通用农业物联网数据用户管理平台

概要设计

目录

[一、引言 2](#_Toc13733791)

[1.1 编写目的 2](#_Toc13733792)

[1.2 背景 2](#_Toc13733793)

[1.3定义 2](#_Toc13733794)

[1.4 参考资料 3](#_Toc13733795)

[二、总体设计 3](#_Toc13733796)

[2.1需求规定 3](#_Toc13733797)

[2.1.1 系统功能 3](#_Toc13733798)

[2.1.2 系统性能 7](#_Toc13733799)

[2.1.3 输入输出要求 7](#_Toc13733800)

[2.1.4 数据管理能力要求 7](#_Toc13733801)

[2.1.5 故障处理要求 8](#_Toc13733802)

[2.2 运行环境 8](#_Toc13733803)

[2.2.1 设备与支持软件 8](#_Toc13733804)

[2.3 系统层次结构 8](#_Toc13733805)

[2.4 功能需求与程序的关系 11](#_Toc13733806)

[三、接口设计 12](#_Toc13733807)

[3.1 用户接口 12](#_Toc13733808)

[3.2 外部接口 12](#_Toc13733809)

[四、系统数据结构设计 13](#_Toc13733810)

[4.1 逻辑结构设计要点 13](#_Toc13733811)

[4.2 物理结构设计要点 17](#_Toc13733812)

[五、系统出错处理设计 18](#_Toc13733813)

[5.1 出错信息 18](#_Toc13733814)

[5.2 补救措施 18](#_Toc13733815)

[5.3 系统维护设计 18](#_Toc13733816)

# 一、引言

## 1.1 编写目的

本概要设计文档主要用来指导通用物联网平台的详细设计工作，为详细设计提供统一的参照标准，其中包括系统的内外部接口、系统架构、编程模型以及其他各种主要问题的解决方案。在此文档被经过同行评审后，所有有关本系统的详细设计必须遵照此文档的相关标准和约束来进行。另外，此文档也作为对详细设计文档进行同行评审所依照的标准之一。

本文档主要描述的是通用物联网平台的概要设计，其中包括定义系统的内外部接口、相关的系统架构和设计标准，不会涉及系统业务逻辑现实的细节。

## 1.2 背景

首先来看看智慧农业出现之前的农业景象，成绩当然有，那就是粮食产量“十二连增”，蔬菜、水果、肉类、禽蛋、水产品的人均占有量也排在世界前列，但是在这些显著成绩的背后，带来的是一系列的问题，代价不菲。一是化肥农药滥用、地下水资源超采以及过度消耗土壤肥力，导致生态环境恶化，食品安全问题凸显；二是粗放经营，导致农业竞争力不强，出现农业增产、进口增加与库存增量的“三量齐增”现象，越来越多低端农产品滞销。而解决这些问题的最可靠方式就是大力发展智慧农业。

智慧农业的出现是十分及时也是十分必要的，其主要手段是大力发展并运用智能设备、物联网、云计算与大数据等先进技术来开展生产和管理，升级生产领域，由人工走向智能；实现精细化，保障资源节约、产品安全；实现高效化，提高农业效率，提升农业竞争力；实现绿色化，推动资源永续利用和农业可持续发展。因此在可持续发展的道路上，智慧农业的发展越来越广阔，不仅改变了传统农业的许多弊端，而且实现了农业的精细化、高效化、绿色化发展。

在此现状下，为农产品生产者提供一个通用的管理平台是十分必要的。

## 1.3定义

|  |  |  |
| --- | --- | --- |
| 术语缩写 | 术语全称 | 中文翻译(供参考) |
| CRUD | Create/Retrieve/Update/Delete | 增删改查(四种基本的数据操作) |
| BP | Business Process | 业务过程 |
| BO | Business Object | 业务对象 |
| VO | Value Object | 值对象 |
| DAO | Date Access Object | 数据访问对象 |

## 1.4 参考资料

《通用农业物联网数据用户管理平台需求分析》

# 二、总体设计

## 2.1需求规定

### 2.1.1 系统功能

1. 用户注册功能

输入：用户名、密码、邮箱号（必填）、手机号（选填）、验证码

输出：“name\_repeat”/“regist\_success”

前端判断合法性后，使用md5对密码进行加密，然后发送到后端

后端确认之后发送注册成功的邮件给用户

1. 修改密码功能

输入：用户名、新密码

输出："updatepasswd\_success"/"updatepasswd\_error"

使用验证码进行安全判断

发送验证码到邮箱

1. 实时数据查询功能

输入：某个设备 id

输出：实时环境数据列表

使用websocket协议进行前后端长连接，后端主动推送数据到前端

1. 历史数据查询功能

输入：开始时间（秒）、结束时间（秒）、选择设备

输出：历史数据列表

支持查询近一天、近一周、近一月、近一年的数据，并且支持起止时间自定义

1. 用户产业显示功能

输入：用户名、产业id

输出：用户产业信息列表

1. 添加产业功能

输入：用户名、产业名、备注信息

输出：对应industryId（产业id）

1. 删除产业功能

输入：用户名、产业id

输出："delete industry success"/"delete industry fail"

1. 产业采集点显示功能

输入：产业id

输出：采集点信息列表

1. 添加采集点功能

输入：产业id、采集点名称、备注

输出：对应unitId（采集点id）

1. 删除采集点功能

输入：产业id、采集点id

输出："delete acqunit success"/"delete acqunit fail"

1. 采集点设备显示功能

输入：产业id、采集点id

输出：设备信息列表

1. 添加设备功能

输入：产业id、采集点id、设备id、设备名、备注、发送速率

输出：“产业id/采集点id”

1. 修改设备信息功能

输入：产业id、设备id、新设备名、新备注、新发送速率

输出："update deviceinfo fail"/"update deviceinfo success"

1. 删除设备功能

输入：产业id、设备id

输出："delete device success"/"delete device fail"

1. 检查设备入网状态

输入：设备id

输出："已入网"/"未入网"

1. 添加自定义传感器

输入：设备id、传感器站号、485指令、所测数据类型等等

输出："success"/非"success"

1. 添加预设传感器

输入：设备id、传感器站号

输出："success"/非"success"

1. 添加传感器时的连接测试

输入：设备id、传感器站号

输出："success"/非"success"

1. 修改传感器信息

输入：设备id、传感器站号、传感器名

输出："updatename\_success"/"updatename\_fail"

1. 修改传感器检测数据的某一类型的阈值

输入：设备id、传感器站号、数据类型、最大值、最小值

输出："update threshole success"/"delete threshole fail"

1. 删除传感器

输入：设备id、传感器站号

输出："delete sensor success"/"delete sensor fail"

1. 添加继电器

输入：设备id、继电器站号、继电器基本信息

输出："success"/非"success"

1. 修改继电器信息

输入：设备id、继电器站号、更新的信息

输出："success"/非"success"

1. 删除继电器

输入：设备id、继电器站号、强电设备位置

输出："success"/非"success"

1. 添加强电设备

输入：设备id、继电器站号、强电设备位置、强电设备基本信息

输出："success"/非"success"

1. 修改强电设备信息

输入：设备id、继电器站号、强电设备位置、更新的信息

输出："success"/非"success"

1. 删除强电设备

输入：设备id、继电器站号、强电设备位置

输出："success"/非"success"

1. 开启/关闭 某个强电设备

输入：设备id、继电器站号、强电设备位置

输出："success"/非"success"

1. 添加一条规则

输入：设备id、规则具体信息

输出："success"/非"success"

1. 删除一条规则

输入：设备id、规则id

输出："success"/非"success"

1. 设置一条规则的启用状态

输入：设备id、规则id、开关状态  
输出："success"/非"success"

1. 自主调试功能

输入：用户自定义指令

输出：设备返回的信息

对于终端：

1. 终端能够循环获取传感器数据
2. 能够远程控制消息，实现远程控制以及更改发送速率
3. 能够在运行时添加新的传感器
4. 能够控制继电器的下设备的通断电状态
5. 添加日志，记录操作
6. 规则功能（终端自动化控制）
7. 远程自助调试

### 2.1.2 系统性能

1. 实时数据，Web端数据展示性能要求：考虑直接通过WebSocket将MQTT中的数据实时传给Web前端，实时数据直接将网络服务器获取的传感器数据不经过数据库加载到页面。
2. 历史数据加载性能要求，参考InfluxDB+Grafana时序数据库方案历史记录

（十万条）的加载时间。

1. Web端完成继电器设备的控制，通过建立长连接实现Web端与服务器端的双向通信，完成继电器设备的启停控制。
2. 告警事件通知，设置传感器指标的安全范围，当超过安全范围后（持续恶化，排除异常数据引发的告警，例如传感器本身）发出告警消息，控制告警消息的发送频率
3. 终端能够长时间稳定处理传感器数据的获取。数据获取应以锁的方式实现一定的间隔时间，防止传感器的485模块损坏

### 2.1.3 输入输出要求

前端传给后端的数据为json字符串，并按照规定的加密方式进行加密

后端返回的响应数据为json字符串

后端服务器之间使用Feign（http）进行交互

终端和后端的交互数据进行base64加密

### 2.1.4 数据管理能力要求

谨慎使用CRUD操作，在使用CRUD前对输入进行合法性判断。

对数据库进行定期备份。

### 2.1.5 故障处理要求

捕捉程序运行时可能会出现的异常，并对出现故障的事务进行撤销重做。

为程序可能出现的各种意外情况做好出错处理，确保系统能够长时间稳定运行。

对出现的故障和错误做好日志记录，以便用户和开发者查看。

## 2.2 运行环境

### 2.2.1 设备与支持软件

支持浏览器：Chrome，IE，Firefox，360

硬件：GT6502工业嵌入式计算机一台，485传感器若干

## 2.3 系统层次结构

**整体使用B/S架构**

**客户端：**

客户端指的是访问应用的web浏览器终端，通过web浏览器来访问通用物联网平台。

**控制层：**

对用户发来的请求进行解析，根据不同业务将数据分配到不同的业务进行处理。最后将处理完成的数据返回给客户端。

**服务层：**

核心层。它接收控制层分发的请求，完成业务逻辑的具体实现。对不同的业务数据进行处理，处理完成后，将处理结果返回控制层。

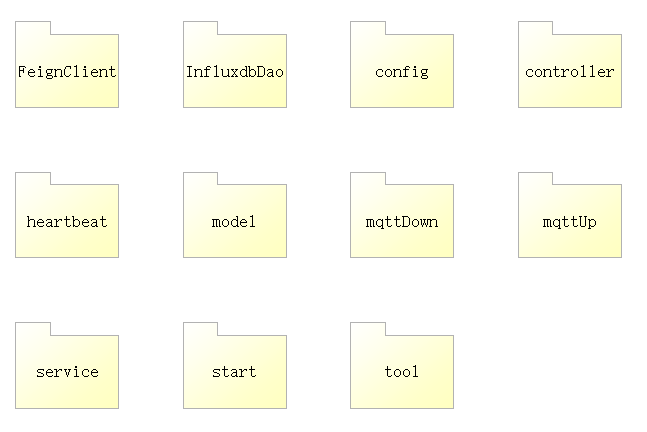
**集成层（DAO 层）：**

集成层向业务层提供统一的内部和外部资源访问，为业务层的数据访问请求屏蔽不同的数据存储访问技术，以及与外部系统整合技术的差异性。

**数据层（资源层）：**

数据层主要指数据库、文件系统和外部系统。

MqttService：



FeignClient包 ：其他微服务提供的接口

InfluxDao包：对influxDB的操作

Config包 ：influxDB和mqtt的一些配置

Controller包 ：提供给其他微服务的接口

Heartbeat包 ：服务器和其他终端设备的连接心跳检测

model包 ：存放所有交互用的类模型

mqttDown包 ：存放用来发送下行信息的相关类

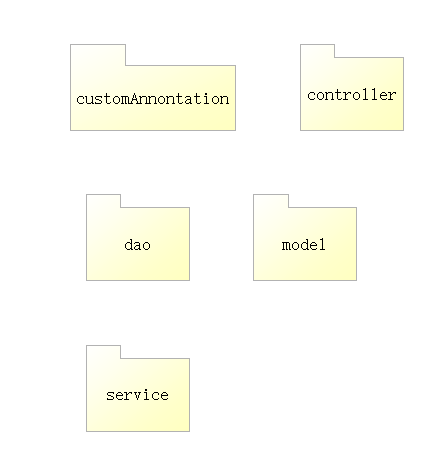
mqttUp包 ：存放用来接收上行信息的相关类

service包 ：存放业务处理类

start包 ：存放自定义启动类

tool包 ：存放用到的帮助（工具）类

MongoService：



CustomAnnontation包 ：存放自定义注解

controller包 ：给其他微服务提供接口

dao包 ：对mongodb的操作

model包 ：存放所有交互用的类模型

service包 ：存放业务处理类

## 2.4 功能需求与程序的关系

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 用户管理 | 历史数据管理 | 实时数据管理 | 邮件管理 | 用户产业设备管理 | 预警管理 |
| 用户注册、登录功能 | √ |  |  | √ |  |  |
| 修改密码功能 | √ |  |  | √ |  |  |
| 添加设备功能（包含产业、采集点、开发板、传感器、继电器、强电设备） |  |  |  |  | √ |  |
| 设备控制功能 |  |  |  |  | √ |  |
| 预警功能 |  |  |  |  |  | √ |
| 用户设备显示功能 |  |  |  |  | √ |  |
| 实时数据查询功能 |  |  | √ |  |  |  |
| 历史数据查询功能 |  | √ |  |  |  |  |

# 三、接口设计

## 3.1 用户接口

详见《接口文档User-Service》

## 3.2 外部接口

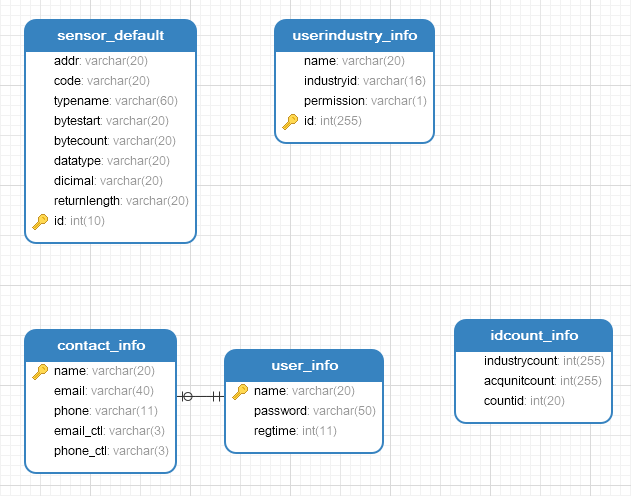
第三方包（使用Maven集成）：

|  |  |
| --- | --- |
| 操作InfluxDB的包 | influxdb-java（2.8） |
| 操作mongodb的包 | spring-boot-starter-data-mongodb |
| 使用mqtt协议的包 | org.eclipse.paho.client.mqttv3（1.2.0） |
| Json格式转换的包 | json-lib（2.2.3） |
| SpringCloud组件 | spring-cloud-starter-netflix-eureka-client  spring-cloud-starter-netflix-eureka-server  spring-cloud-starter-openfeign  （Greenwich.SR1） |
| 使用coap协议的包 | californium-core（2.0.0-M7）  element-connector（2.0.0-M7）  scandium（2.0.0-M7） |

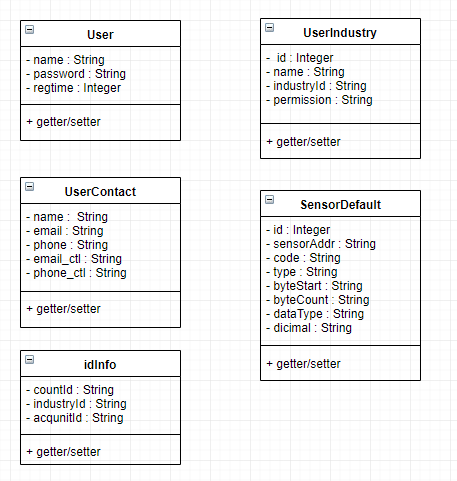
# 四、系统数据结构设计

## 4.1 逻辑结构设计要点

MySQL:

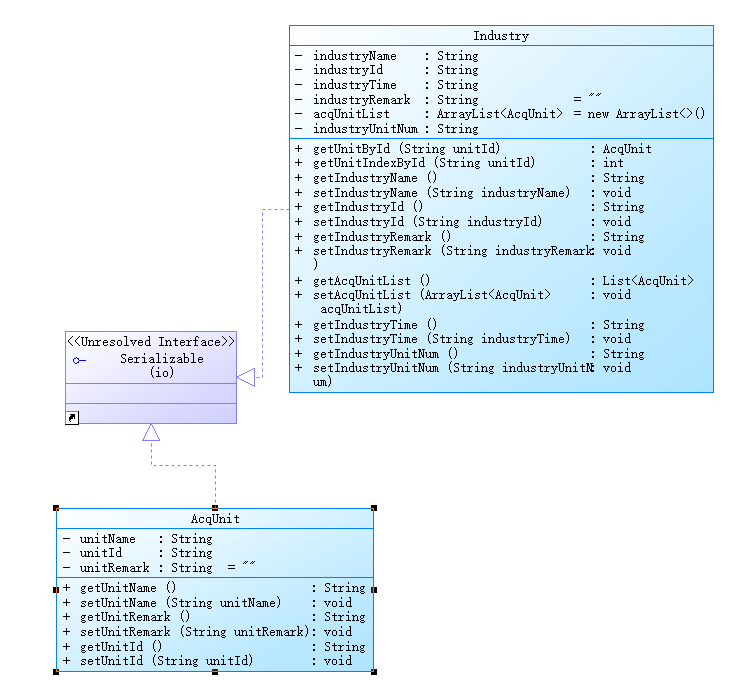


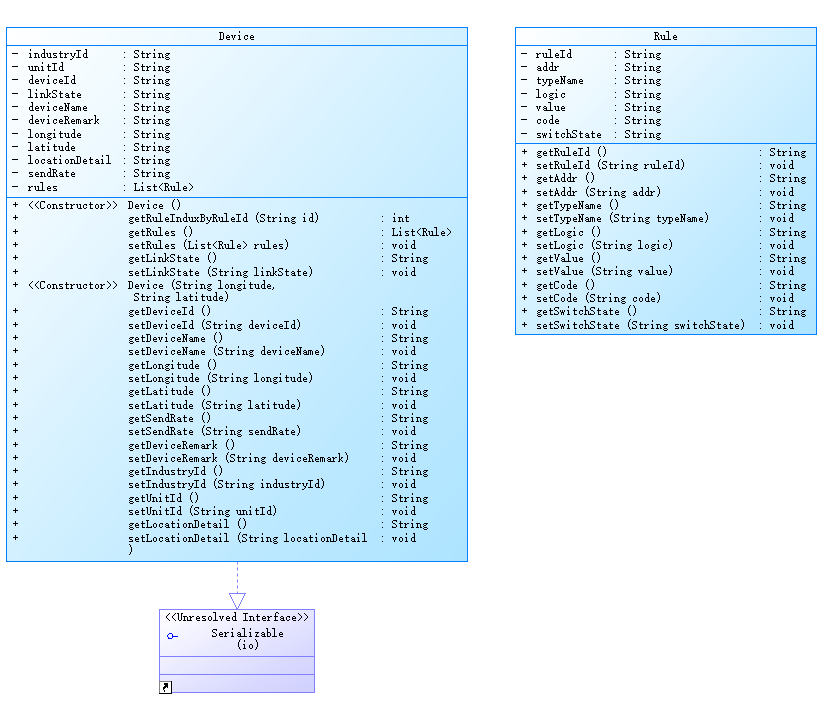
相应类图：

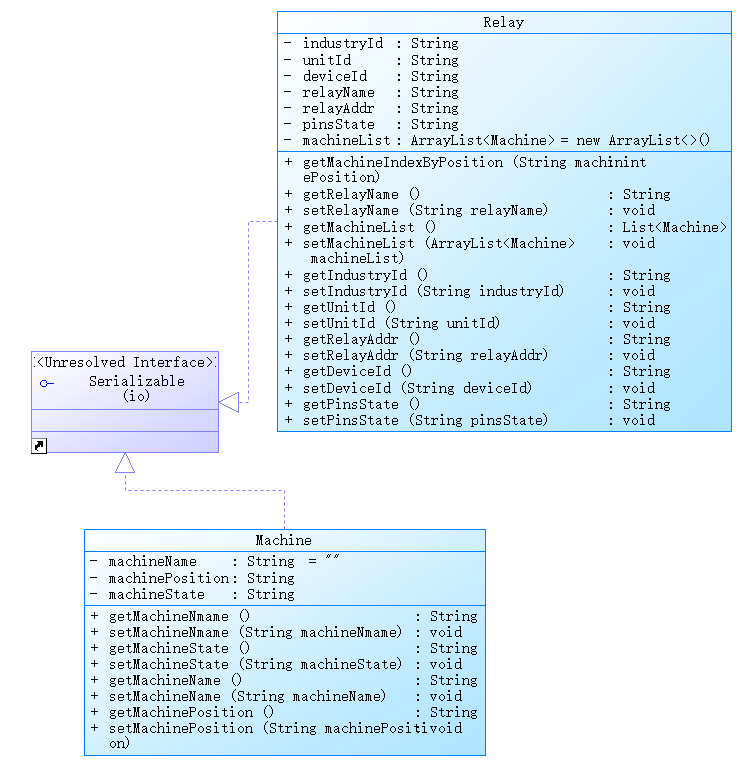


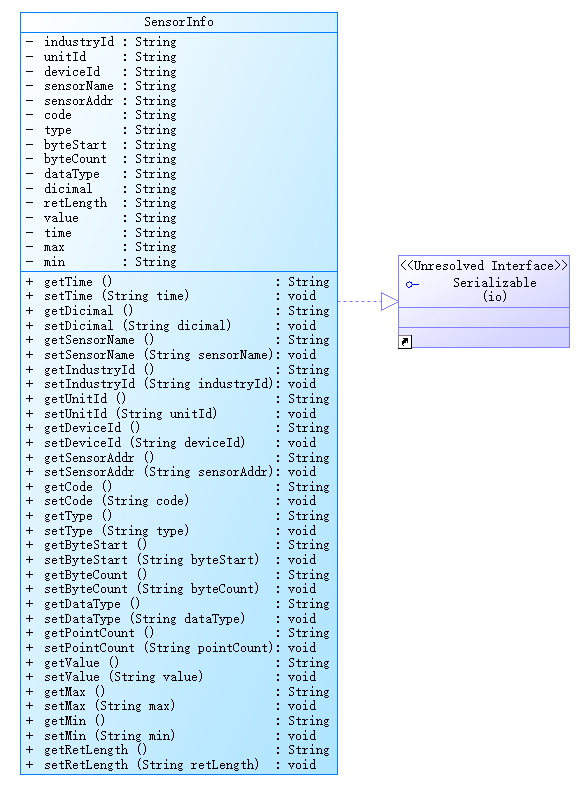
MongoDB：

产业信息对应类图：









InfluxDB中表结构：

1.环境数据表（动态生成）

表名： 设备id\_传感器站号\_所测数据类型

例：6af6188e14aa\_01\_temperature

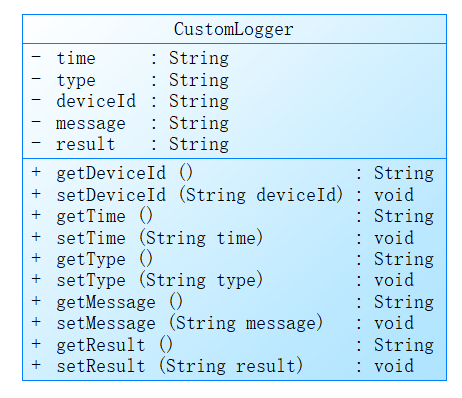
表字段：time value

2.操作日志表（动态生成）

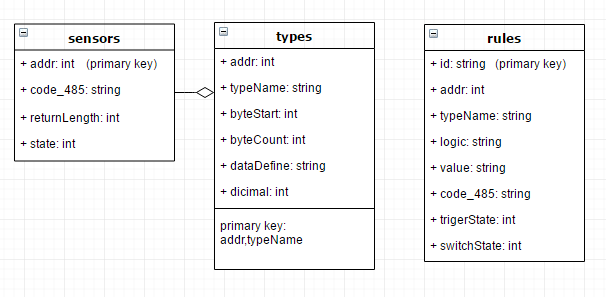
表名： 产业id\_opslog

例： 22\_opslog

对应类：



终端sqlite：



## 4.2 物理结构设计要点

使用Mysql 存储用户、用户对应产业信息、预设传感器信息

使用InfluxDB 存储设备检测到的环境信息

使用MongoDB存储用户拥有的产业所有的信息（包括采集点，设备，传感器等等）

使用Redis做缓存

终端使用 SQLlite 存储设备的传感器信息、传感器值的提取方式、用户自定义规则

Mysql使用默认的Innodb引擎

使用RAID10存储数据

MqttService服务器用来和设备进行通信，同时操作InfluxDB，处理部分业务操作

MongoService服务器用来对MongoDB进行操作，同时处理部分业务操作

UserService服务器用来和用户进行交互，处理绝大部分业务操作，同时给MqttService服务器和MongoService服务器分发任务

# 五、系统出错处理设计

## 5.1 出错信息

|  |  |
| --- | --- |
| 编号 | 出错状况 |
| 1 | 丢失信息 |
| 2 | 系统遭到破坏 |
| 3 | 系统运行效率降低 |

## 5.2 补救措施

1. 对数据库进行备份

2. 用备份文件进行恢复

3. 重新组织数据库

## 5.3 系统维护设计

为了便于维护，应该设计三种日志，系统运行日志，操作日志，出错日志。三种日志根据不同的重要程度采取存放在文件和数据库的方式，系统管理员可以很轻松的监控那个系统的运行情况。数据表的建立和删除由管理员予以维护。