**小龙潭监狱移动执法**

**执法门户系统**

**概**

**要**

**设**

**计**

**说**

**明**

**书**

北京华宇信息技术有限公司

BEIJING THUNISOFT INFORMATION TECHNOLOGY CORPORATION LIMITED

目录

[第一章、 总体设计方案 4](#_Toc24002)

[1.1 系统总体结构 4](#_Toc21565)

[1.1.1 系统总体架构设计 4](#_Toc1571)

[1.1.2 系统部署结构 5](#_Toc26909)

[1.2 核心问题的解决方案 6](#_Toc27674)

[1.2.1 保证系统先进开放 6](#_Toc23957)

[1.2.2 确保系统安全性 6](#_Toc24382)

[1.2.3 系统易于维护 7](#_Toc18347)

[1.2.4 基于浏览器操作模式提供输入输出易用性 8](#_Toc14109)

[1.2.5 对性能的规定 8](#_Toc23496)

[1.2.6 提供完善的数据管理能力 9](#_Toc3373)

[1.2.7 提供完善的故障处理与恢复能力 10](#_Toc29052)

[1.2.8 推送服务安全便捷可靠 11](#_Toc5893)

[第二章、 应用服务设计 12](#_Toc15161)

[2.1 TAP基础服务 12](#_Toc11165)

[2.1.1 身份认证服务 12](#_Toc12579)

[2.1.2 单点登录服务 13](#_Toc4974)

[2.1.3 权限控制服务 15](#_Toc28455)

[2.1.4 授权服务 16](#_Toc674)

[2.1.5 加密 17](#_Toc2288)

[2.2 Token认证服务 19](#_Toc21943)

[2.2.1 授权中心 19](#_Toc10878)

[2.2.2 基于IMEI号码保证设备安全 20](#_Toc11329)

[2.3 推送服务 20](#_Toc21092)

[2.3.1 基于MQTT协议推送服务 20](#_Toc6827)

[2.3.2 基于http接口的推送认证服务 20](#_Toc6899)

[第三章、 APP开发设计 20](#_Toc13105)

[3.1 APP整体开发方案 20](#_Toc29159)

[3.2 基础组件依赖 21](#_Toc26465)

[第四章、 系统接口设计 21](#_Toc23119)

[4.1 系统接口设计原则 21](#_Toc3810)

系统概要设计包括系统总体方案设计、应用支撑服务设计、APP开发设计和系统接口设计四大部分。

# 总体设计方案

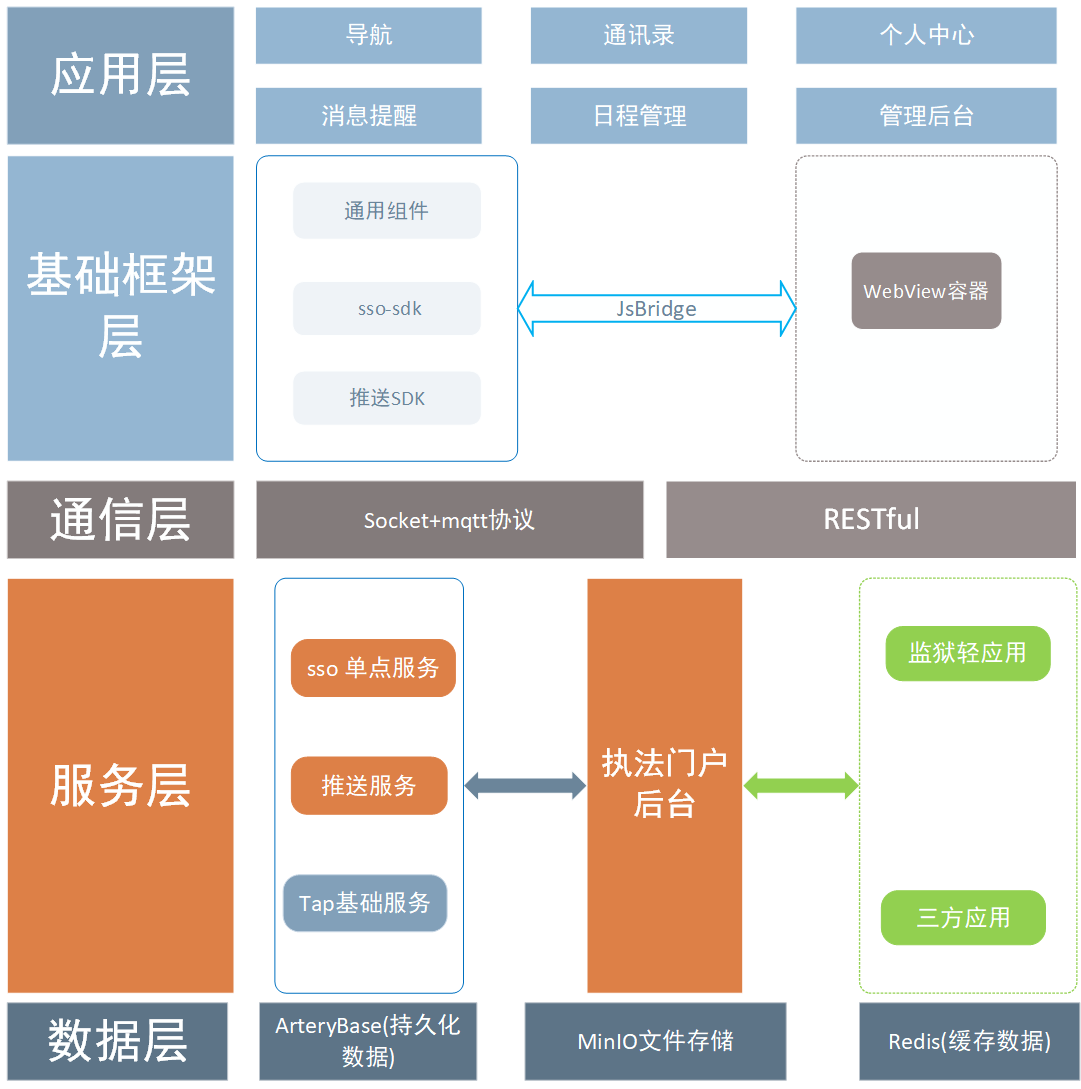
系统的总体架构设计，包括：系统逻辑模型、系统数据结构和系统部署结构，然后将阐述我们对招标文件中的技术要求和规范的实现思路，对核心问题的解决办法。

## 系统总体结构

* + 1. 系统总体架构设计

本系统架构设计在统一的网络基础设施支撑下，整体采用MVC分层架构，将平台拆分为Model+Service+Controller三层结构，整体基于公司自研框架Artery最新版本Artery6进行开发，通过http+Basic认证方式与其他依赖服务进行通信，不仅可以提高系统整体安全性，同时降低整体架构耦合性。

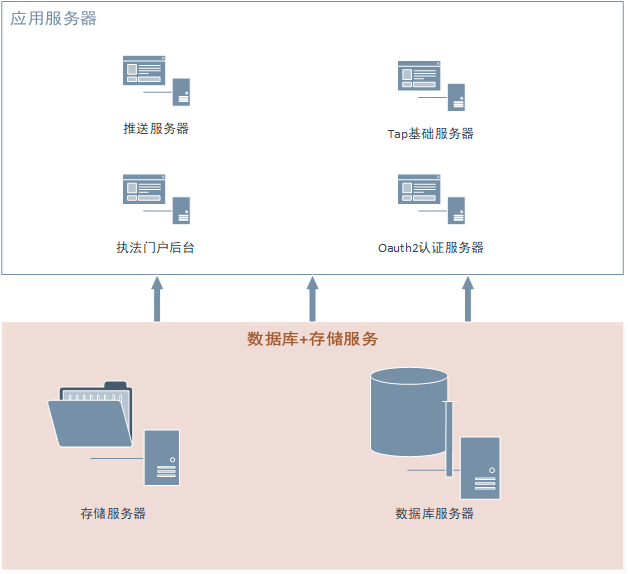
图表 1系统逻辑模型示意图



总体上包括四层应用架构整体设计，分别为数据层，通信层，基础框架层，应用层。其中四层主要包括：

1. 数据层: 数据层主要包括三方面ArteryBase提供持久化数据存储，MinIO提供对象存储服务，Redis提供高速缓存服务。
2. 服务层: 主要包括两个方面，应用内部依赖服务，SSO单点服务提供APP作为登录校验，推送服务提供APP接收、分发即时消息，Tap基础服务提供组织只够数据以及权限管理；oauth2认证服务提供APP接口认证；外部依赖服务主要监狱通知接口API、监狱轻应用服务
3. 基础框架层: 考虑到APP开发的高效性与扩展性，采用SDK集成方式对APP进行模块化开发，主要包括通用组件（网络请求，UI显示等）、sso单点SDk、推送SDK；全部组件做到热插拔，可根据具体需求与版本对相应组件进行升级或者替换。
4. 应用层: 应用层分为安卓项和后台管理前端项目，APP和服务间采用Restful接口交互，后台管理采用Artery6 UI组件进行开发。
   * 1. 系统部署结构

本系统是采用集中式部署，核心分为数据服务器和应用服务器。整体架构图如下：



图表 2 软件部署结构示意图

部署说明：

推送服务器可根据具体设备数量考虑是否独立部署，设备数量<5000可以考虑与应用服务器共同部署

## 核心问题的解决方案

根据经验，为了更好地实现本系统，我们需要解决签章，文书，流程等核心技术问题。这些核心技术问题是否能够解决得好，直接影响到整个软件系统能否达到理想的应用效果。下面将分别叙述我们公司对这些核心技术问题的解决方案：

* + 1. 保证系统先进开放

**我们采用以下系统实现思路和技术来确保系统的先进性：**

* 采用浏览器/服务器开发模式开发，支持大规模数据处理应用。
* 采用JAVA开发语言和J2EE中间件平台。
* 专门设计增量数据同步系统，保证监狱及监区的数据共享和数据一致性。
* 采用Artery6的前端mvx框架模式。
* 采用消息中间件解耦服务和第三方系统，增强系统的稳定性。

**我们采用以下系统实现思路和技术来确保系统的开放性：**

* 系统采用标准SQL语句，并对数据存储进行抽象与封装，采用自主可控的数据库arteryBase。
* 采用Restfull接口设计，增强系统的对外提供服务的能力。
* 基于TAP开发，采用统一的组织机构，单点登录，权限管理的能力。
  + 1. 确保系统安全性

我们采用CA、SSL技术确保系统安全性，当浏览器与一个网站建立https连接时，浏览器与Web Server之间使用司法系统统一CA身份认证系统经过一个握手的过程来完成身份鉴定与密钥交换，从而建立安全连接。具体过程如下：

* 用户浏览器将其SSL版本号、加密设置参数、与session有关的数据以及其它一些必要信息发送到服务器。
* 服务器将其SSL版本号、加密设置参数、与session有关的数据以及其它一些必要信息发送给浏览器，同时发给浏览器的还有服务器的证书。如果配置服务器的SSL需要验证用户身份，还要发出请求要求浏览器提供用户证书。
* 客户端检查服务器证书，如果检查失败，提示不能建立SSL连接。如果成功，那么继续。
* 客户端浏览器为本次会话生成pre-master secret，并将其用服务器公钥加密后发送给服务器。
* 如果服务器要求鉴别客户身份，客户端还要再对另外一些数据签名后并将其与客户端证书一起发送给服务器。
* 如果服务器要求鉴别客户身份，则检查签署客户证书的CA是否可信。如果不在信任列表中，结束本次会话。如果检查通过，服务器用自己的私钥解密收到的pre-master secret，并用它通过某些算法生成本次会话的master secret。
* 客户端与服务器均使用此master secret生成本次会话的会话密钥(对称密钥)。在双方SSL握手结束后传递任何消息均使用此会话密钥。这样做的主要原因是对称加密比非对称加密的运算量低一个数量级以上，能够显著提高双方会话时的运算速度。
* 客户端通知服务器此后发送的消息都使用这个会话密钥进行加密。并通知服务器客户端已经完成本次SSL握手。
* 服务器通知客户端此后发送的消息都使用这个会话密钥进行加密。并通知客户端服务器已经完成本次SSL握手。
* 本次握手过程结束，会话已经建立。双方使用同一个会话密钥分别对发送以及接受的信息进行加、解密。
  + 1. 系统易于维护

为了满足用户日常管理的需要，系统必须为用户的日常管理提供简便的方法。我公司提供的系统在维护方面采用了如下方法：

* 采用可视化的应用系统建模、开发和维护技术，甚至是系统管理员通过专门培训后，都可以对应用系统进行修改和维护。
* 应用系统的日常管理维护方面，本系统提供了基于WEB界面的系统管理子系统，可对应用系统日常管理的各个方面提供全方位服务，如组织机构管理、权限管理、数据管理等。
* 应用系统和数据的升级维护方面，当应用系统需要升级更新或者数据库中的数据需要进行调整时，如果采用传统的手工运行升级程序的方法，存在两个问题：首先，速度缓慢，可能需要许多时间才能完成系统的更新；其次，由于升级的时间不可能完全同步，可能发生某些节点采用了新数据而另外一些节点则使用老数据而导致数据混乱的情况。为此，我公司将提供应用系统和数据库自动发布系统，每次对系统的更新仅需对总数据中心服务器上的系统进行更新，其他所有服务器的应用程序和数据将自动被更新升级，这就极大地降低了系统升级维护的工作量。
* 操作系统和数据库管理系统的维护，由于我公司建议采用基于Linux系统的服务器，又提供了断电保护程序保证系统在断电的情况下也能够正常关机，且系统本身就具有良好的稳定性，日常维护工作量很少。一旦发生异常情况，只要网络可用，维护人员可通过加密的SSH连接迅速对故障服务器的操作系统或数据库服务器进行远程维护，不需要到达故障现场。这对于提高故障维护的响应速度有极大的帮助。
  + 1. 基于浏览器操作模式提供输入输出易用性

由于操作系统的浏览器的普及率已经相当高，因此用户在使用浏览器访问本项目的各种应用功能的时候，将能够极大的提供操作的简便性，降低学习曲线。对于各种功能的输入输出，包括字段输入、查询结果列表的显示、附件上传下载等，都将和普通网页的操作方式一致。

* + 1. 对性能的规定

**精度**

* 数值型：本系统对于数值型的数据精度一般通过数据库进行定义，对于实际应用中可能存在歧义或者异议的精确到小数点后两位；
* 字符型：本系统对于字符型的数据保证数据定义的明确性（字段名以中文解释的首拼字母或者英文单词表示，同时配合详细的设计文档），以确保不产生歧义为第一要素，同时根据业务的不同，用户能够自定义相关常用字符数据；
* 媒体文件：本系统对于直接上传到服务器硬盘的媒体文件能够进行MD5或HASH值的效验，并将校验值存入数据库，以确保数据的完整性。日后打开相关文件时系统能够自动对比校验值并给出提示。

**时间特性**

本系统的软件的时间特性设计如下：

* 软件一般响应时间不大于1秒，一般检索、统计类应用的响应时间小于３秒，基于大型数据检索和统计的统计分析和图表生成小于１０秒
* 数据的转换和传送时间不超过9秒；
* 其他的应用和业务处理在1秒内返回信息，即使处理失败的也会在3秒内返回错误信息；
* 单次推送10000条1k大小的消息整体延时不高于2秒

**灵活性**

当需求发生某些变化时，本系统对这些变化的适应能力包括：

* 除非发生根本性的需求变化或者重新设置，需求的变化不会导致用户操作的重新设定；
* 运行环境的不会有大的改变，客户提出更换更高档配置的服务器等情况除外；
* 同其他软件的接口的变化将以详细的方式予以说明并提供范例；
* 精度和有效时限的变化能够满足精度的要求；
* 计划的变化或改进将以不改变用户操作习惯为前提，除非这种改变是对用户使用体验的提升。
  + 1. 提供完善的数据管理能力

数据库的数据由系统应用层进行存储和操作，这种操作是基于事务的。而数据库的管理是由DBMS来完成。

在应用层，系统通过项目配置文件已经调研确定的业务流程，将各种有关联的操作设计开发成基于事务的一连串用户操作。当整个操作过程中发生故障，系统平台能够将事务回滚到最初状态。

而通过可视化的DBMS来管理数据库，进行日常的各种数据备份、监测、日志清除等操作，能够大大的方便用户管理员的日常工作。同时，本系统也可以将一些常用的数据库管理步骤固化成程序，并在预定的时间自动执行，以减轻管理员的工作负担。

针对数据的不断增长，系统可以考虑提供基于数据库集群的存储和管理模式。根据客户单位的实际业务量，以及本系统提供的硬件配置，数据库的存储容量在5－10年内不会有问题。而为了提高用户的并发访问，日后系统可以考虑对数据库进行升级，采用集群的存储方式以便支持更多的并发量。不过鉴于本系统面向监狱内部，且在各单位都部署独立的服务器，各单位的操作人员数量也存在一定的上限，因此不管是否提供数据库集群的存储和管理模式，本系统都能够为客户提供良好的服务。

* + 1. 提供完善的故障处理与恢复能力
* 对于可能的软件本身造成的系统故障或运行障碍，系统本身提供详尽的处理文档和说明；
* 对于用户操作造成的系统故障，提供系统的故障恢复机制，确保系统的正常和数据的完整；
* 对于软件附带硬件系统故障造成的系统瘫痪，系统提供备用的硬件或者硬件的应急恢复机制；（此项，需要用户提供额外的服务器资源支持）
* 对于服务器硬件故障造成的系统瘫痪，系统提供紧急的恢复点以供恢复系统的运行和数据的完整性。
  + 1. 推送服务安全便捷可靠
* 基于发布/订阅（publish/subscribe）模式的"轻量级"通讯协议MQTT，可以以极少的代码和有限的带宽，为连接远程设备提供实时可靠的消息服务
* 推送服务验证插件化，可以添加多种认证方式（数据库，缓存，http等），保证推送服务安全
* 强大的后台管理界面，方便用户排错，监控、配置，管理与升级

# 应用服务设计

## TAP基础服务

TAP是华宇应用平台的简称，TAP基础服务包括身份认证服务、单点登录服务、授权服务、消息平台，为系统提供安全方面的服务。具体包括。

* + 1. 身份认证服务

本平台可以集成监狱已有的身份认证平台，如果不具备集成条件，按如下方案实现。

身份认证是在计算机网络中确认操作者身份的过程。身份认证可分为用户与主机间的认证和主机与主机之间的认证。用户与主机之间的认证可以基于以下一个或几个因素：

* 用户所知道的：例如口令、密码等。
* 用户拥有的，例如印章、智能卡（如信用卡等）。
* 用户所具有的生物特征：例如指纹、声音、视网膜、签字、笔迹等。

身份认证组件支持如下的认证方式：

* 静态密码

用户的密码是由用户自己设定的。在网络登录时输入正确的密码，计算机就认为操作者就是合法用户。

* 智能卡

　　一种内置集成电路的芯片，芯片中存有与用户身份相关的数据，智能卡由专门的厂商通过专门的设备生产，是不可复制的硬件。智能卡由合法用户随身携带，登录时必须将智能卡插入专用的读卡器读取其中的信息，以验证用户的身份。

* 静态密码认证方式

基于用户名与口令进行身份认证机制，是一种最简单和最常用的身份认证机制；

* + 1. 单点登录服务

本平台可以集成监狱已有的单点登录系统，实现行内用户的访问控制；对于行外用户，使用CAS作为单点登录设计方案，如下阐述。

* CAS实现方案

CAS的参与者包括：

* + - CAS认证中心(CAS Server)：一个集中所有用户认证的唯一的服务器，提供认证服务。
    - CAS认证客户端(CAS Client)：使用认证中心服务实现单点登录的应用系统。
    - 授权票据Cookie( Ticket Granting Cookie，TGC)：浏览器在登录CAS认证中心成功后，认证中心在该浏览器中写下的一个Cookie，主要表示该用户已经被认证中心认证通过，为以后登录应用系统提供依据。
    - 服务票据(Service Ticket，ST)：当用户从浏览器请求使用受保护的资源时，该应用的CAS认证客户端会自动重新定向到认证中心，认证中心根据TGC分发给该请求一个服务票据，然后CAS认证客户端持有该ST去确认并取得该用户的身份。
    - Shiro：Apache Shiro是一个强大易用的Java安全框架，提供了认证、授权、加密和会话管理功能，可为任何应用提供安全保障 - 从命令行应用、移动应用到大型网络及企业应用。
* 实现原理

CAS(Central Authentication Service)是美国耶鲁大学为Web应用统一设计的单点登录认证方案。它能够适合任何语言编写的客户端应用，实现简单实效。它使用SSL作为服务器端和客户端之间的通讯协议，安全性可以保证，是很好的SSO解决方案。

CAS认证中心提供系统的认证服务，其主要支持的业务有：向所有应用系统提供统一认证服务，同时提供用户调用接口。所有经CAS认证中心授权的应用系统在整个单点登录系统中地位是对等的，也就是只要在其中一个应用系统上进行登录，就可以直接访问其它应用系统而无需重复登录。每个应用系统进行用户身份认证的方式可能略有不同，要根据具体情况实现。单点登录只负责决定用户是否能进入某个应用，这些应用系统有着各自的资源权限管理策略，用户对资源的访问权限由各应用系统独立控制。

* 单点登录流程



* 浏览器访问受保护的Web应用系统资源，如：在这个过程中，CAS认证客户端过滤从浏览器传来的每一个Web请求并分析请求中是否包含服务票据(ST)，如果没有包含，则说明该用户没有经过认证。
* 该应用的CAS认证客户端需要重定向到CAS认证中心请求服务票据(ST)，并把Web请求的URL编码后作为参数传递。
* 用户认证过程，如果用户提供了正确的凭证，CAS认证中心分发给该请求一个随机的包含用户名等信息的服务票据(ST)，该ST是不可以仿造的，而且有一定的时效性。最后，重定向CAS认证客户端在（2）中通过参数传递的URL，并将服务票据(ST)作为URL参数传递。在这种过程当中，CAS认证中心还会向经过认证的客户端浏览器发送一个允许票据Cookie(TGC)，TGC相当于CAS认证中心发送给用户的通行证，如果用户持有TGC且其还没失效，那么就重定向到受保护的Web应用系统资源，达到了SSO的效果；如果TGC失效，那么用户需要重新进行认证；同时，由于对用户最有价值的TGC只是跟CAS认证中心有关，CAS能很好的解决Cookie不能跨域的问题。
* CAS认证客户端持有该服务票据(ST)到CAS认证中心确认并取得该用户的身份，这个过程不通过浏览器而是由CAS认证客户端应用直接访问CAS认证中心。CAS认证中心确认后返回该用户的身份信息；如果服务票据(ST)不正确，则返回错误信息并提示用户重新进行认证。
  + 1. 权限控制服务

根据组织模型和权限模型，对资源模型进行用户权限控制。

主要功能包括：

* 组织管理，对用户所属的组织进行管理，组织以树形结构展示，组织管理具有组织的增、删、改、查功能；
* 组权限管理，管理组的权限，包括包含用户、所属角色、组权限资源和组总权限资源四部分，某个组的权限信息可用公式表示：组权限 = 所属角色的权限合集 + 组自身的权限；
* 角色权限管理，管理角色权限，包括包含用户、包含组和角色权限三部分，某个角色的权限的计算公式为：角色权限 = 角色自身权限；
* 用户权限管理，管理用户权限，包括所属角色、所属组、用户权限、用户总权限资源和组织管理五部分。某个用户总的权限信息存在如下计算公式：用户权限 = 所属角色权限合集 + 所属组权限合集 + 用户自身权限；
* 操作日志管理：管理和记录本系统的操作日志。

权限过滤的实现原理与方案如下：

* 实现原理

Shiro 认证与授权处理过程。

* 被 Shiro 保护的资源，才会经过认证与授权过程。
* Shiro 首先检查用户是否已经通过认证，如果未通过认证检查，则跳转到登录页面，否则进行授权检查。这块是身份认证的一部分。
* 认证通过后接受 Shiro 授权检查，授权检查需要通过Realm 获取用户权限信息。

Shiro 需要的用户权限信息包括 Role 或 Permission，可以是其中任何一种或同时两者，具体取决于受保护资源的配置。如果用户权限信息未包含 Shiro 需要的 Role 或 Permission，授权不通过。只有授权通过，才可以访问受保护 URL 对应的资源，否则跳转到“未经授权页面”。

* 实现方案

结合Apache Shirio安全控制框架来辅助web应用的权限控制。

1、权限控制过滤器

用于控制用户访问的菜单，URL等资源控制。权限判断逻辑放在过滤器中，当访问菜单或URL在用户权限数据中，则可以访问，否则禁止访问。

平台中扩展shiro的默认过滤器实现，在继承方法中获取权限信息，当前请求与获取的权限信息进行匹配，匹配成功则跳到要访问页面。

过滤器在web.xml声明，根据配置的地址进行拦截过滤；

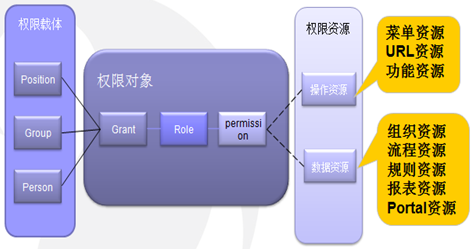
2、权限控制标签

用于控制页面的下拉框、选择框、按钮等页面资源的控制。权限判断逻辑在自定义标签类中，得出资源的可见，不可见，可见不可用等判断结果。

* + 1. 授权服务

授权服务实现平台的权限管理功能，按照权限模型，维护角色和角色的权限，为不同的平台提供角色的权限信息、不同用户的角色信息、不同用户的权限信息。

平台的权限模型如下：



权限控制支持如下的功能：

* 支持机构、部门、工作组、岗位、群组及员工等类型丰富的组织实体；
* 实现资源化的统一管理方式管理应用系统中的所有信息；
* 以角色作为授权的载体，人员，群组，岗位和部门可以通过与角色关联的方式实现授权；
* 实现对于URL，菜单，按钮，页面控件和数据列的统一授权管理；
* 机构、资源和授权模块之间耦合松散，可以独立使用；
* 角色及权限管理组件需要能够灵活配置多种用户角色，以及多种用户权限。并能够实现平台无关性，为上层提供用户角色识别，以及权限管理等操作。
  + 1. 加密

哈希和加密技术堪称安全技术的两块基石。哈希（Hash）是将目标文本转换成具有相同长度的、不可逆的杂凑字符串（或叫做消息摘要），而加密（Encrypt）是将目标文本转换成具有不同长度的、可逆的密文。两者有如下重要区别：

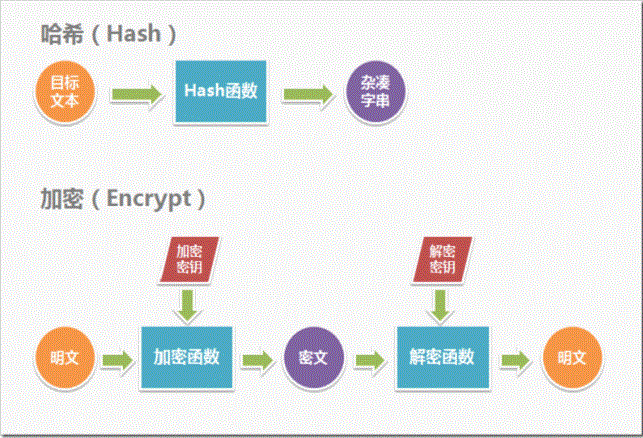
　　哈希算法往往被设计成生成具有相同长度的文本，而加密算法生成的文本长度与明文本身的长度有关。

　　哈希算法是不可逆的，而加密算法是可逆的。

　　一个哈希算法 是一个多对一映射，给定目标文本S，H可以将其唯一映射为R，并且对于所有S，R具有相同的长度。由于是多对一映射，所以H不存在逆映射 使得R转换为唯一的S。

一个加密算法 是一个一一映射，其中第二个参数叫做加密密钥，E可以将给定的明文S结合加密密钥Ke唯一映射为密文R，并且存在另一个一一映射 ，可以结合Kd将密文R唯一映射为对应明文S，其中Kd叫做解密密钥。

下图是哈希和加密过程的图示：

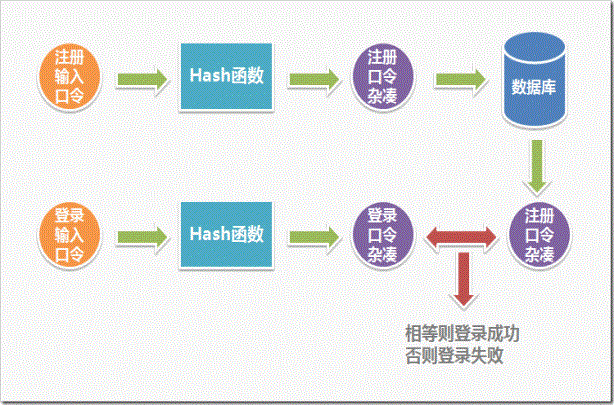


哈希和加密技术应用领域非常广泛，在平常的软件开发中最常见的应用——数据保护。

所谓数据保护，是指在数据库被非法访问的情况下，保护敏感数据不被非法访问者直接获取。 要实现上述的数据保护，可以选择使用哈希或加密两种方式。那么在什么时候该选择哈希、什么时候该选择加密呢？基本原则是：如果被保护数据仅仅用作比较验证，在以后不需要还原成明文形式，则使用哈希；如果被保护数据在以后需要被还原成明文，则需要使用加密。

　　在开发框架中，内置了对主要哈希和加密（对称/不对称）算法的支持。

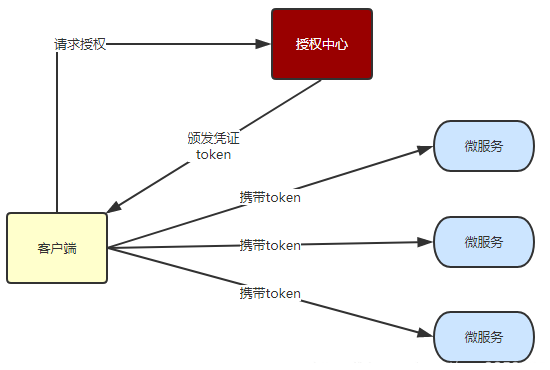
　　应用实例如下图：



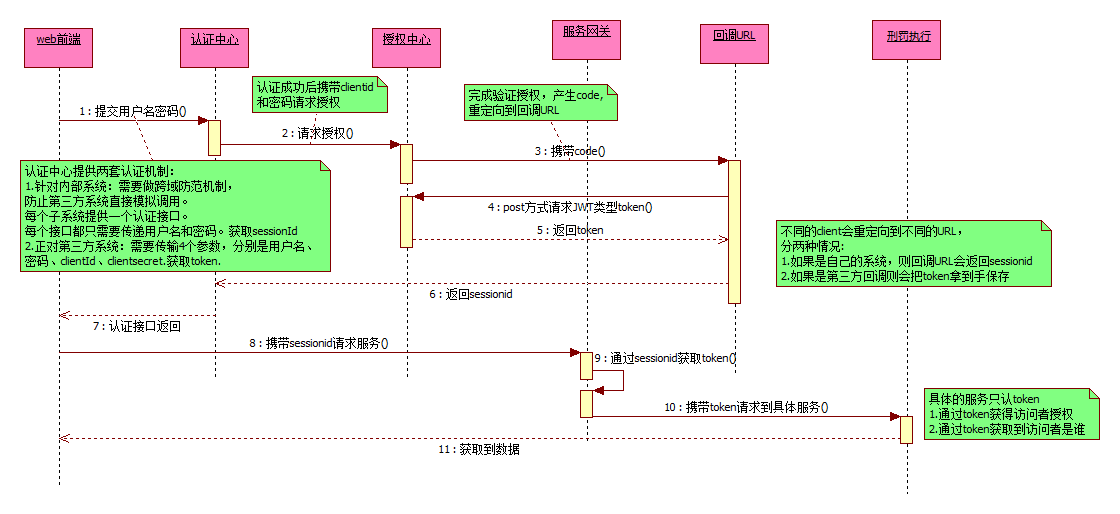
## Token认证服务

* + 1. 授权中心

移动端接口调用需要做授权和鉴权，采用oauth2协议作为服务的授权和鉴权，结构图如下：



服务的授权与鉴权流程如下：



* + 1. 基于IMEI号码保证设备安全

移动认证采用IMEI号码作为设备的唯一标识，通过IMEI号码与人员信息绑定实现登录认证，整体过程无需人员信息输入，简化登录的整体操作，同时基于IMEI号码的唯一性与设备的后台可维护（启迪国信做统一维护），保证了认证系统的整体安全性。

## 推送服务

* + 1. 基于MQTT协议推送服务

MQTT是一个基于客户端-服务器的消息发布/订阅传输协议。MQTT协议是轻量、简单、开放和易于实现的，这些特点使它适用范围非常广泛。在很多情况下，包括受限的环境中，如：机器与机器（M2M）通信和物联网（IoT）。其在，通过卫星链路通信传感器、偶尔拨号的医疗设备、智能家居、及一些小型化设备中已广泛使用。考虑到推送服务的可移植性、高效性，本次推送服务基于MQTT协议实现

* + 1. 基于http接口的推送认证服务

推送服务的服务采用Http接口方式进行验证，每次连接推送服务需要请求验证当前用户是否可连接推送服务，具体校验规则可在Tap扩展字段添加验证规则。基于http的验证方式，既保证了推送服务的安全，保证推送服务与业务系统解耦，降低了体统整体复杂性。

# APP开发设计

## APP整体开发方案

APP整体基于MVP（Model-View-Present）架构采用模块化方案进行开发，整体分为：Base基础模块、Push推送模块、Schedule日程模块、Message消息模块、Contacts联系人模块等。此种方式开发优势：

* 降低耦合度
* 模块职责划分明显、利于测试驱动开发
* 代码灵活性、复用性高
* 更小的系统控制粒度，降低系统复杂性

## 基础组件依赖

|  |  |  |
| --- | --- | --- |
| 基础组件名称 | 主要用途 | 版本号 |
| butterknife | 依赖注入 | 9.0.0 |
| Glide | 图片显示 | 4.8.0 |
| Permissions-Dispatcher | 动态权限管理 | 3.3.2 |
| Retrofit+RxJava | 网络请求框架 | 2.5.0 |

# 系统接口设计

## 系统接口设计原则

* 通用性：系统采用Restful的架构方式，以及通过json来传输数据，确保了本系统和其他业务系统之间的接口能够方便且通用的对接。
* 可操作性：系统之间的接口不改变原系统的体系结构和信息模型结构；所有的业务系统内部结构，不会因为与其它系统的接口而发生改变。
* 安全性：本系统以通用开放的数据接口标准，最大限度的提高了不同系统间互访的兼容性。同时，本系统通过各种安全手段（包括防火墙、IP黑白名单、授权机制），确保了只有可信的系统才能够访问本系统的接口数据，防止安全漏洞的产生。
* 可控性：本系统的接口具有双向可控性，保证了业务系统内部模块之间以及其他业务系统之间的协调。系统对所有的接口功能提供了开启关闭功能，同时对接口的开放权限也能通过系统管理模块得到授权控制，确保了接口的可控性。
* 一致性：本系统的接口的如果和本系统的业务系统有关联的话，接口将通过后台中间件代码直接访问底层数据库，确保取得的数据是最新且有效的数据，保证了一致性。
* 可记录和恢复性：系统通过完善的日志管理系统，将所有接口通信记录都保存到日志系统中，以备查案。同时当接口程序对本系统的业务数据带来了一定的修改操作时，系统提供恢复操作的功能，以防调用接口的程序方出错造成数据破坏。
* 接口处理方式：本系统的接口，将默认采取自动的计划任务的后台处理方式，省去了用户的人工管理方式。同时，用户也可以在一定的情况下对系统进行人工干预。另外，系统通过各种安全保障机制，确保传输安全可靠。数据在传输前进行加密，并在到达接口业务系统后解密接收。