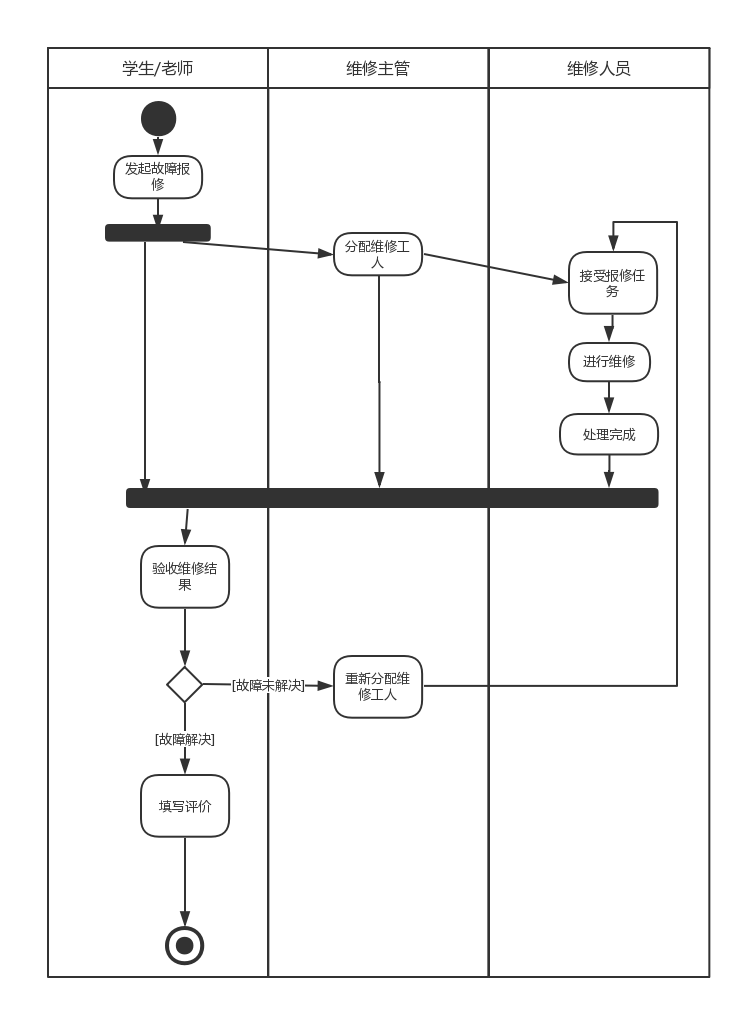
**登录/注册活动图：**

****

**签到活动图：**

****

**故障报修活动图：**

****

**学生/老师报修流程图：**

**维修工人处理报修流程图：**

**维修主管处理报修流程图：**

1. **Android 客户端整体架构设计：**

Android原生 APP开发通过java编写，其本身就有一套相对完整的开发框架，以每个Activity界面作为中心进行编写，通过XML语言来编写界面样式。通过Gradle项目管理语言打包成APK。这些是由Google官方提供给开发者的，所以在此的基础上进行项目整体架构上的设计：

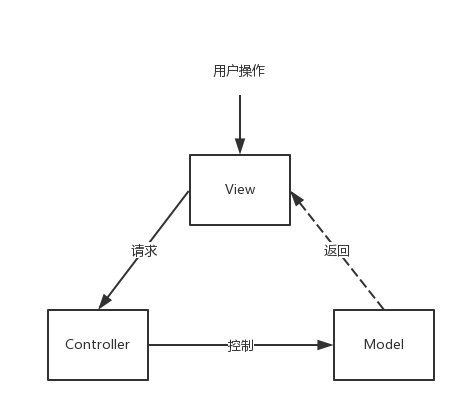
Android整体设计架构为MVC架构，以每个界面为中心的设计方式。

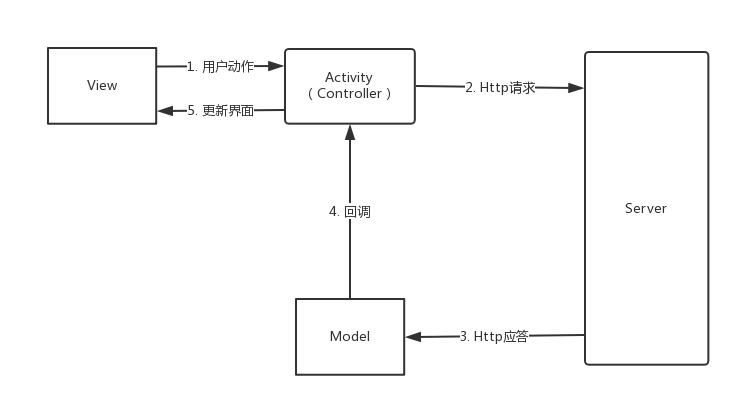
1. 采用MVC 模式

Model模型层负责数据的获取（JSON，Java Bean）

View 视图层负责用户界面编写（XML, JAVA）

Control 控制层负责将视图层和模型层的交互。(Activity, Fragment)





用户通过与界面进行交互操作，交互信息会传递到控制层Activity/Fragment上，控制层根据交互内容通过Http请求访问服务器，服务器应答返回为JSON格式，通过Model层进行转换处理，将结果放回给控制层，控制层根据返回结果更新界面。

1. 包图和类图：

因为采用前后端分离开发的模式，Android前端主要负责界面的实现，通过Http获取数据，显示在界面上；获取用户的动作和输入，通过Http请求上传给后端。整体的设计中心还是围绕在每个具体界面的实现上的：