

**吉利汽车集团**

**信息工程部**

数据收集标定平台

详细设计说明书

**<Version 2.0>**

<秘密>

信息工程部保留所有权利。

未经版权所有者的书面许可，禁止通过直接复印机、缩微胶片、静电复印术或任何其他方式以任何形式复制本文任何部分。

**修订及复核记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 修订人 | 修订说明 | 批准人 | 修订日期 |
| V1.0 | 崔晓乐 | 初始版本: 添加XCP功能设计模块和数据转发模块 |  | 2023/08/26 |
| V1.1 | 陆文韬 | 初始版本: 添加边缘计算功能设计模块 |  | 2023/09/26 |
| V1.2 | 陆文韬 | 初始版本: 添加UDS诊断功能设计模块 |  | 2023/10/11 |
| V1.3 | 沈化义 | 更新总体设计章节 |  | 2024/01/09 |
| V1.4 | 徐杰 | 初始版本：添加OTA和Graph功能设计模块 |  | 2024/01/25 |
| V1.5 | 徐杰 | 初始版本：添加设备管理功能设计模块 |  | 2024/01/26 |
| V1.6 | 陆文韬 | 初始版本：补充设备管理功能设计模块,XCP,边缘计算，UDS诊断，UDS 烧写，模版导入导出 |  | 2024/01/26 |
| V1.7 | 彭俊峰 | 初始版本：Trace |  | 2024/01/26 |
| V1.8 | 崔晓乐 | 初始版本：非功能设计 |  | 2024/01/26 |
| V1.9 | 崔晓乐 | 升级目录，优化格式 |  | 2024/02/20 |
| V2.0 | 陆文韬 | 升级目录，英文单词拼写纠错，统一大小写，各级目录校验格式 |  | 2024/02/26 |
| V2.1 | 崔晓乐 | 更改设备信息设置的流程图，更新部署架构图 |  | 2024/4/12 |
| V2.2 | 娄伟 | 添加最新功能说明，更换最新截图 |  | 2024/7/26 |
| V2.3 | 徐杰 | 添加远程诊断功能说明 |  | 2025/03/31 |

**DocapCloud系统详细设计书**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 当前版本号: | Version 2.3 | | | |
| 作者: | 崔晓乐 | | | |
| 创建日期: | 2023-08-15 | | | |
| 最近一次的更新日期: | 2025-03-31 | | | |
| 最近一次的更新人： | 徐杰 | | | |
| 审批人员: | IT领域负责人 | <Name> | 日期: |  |
|  |  |  |  |
|  |  |  |  |

[1 引言 6](#_Toc172908356)

[1.1 目的和范围 6](#_Toc172908357)

[1.2 术语和缩略语 6](#_Toc172908358)

[1.3 参考文档 6](#_Toc172908359)

[2 总体设计 6](#_Toc172908360)

[2.1 系统总体架构确认 6](#_Toc172908361)

[2.2 部署架构 8](#_Toc172908362)

[2.2.1 部署方式 8](#_Toc172908363)

[2.2.2 部署架 构图 8](#_Toc172908364)

[2.2.3 部署架构描述 8](#_Toc172908365)

[2.3 系统总体架构确认 9](#_Toc172908366)

[2.4 应用间接口设计 9](#_Toc172908367)

[3 详细功能设计 10](#_Toc172908368)

[3.1 设备管理功能设计 10](#_Toc172908369)

[3.1.1 设备管理总体结构 10](#_Toc172908370)

[3.1.2 模块及功能点划分 11](#_Toc172908371)

[3.1.3 数据库设计 23](#_Toc172908372)

[3.1.4 关键类设计 29](#_Toc172908373)

[3.1.5 UI/UE设计 30](#_Toc172908374)

[3.2 XCP应用功能设计 35](#_Toc172908375)

[3.2.1 XCP应用总体结构 35](#_Toc172908376)

[3.2.2 模块及功能点划分 35](#_Toc172908377)

[3.2.3 数据库设计 43](#_Toc172908378)

[3.2.4 关键类设计 44](#_Toc172908379)

[3.2.5 UI/UE设计 45](#_Toc172908380)

[3.3 数据转发模块及功能点划分 45](#_Toc172908381)

[3.3.1 数据转发模块总体设计 45](#_Toc172908382)

[3.3.2 模块及功能点划分 46](#_Toc172908383)

[3.3.3 实现描述 46](#_Toc172908384)

[3.4 边缘计算应用功能设计 46](#_Toc172908385)

[3.4.1 边缘计算应用总体结构 46](#_Toc172908386)

[3.4.2 模块及功能点划分 47](#_Toc172908387)

[3.4.3 数据库设计 51](#_Toc172908388)

[3.4.4 关键类设计 52](#_Toc172908389)

[3.4.5 UI/UE设计 52](#_Toc172908390)

[3.5 UDS诊断应用功能设计 55](#_Toc172908391)

[3.5.1 UDS诊断应用总体结构 55](#_Toc172908392)

[3.5.2 模块及功能点划分 55](#_Toc172908393)

[3.5.3 数据库设计 58](#_Toc172908394)

[3.5.4 关键类设计 58](#_Toc172908395)

[3.5.5 UI/UE设计 58](#_Toc172908396)

[3.6 OTA升级应用功能设计 60](#_Toc172908397)

[3.6.1 OTA升级应用总体结构 60](#_Toc172908398)

[3.6.2 模块及功能点划分 61](#_Toc172908399)

[3.6.3 数据库设计 63](#_Toc172908400)

[3.6.4 关键类设计 64](#_Toc172908401)

[3.6.5 UI/UE设计 64](#_Toc172908402)

[3.7 Graph仿真应用功能设计 66](#_Toc172908403)

[3.7.1 Graph仿真应用总体结构 66](#_Toc172908404)

[3.7.2 模块及功能点划分 67](#_Toc172908405)

[3.7.3 数据库设计 69](#_Toc172908406)

[3.7.4 关键类设计 71](#_Toc172908407)

[3.7.5 UI/UE设计 71](#_Toc172908408)

[3.8 UDS FLASH应用功能设计 72](#_Toc172908409)

[3.8.1 UDS FLASH应用总体结构 72](#_Toc172908410)

[3.8.2 模块及功能点划分 73](#_Toc172908411)

[3.8.3 数据库设计 75](#_Toc172908412)

[3.8.4 关键类设计 76](#_Toc172908413)

[3.8.5 UI/UE设计 76](#_Toc172908414)

[3.9 模板导入导出应用功能设计 77](#_Toc172908415)

[3.9.1 模板导入导出应用总体结构 77](#_Toc172908416)

[3.9.2 模块及功能点划分 78](#_Toc172908417)

[3.9.3 数据库设计 80](#_Toc172908418)

[3.9.4 关键类设计 80](#_Toc172908419)

[3.9.5 UI/UE设计 81](#_Toc172908420)

[3.10 Trace应用功能设计 82](#_Toc172908421)

[3.10.1 Trace总体设计 82](#_Toc172908422)

[3.10.2 模块及功能点划分 82](#_Toc172908423)

[3.10.3 数据库设计 83](#_Toc172908424)

[3.10.4 UI/UE设计 84](#_Toc172908425)

[4 外部接口详细设计 84](#_Toc172908426)

[4.1 硬件接口设计 84](#_Toc172908427)

[4.2 软件接口设计 85](#_Toc172908428)

[5 数据迁移 85](#_Toc172908429)

[6 非功能性设计 85](#_Toc172908430)

[6.1 性能设计 85](#_Toc172908431)

[6.2 安全性设计 85](#_Toc172908432)

[6.2.1 强认证 86](#_Toc172908433)

[6.2.2 访问控制最小化 86](#_Toc172908434)

[6.2.3 会话的安全性 87](#_Toc172908435)

[6.2.4 日志安全审计 87](#_Toc172908436)

[6.2.5 软件容错 88](#_Toc172908437)

[6.2.6 数据安全 88](#_Toc172908438)

[6.2.7 输入输出安全性 89](#_Toc172908439)

[6.3 其他非功能性设计 89](#_Toc172908440)

[7 附录 90](#_Toc172908441)

[7.1 【模板】接口清单 90](#_Toc172908442)

[7.2 【模板】数据库清单 90](#_Toc172908443)

# 引言

## 目的和范围

本文档编写的目的是针对数据收集标定平台系统的各个子系统或模块的设计实现进行考虑，为开发人员编码提供基础，设计内容包括总体架构确认，各子系统的详细功能设计、数据库设计和接口设计等。

本文预期的读者包括：开发经理、架构师、项目经理、开发工程师、UI/UE设计师、QA等

## 术语和缩略语

| **术语/缩略语** | **描述** |
| --- | --- |
| 应用服务  （子系统） | 可独立部署的单元，也可称为子系统 |
| 模块 | 模块是从业务维度上对功能点集合的划分，一个模块对应于代码里的一个module目录 |
| 功能点 | 功能点是指通过用户交互触发、外部系统触发或后台程序触发来完成的一个完整的业务场景 |

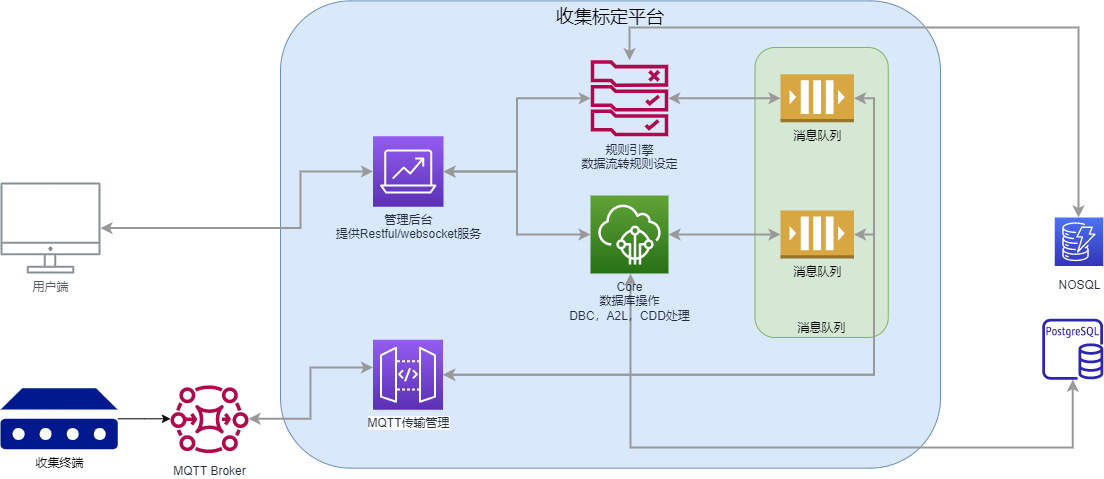
## 参考文档

| **文档名** | **内容简介** | **类别** |
| --- | --- | --- |
| 数据收集标定平台系统架构及概要设计说明书 | 描述数据收集标定平台的整体架构及整体设计 | 依据 |
| 数据收集标定平台系统需求规格说明书 | 描述数据收集标定平台系统的需求规格 | 依据 |
| 接口设计文档 | 描述接口设计的参考规范 | 参考 |

# 总体设计

## 系统总体架构确认

在总体架构中，将主服务拆分成子服务的形式设计和开发，一方面方便进行业务拆分，另一方面，在进行负责均衡时，方便把业务负荷进行分担，使得整体的服务的负荷得以均衡分布，另外，除了逻辑服务以外，还需要相关的第三方服务，以完成分布式部署。



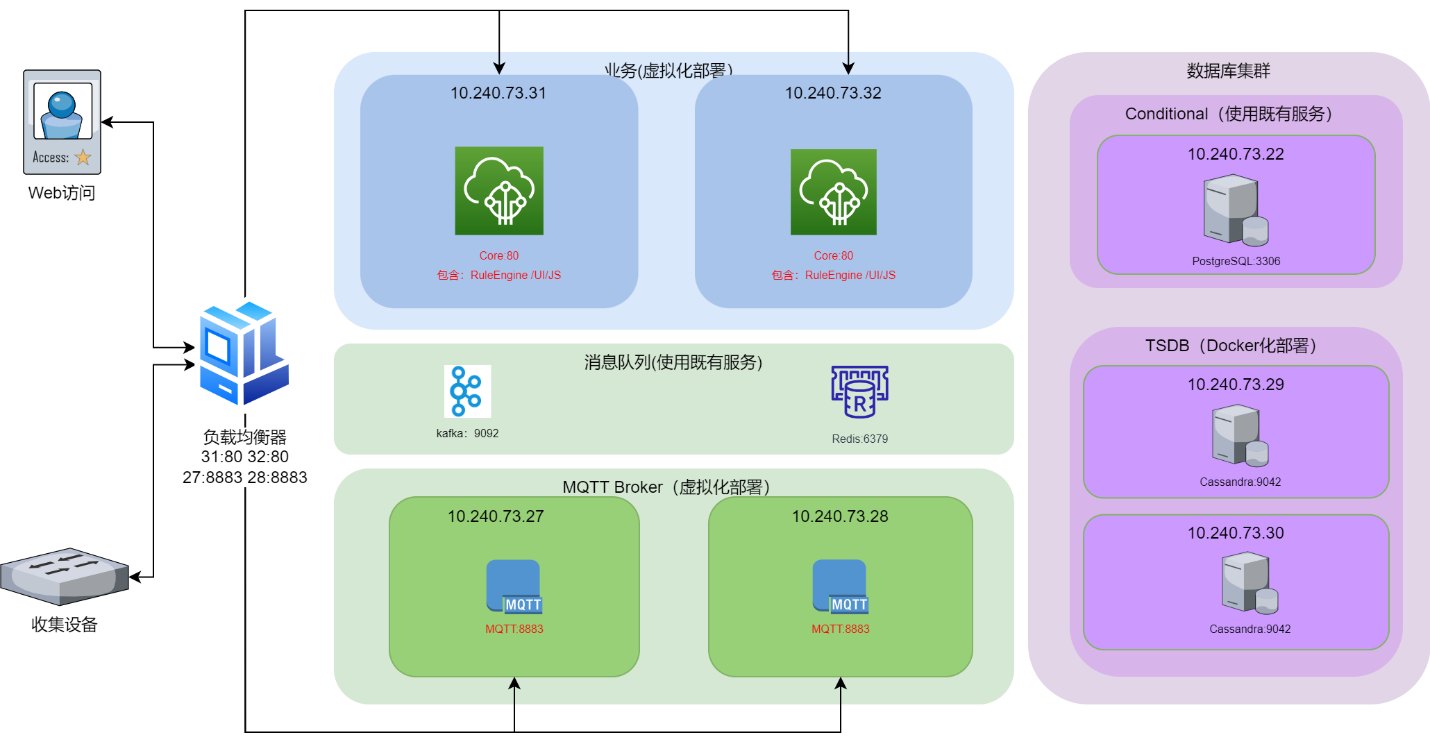
|  |  |  |  |
| --- | --- | --- | --- |
| **逻辑分层（水平分层/垂直分层）** | **应用服务名称** | **应用服务的功能描述** | **开发方式** |
| 云端 | UI | 前端页面 | 自主开发 |
| JS-Executor | JS执行器 | 自主开发 |
| RuleEngine | 规则引擎 | 自主开发 |
| MQTT Broker | MQTT代理 | 自主开发 |
| Core | 负责处理REST API调用和WebSocket订阅 | 自主开发 |
| Kafka | 负责内部信息交互 | 既有服务 |
| Redis | 负责处理REST API调用和WebSocket订阅 | 既有服务 |
| PostgreSQL | 存储逻辑数据 | 既有服务 |
| Cassandra | 存储时序数据 | 第三方服务 |
| NTP Server | 确保设备和服务器之间的时间一致 | 第三方服务 |

## 部署架构

### 部署方式

基于后期维护考虑，进行容器化和虚拟化结合的部署方式，这其中包括各个子服务的容器，以及第三方服务的容器，另外，考虑高可用以及负载均衡，核心服务均部署双份（后期根据接入设备可动态添加）。

### 部署架 构图



注明：docker内部和外部映射端口相同

### 部署架构描述

|  |  |  |  |
| --- | --- | --- | --- |
| **服务名** | **部署方式** | **作用** | **部署位置及端口** |
| Core | 虚拟化 | 【自研服务】负责核心业务处理包括RuleEngine、UI、Js-Executor，负责消息处理，页面呈现，JS执行，REST API以及Web Socket的处理 | 10.240.73.31:80  10.240.73.32:80 |
| MQTT Broker | 虚拟化 | 【自研服务】MQTT服务支持 | 10.240.73.27:8883  10.240.73.28:8883 |
| Kafka | N/A | 【Geely服务】逻辑数据存储 | 10.240.73.XX:9092 |
| Redis | N/A | 【Geely服务】逻辑数据存储 | 10.240.73.XX:6379 |
| PostgreSQL | N/A | 【Geely服务】逻辑数据存储 | 10.240.73.22:3306 |
| Cassandra | 容器化 | 【第三方服务】时序数据存储  包含热备 | 10.240.73.29:9042  10.240.73.30:9042 |

## 系统总体架构确认

系统总体架构，参考Basic Design文档。

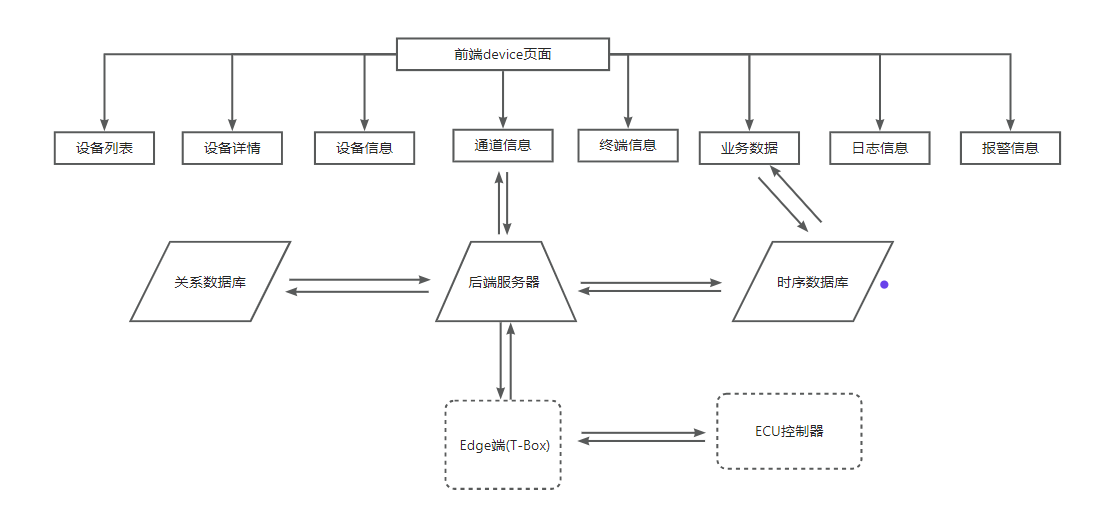
## 应用间接口设计

应用间接口，可以参考DocapCloud的Swagger接口文档。

# 详细功能设计

## 设备管理功能设计

### 设备管理总体结构

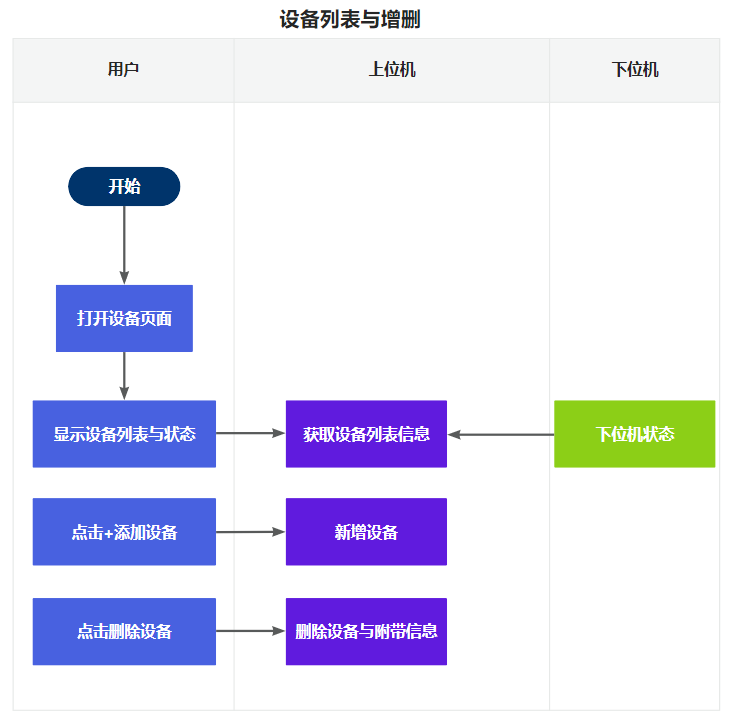


### 模块及功能点划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块代码** | **模块名称** | **功能点编号**  **（系统需求ID）** | **功能点名称**  **（系统需求名称）** | **功能点简要描述** |
| MOD-TBOX-001 | 设备列表 |  | 展示当前租户下的所有设备的列表以及添加删除设备功能 | 当前租户可以看到所有设备列表，包括设备名称，设备配置名称，设备状态，同时可以添加设备，删除设备 |
| MOD-TBOX-001 | 设备详情 |  | 展示和编辑当前设备的配置文件，OTA软件包，访问Token | 选择某个设备后的该设备详情，可以复制Token，编辑配置文件，分配OTA包，分配客户，更改权限 |
| MOD-TBOX-001 | 设备信息展示 |  | 展示当前设备通道端口信息以及状态，和设备经纬度等遥测信息 | 选择某个设备后的设备多路CAN信息及状态展示，设备基本信息展示，包括经纬度，QDR校准度，软硬件版本等 |
| MOD-TBOX-001 | 设备信息配置 |  | 设置时间以及边缘计算缓存时间 | 可以将时间模式设置为手动或者自动并调整边缘计算缓存时间，警告上传，日志上传，上传级别默认不可修改。 |
| MOD-TBOX-001 | 通道信息设置 |  | 用户可以上传某个通道的配置文件，配置波特率和功能信息，开启CAN的原始帧采集，修改LIN,ETH配置 | 上传某个通道的DBC，A2L，CDD文件，选择DBC信号列表以发往Kafka，配置通道波特率与CAN/CANFD，开启原始帧采集 |
| MOD-TBOX-001 | 终端信息 |  | 用户可以配置VIN,BMS,PCM,ECM等参数，并浏览解析数据 | 通过加载的CDD,DBC文件，可以选择信号和DID配置对应参数从而浏览版本号等信息。 |
| MOD-TBOX-001 | 业务数据 |  | 展示CAN，XCP，UDS的业务数据 | 选择精确到毫秒的时间段，选择CAN，XCP，UDS某个业务，查询出时间段内的数据，同时CAN数据支持导出为.mf4文件 |
| MOD-TBOX-001 | 日志信息 |  | 用户可以浏览下位机的日志 | 通过日志页面的控件可以筛选特定时间的日志浏览 |
| MOD-TBOX-001 | 报警信息 |  | 用户可以浏览下位机产生的告警信息 | 通过告警页面的控件可以筛选特定时间特定等级的告警信息 |
| MOD-TBOX-001 | 睡眠唤醒设置 |  | 用户可以设置下位机睡眠以及唤醒配置 | 选择是否睡眠填写睡眠时间，CAN唤醒只支持CAN1/CAN2调整为指定帧或者任意帧，指定帧时可以输入(例:oxoo1),RTC唤醒可以指定时间间隔或者指定时间将下位机唤醒 |

#### 设备列表功能点

1. 流程图：



1. 涉及表和字段信息：

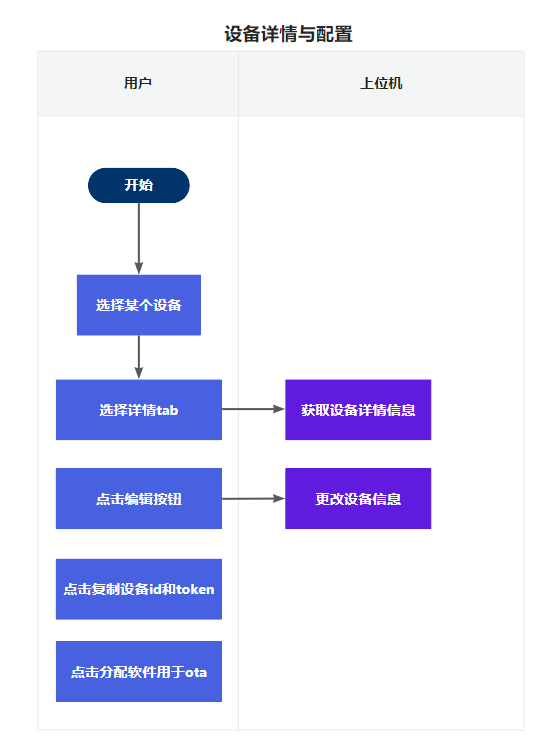
Table：device, attributes\_kv

Column: id, name, tenant\_id, device\_profile\_id, active

1. 实现思路：
2. 通过CRUD新建设备与删除设备，新建device\_info数据库视图从attributes\_kv表的entity\_id = device\_id and attribute\_key = active获取设备列表与状态。
3. 通过Java netty高性能NIO框架实现MQTT协议的连接，每个连接session包括了device\_id。

#### 设备详情功能点

1. 流程图：



1. 涉及表和字段信息：

Table：device\_profile, device\_credentials

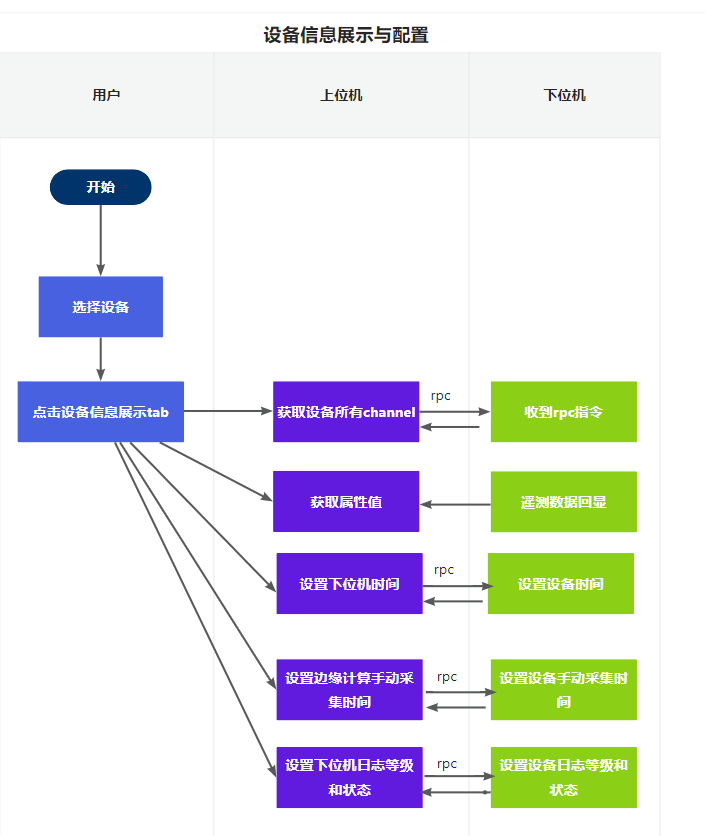
Column: name, profile\_data, default\_rule\_chain\_id, description, token

1. 实现思路：

根据device\_id展示device\_profile, device\_credentials的信息，点击复制设备id和token复制对应的id和credentials，展示ota\_package选项列表，分配对应的package

#### 设备信息展示与配置功能点

1. 流程图：



1. 涉及表和字段信息：

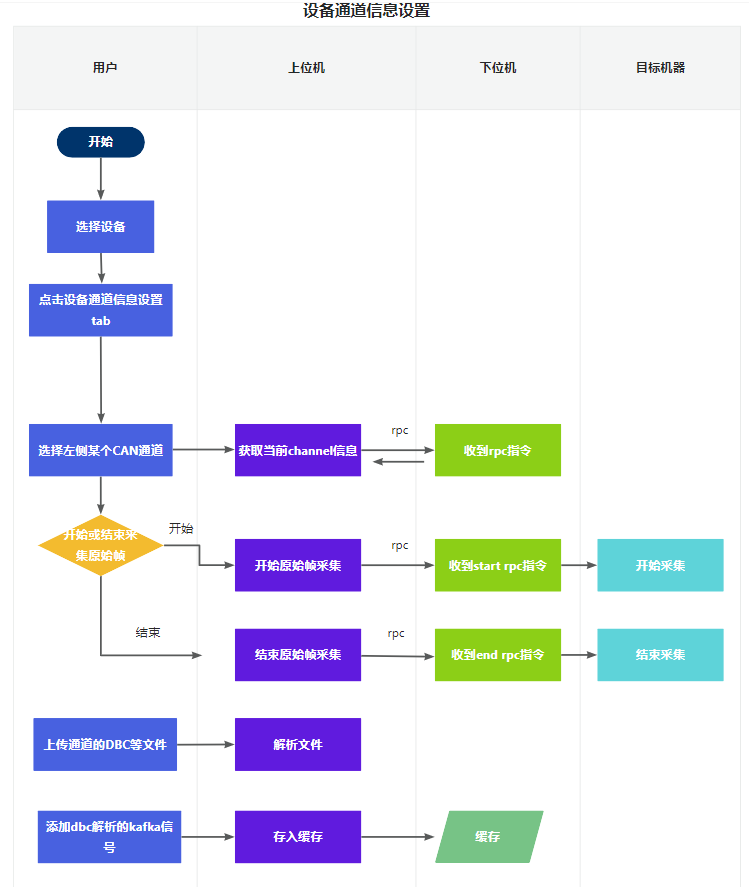
Table: dbc\_channel, attributes\_kv

Column: channel\_id, channel\_name, attribute key, value1

1. 实现思路：
2. 根据device\_id查询dbc\_channel下的channel list，通过RPC向下位机下发所有channel的getChannelStatus指令获取每个channel状态。
3. 从attribute\_kv表根据entity\_id=device\_id查出所有需要的key对应的value值，比如经纬度等并展示。
4. 获取MQTT端口ip是全局的，用户名密码从device\_credentials中获取，MQTT ca证书可以从本地上传。

#### 通道信息设置功能点

1. 流程图：

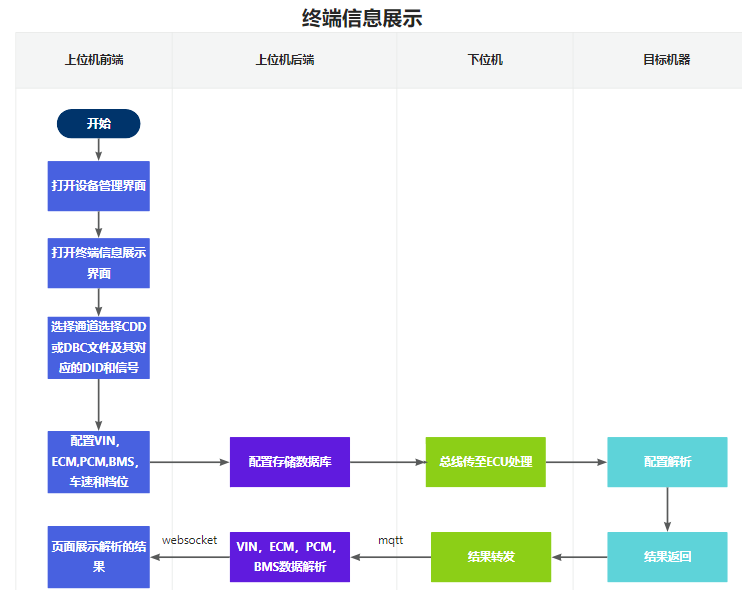


1. 涉及表和字段信息：

Table：dbc\_channel, a2l\_measurement, dbc\_signal, dbc\_message

1. 实现思路：
2. 通过选择的channel name获取channel信息，通过RPC调用获取CAN状态。
3. 通过RPC指令完成下位机原始帧采集功能。
4. 上传DBC,A2L,CDD根据相关协议框架进行解析并存入对应的数据库表。

#### 终端信息展示功能点

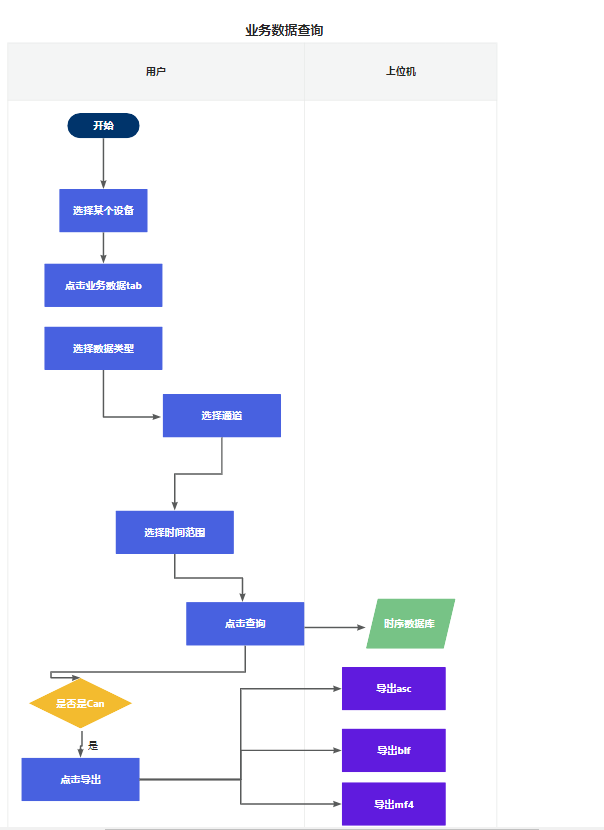
1. 流程图：
2. 涉及表和字段信息：

Table：terminal\_info

1. 实现思路：
2. 先用户在页面上配置的VIN,ECM,PCM,BMS,车速和档位配置存储在后端数据库
3. 随后将配置下发给下位机，交由下位机接触处理
4. 上位机等待下位机将处理完的数据结果通过MQTT发送至对应topic，一旦有数据上传至上位机，便通过先前数据库存储好的CDD文件解析对应数据，用Websocket把解析完的结果（VIN码ECM,PCM,BMS版本号）交由前端展示。

#### 业务数据功能点

1. 流程图：



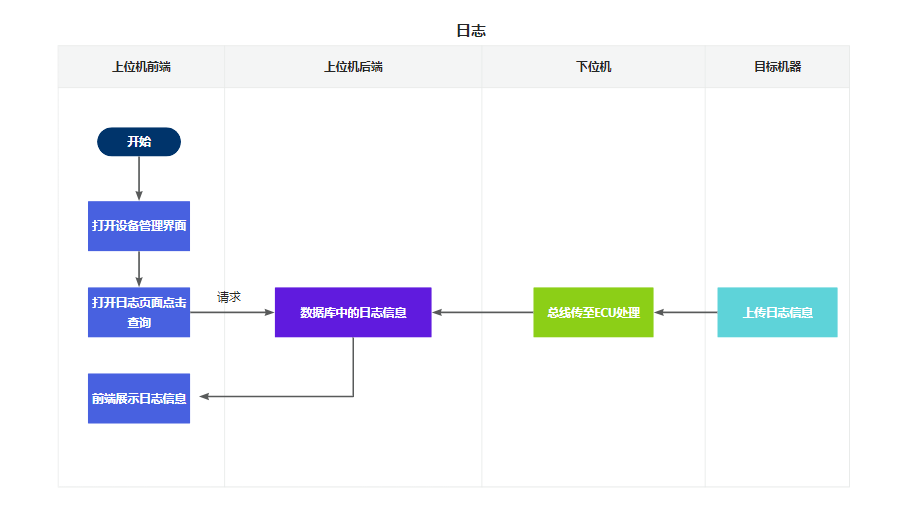
1. 涉及表和字段信息：

Table：grc\_can\_frame\_bytes\_cf, grc\_lin\_frame\_bytes\_cf, grc\_uds\_cf, grc\_xcp\_cf

1. 实现思路：
2. 通过根据查询条件查询Cassandra时序数据库得到原始数据，再进行解析发给前端。
3. 点击导出后先将文件分片上传至minio服务器，根据返回fileName,调用接口请求到对应文件的输出流后，前端通过streamsaver组件的serviceworker模仿长连接边读取缓冲区边写文件，达到文件下载的目的。

#### 日志信息功能点

1. 流程图：



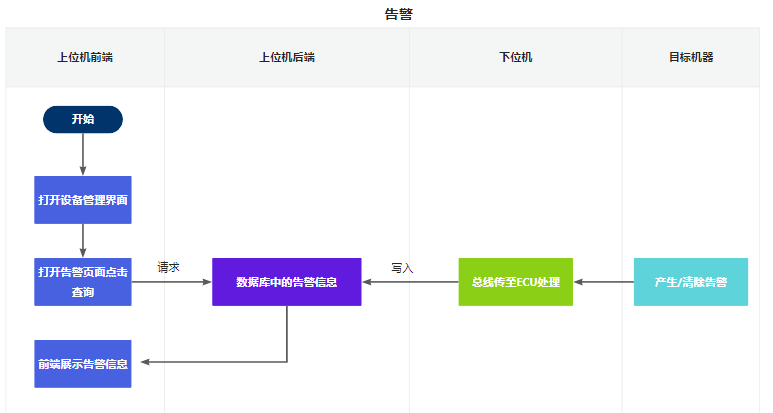
1. 涉及表和字段信息：

Table: log

Column: device\_id, level, module, tip

1. 实现思路：
2. 下位机通过MQTT连接上云端，实时发送log日志，通过订阅MQTT的log指定topic完成上发。
3. 云端需要在规则链新建logParseNode节点来读取MQTT信道里的topic内容，存入log表。
4. 云端后端暴露查询log表的REST API给前端以调用。

#### 报警信息功能点

1. 流程图：
2. 涉及表和字段信息：

Table：grc\_alarm

Column: alarm\_id,clear\_time,device\_id,last\_time,message,number,reason,status,tenant\_id

1. 实现思路：
2. 下位机通过MQTT传输将设备主动发出的告警或清除告警的信息转发给上位机后端并写入其数据库。
3. 当用户点击告警页面按一定筛选条件查询时，前端会去请求之前写入数据库的告警信息并展示在页面上。

#### 睡眠唤醒设置

1. 流程图:

图示

描述已自动生成

1. 涉及表和字段信息:

Table：device\_sleep

Column：device\_id,if\_sleep,sleep\_time,status,can\_id,status2,can\_id2,created\_time,rtc\_mode,rtc\_value

1. 实现思路:

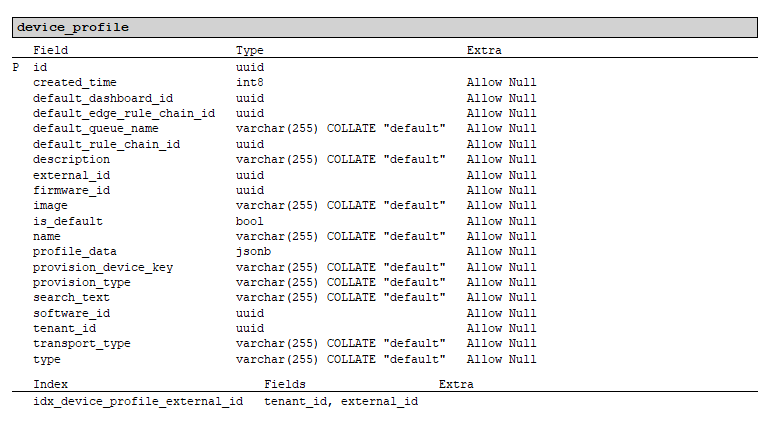
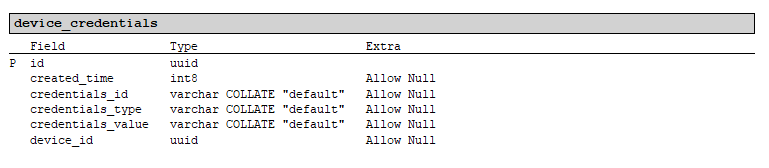
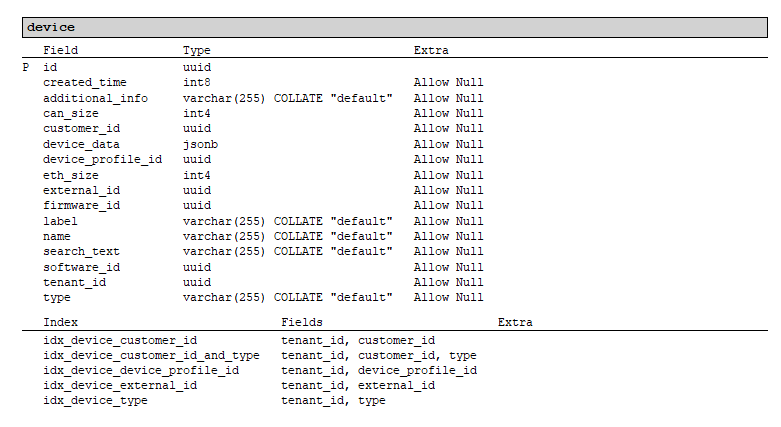
用户每次进入页面时，调用一次RPC获取下位机睡眠唤醒设置后给前端赋值，用户在操作是否睡眠时，只有是才能输入数字（单位为秒），设置CAN唤醒分为任意帧和指定帧，在任意帧时不允许输入帧ID，选择指定帧时可以输入数字或0x＋数字(会转换成0x…),RTC唤醒分为唤醒时间和唤醒间隔，在选择唤醒时间时需要选择固定时间，选择唤醒间隔时需要输入秒数

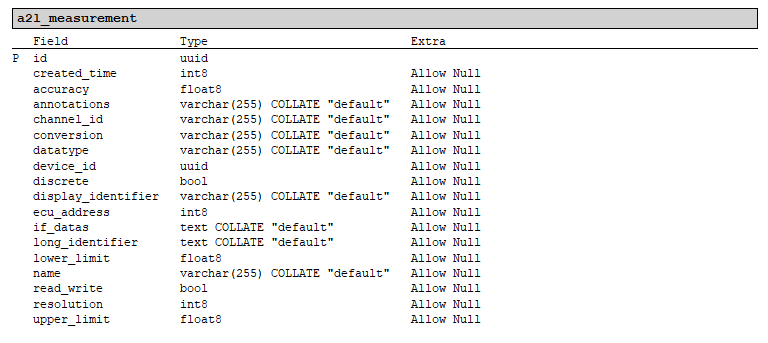
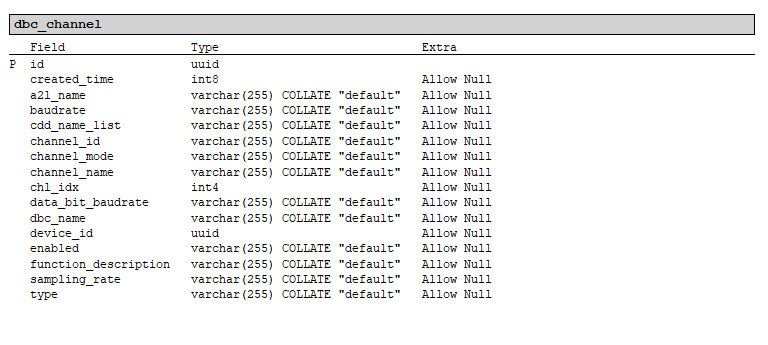
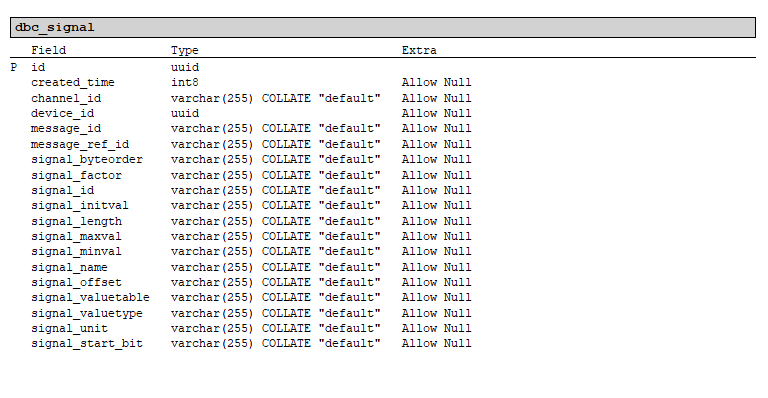
睡眠是指在输入的时间内没有接收到任何信息即下位机掉电

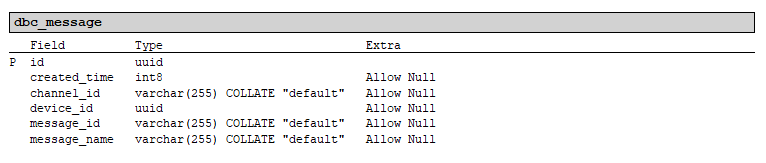
唤醒是指在下位机掉电情况下接收到帧信息或在指定的时间、特定的间隔下唤醒上电

### 数据库设计

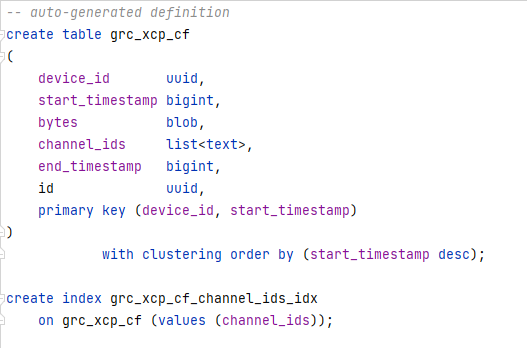
1. 设备相关表

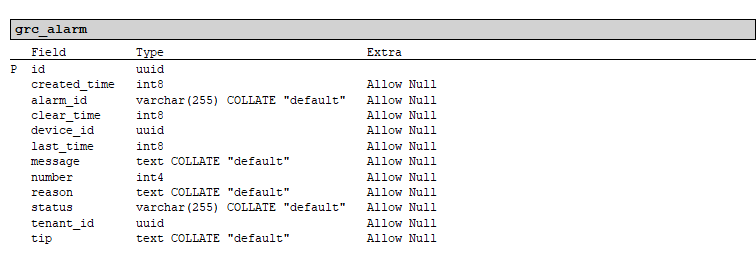


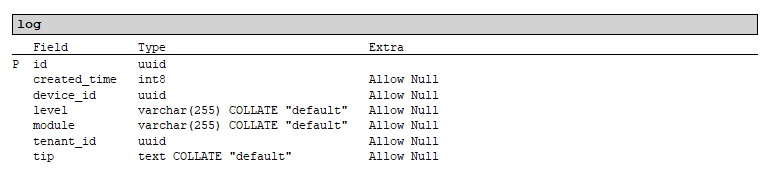
2.通道信息相关表  

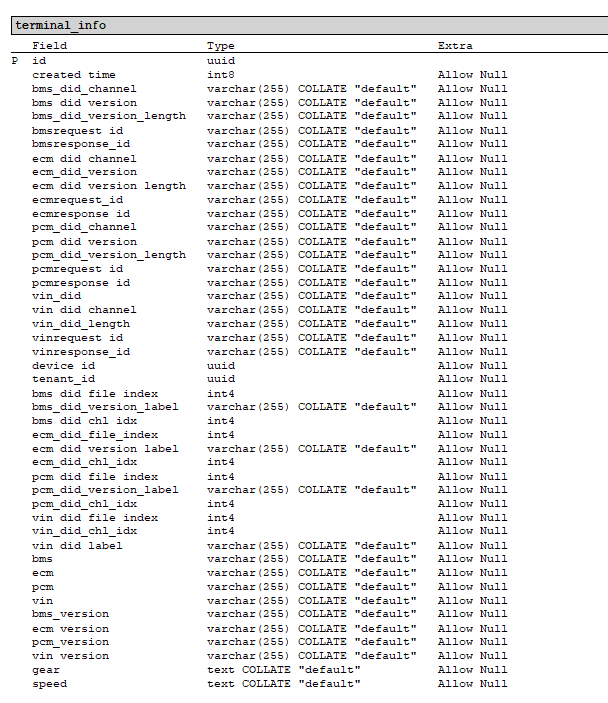


3.业务数据相关表



4.日志报警信息相关表 





5.睡眠唤醒相关数据表

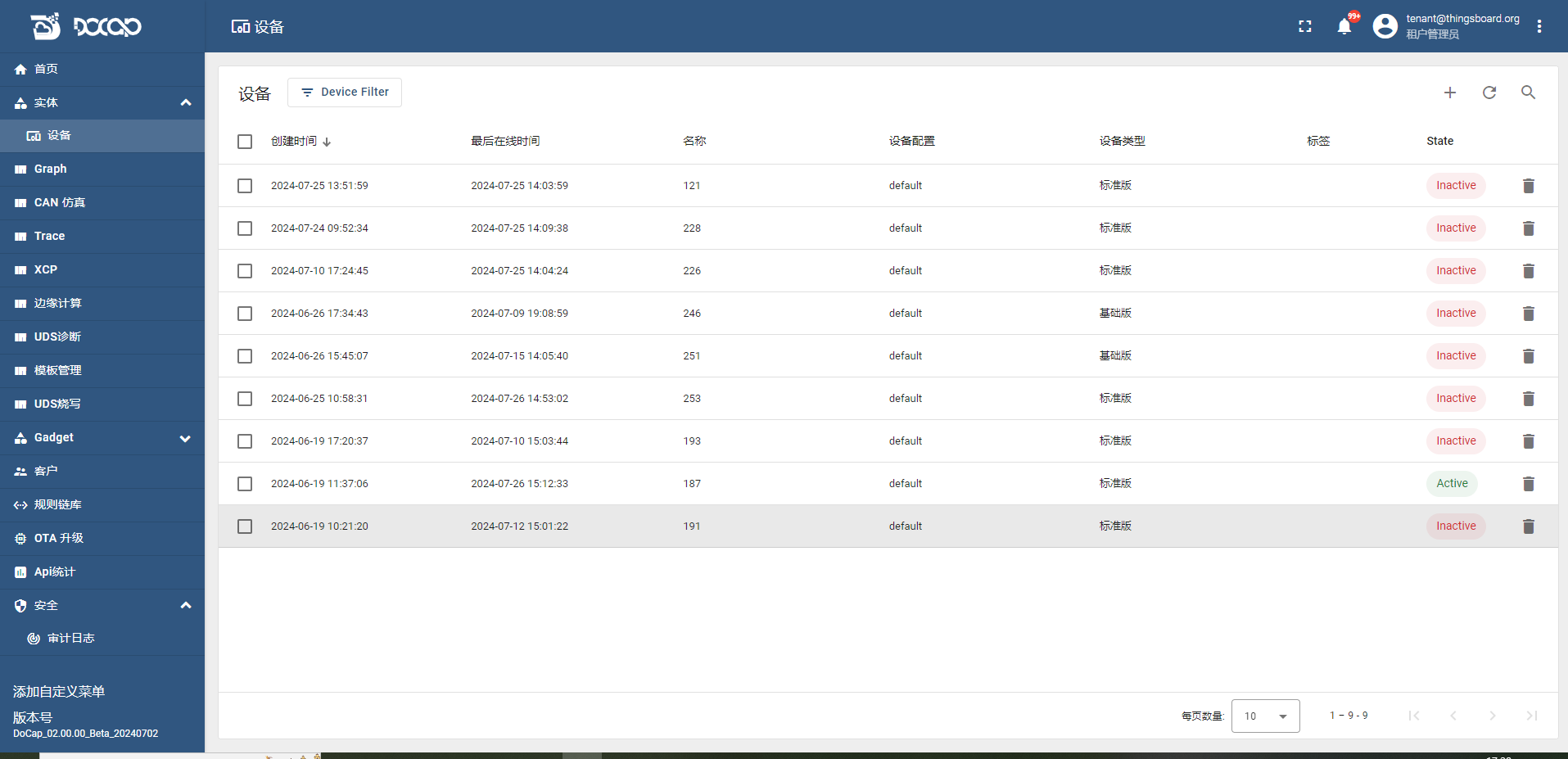


### 关键类设计

DeviceActor.java, DeviceActorProcessor.java, DbcChannelService.java, a2lParser package, LogService.java, CassandraBaseTimeseriesDao.java, TbRpcService.java

### UI/UE设计

1.设备列表页面



2.设备详情页面

图形用户界面, 应用程序, 电子邮件

描述已自动生成

3.编辑设备详情

图形用户界面, 应用程序, 电子邮件

描述已自动生成

4.设备信息展示页面

电脑软件的截图

描述已自动生成

5.设置设备信息

图形用户界面, 应用程序

描述已自动生成

5.通道信息展示页面

图形用户界面, 应用程序

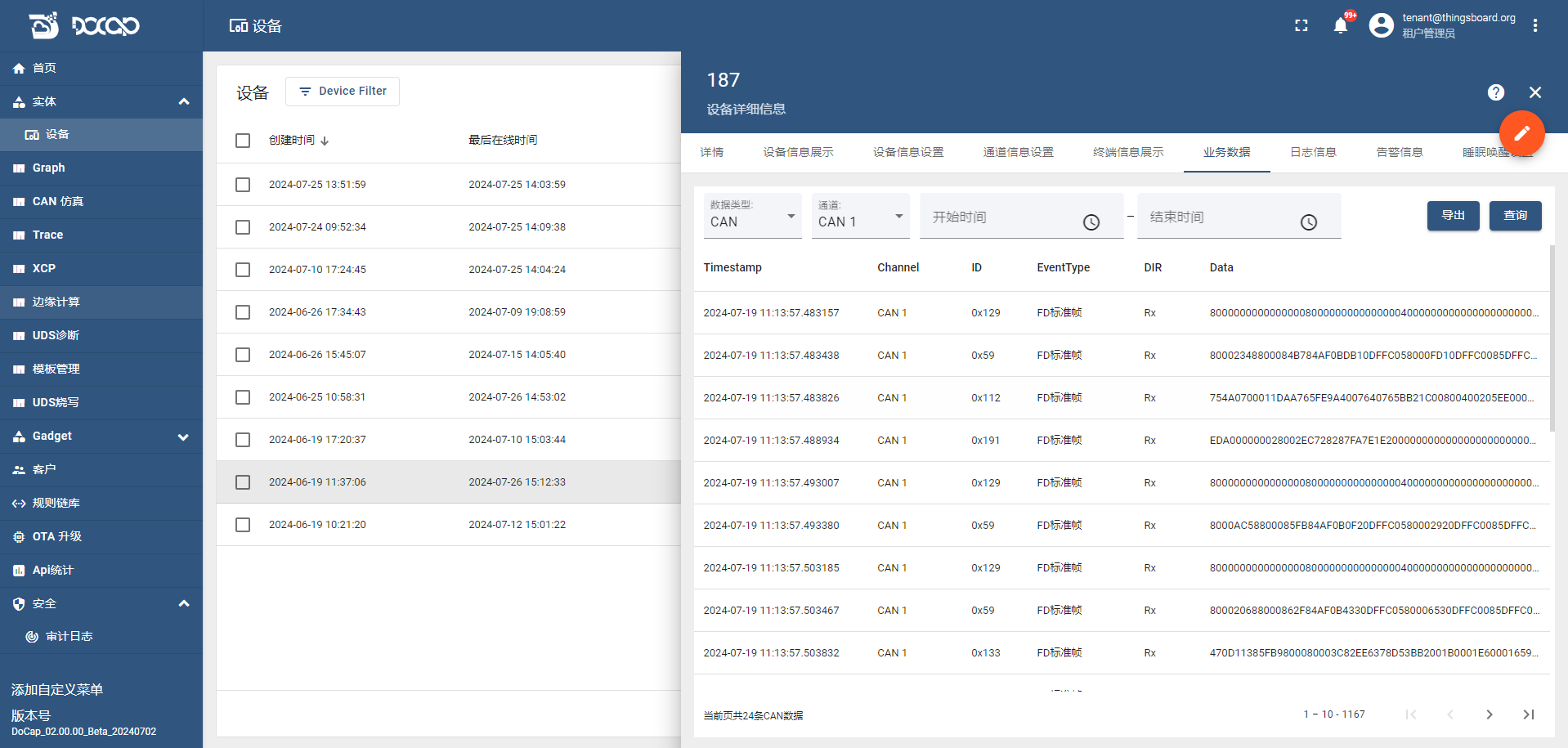
描述已自动生成

6.添加DBC等文件与添加解析后发往消息队列的列表

图形用户界面, 应用程序

描述已自动生成

7.业务数据查询



8.日志报警查询

电脑软件截图

描述已自动生成

电脑软件截图

描述已自动生成

9.睡眠唤醒设置

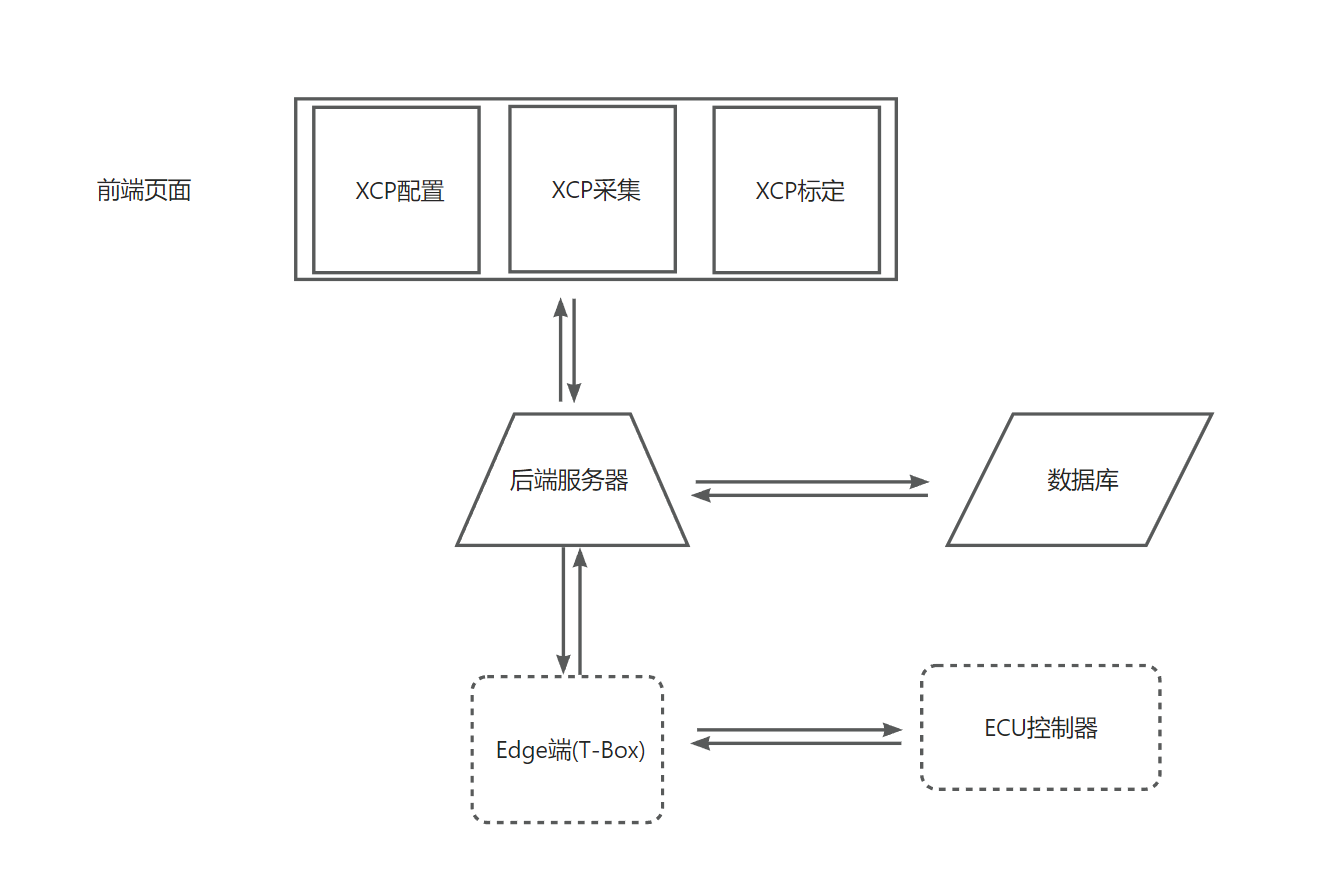
图形用户界面, 应用程序

描述已自动生成

## XCP应用功能设计

### XCP应用总体结构

（Edge端(T-Box)和ECU控制器是辅助说明内容，非本模块功能部分）



### 模块及功能点划分

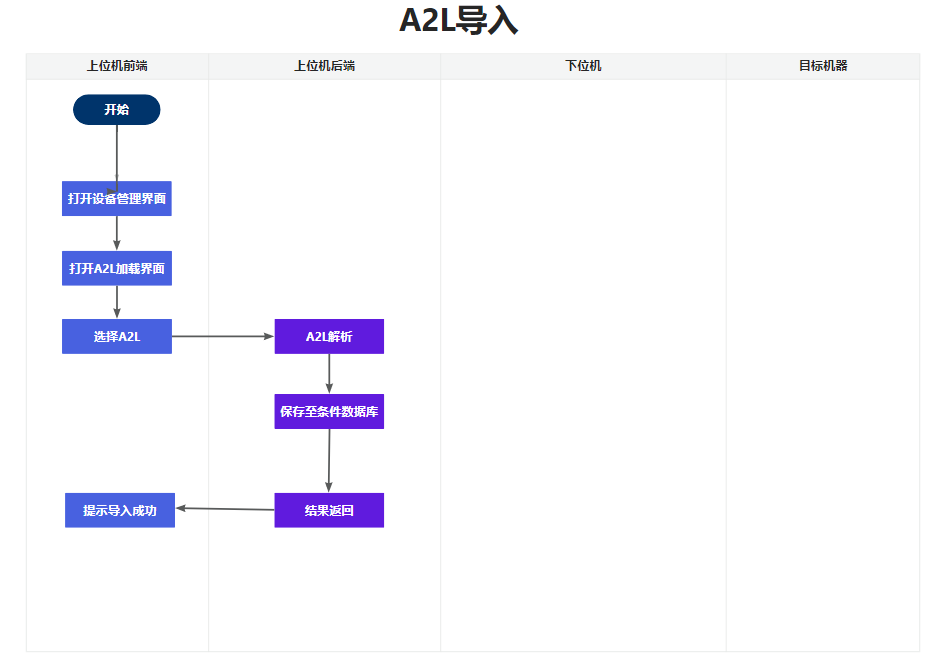
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块代码** | **模块名称** | **功能点编号**  **（系统需求ID）** | **功能点名称**  **（系统需求名称）** | **功能点简要描述** |
| MOD-TBOX-001 | XCP |  | XCP配置 | 加载XCP采集和标定需要的A2L文件 |
| MOD-TBOX-001 | XCP |  | XCP采集 | 显示需要采集的XCP变量 |
| MOD-TBOX-001 | XCP |  | XCP标定 | 用于下发XCP标定量 |

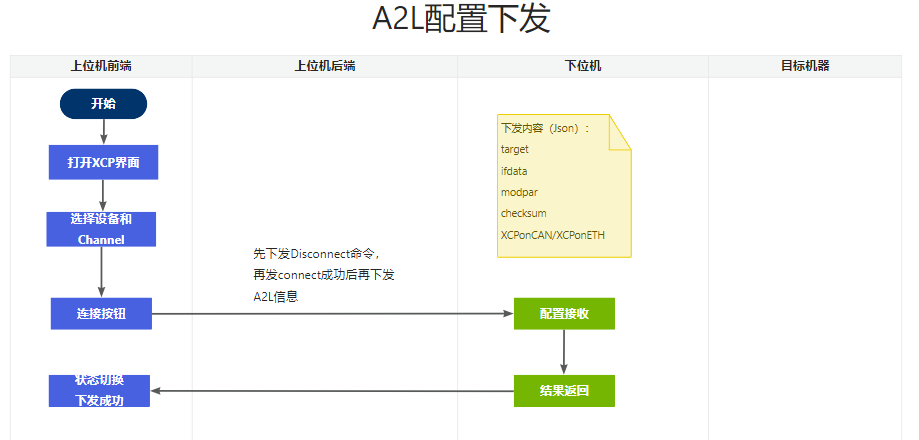
***功能点的描述、业务规则、前置条件、处理流程、输入描述、输出描述等信息详见SRS文档。***

#### 功能点逻辑设计

##### XCP配置功能点

1. 流程图：





1. 涉及表和字段信息：

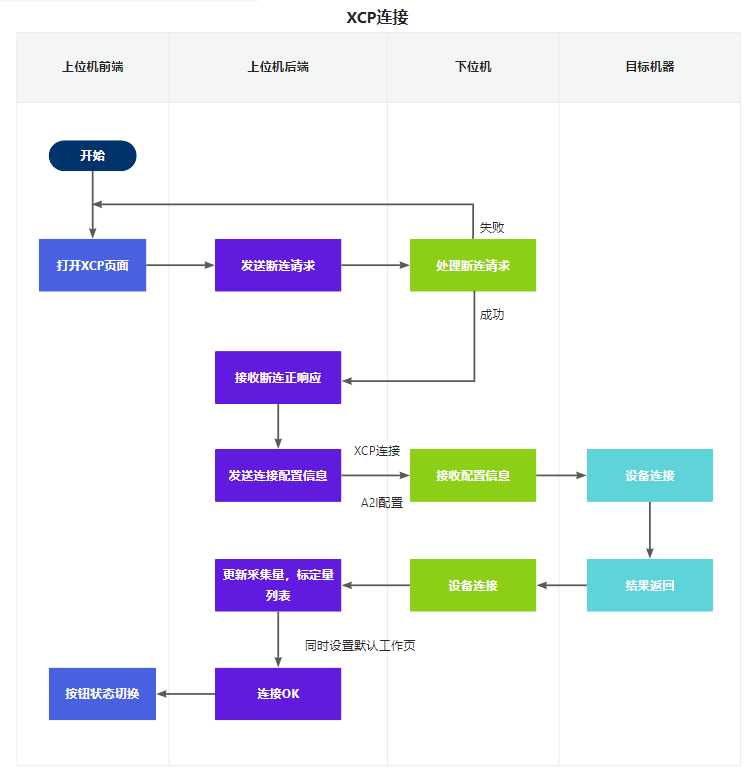
Table：a2l\_json, a2l\_characteristic, a2l\_measurement

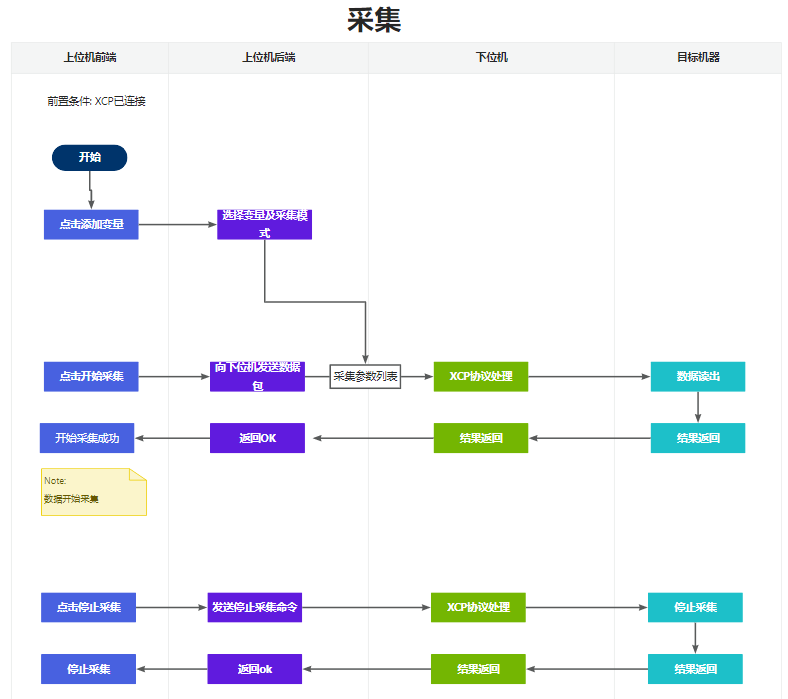
Column: channel\_id(重要)，以及三张表其余相关字段

1. 实现思路：
   1. 此功能点主要是 加载A2L到数据库 和 下发A2L信息到边缘端
   2. 加载A2L功能在设备的对应通道中，加载完后A2L的信息就会保存到数据库中，与设备的通道信息绑定
   3. 点击连接按钮时下发A2L是为了完成XCP采集和标定需要的基本信息。通过XCP界面，选择设备和通道后，点击连接按钮，与设备连接成功后，会把A2L中必要的信息下发到边缘端，以供后边的XCP采集和标定做准备。注意: 这里下发A2L前默认会给边缘端发送一个XCP Disconnect指令。防止边缘端还在接收采集量。之后再发送一个XCP Connect指令与下位机建立连接。

##### XCP采集功能点

1. 流程图





1. 涉及表和字段信息：

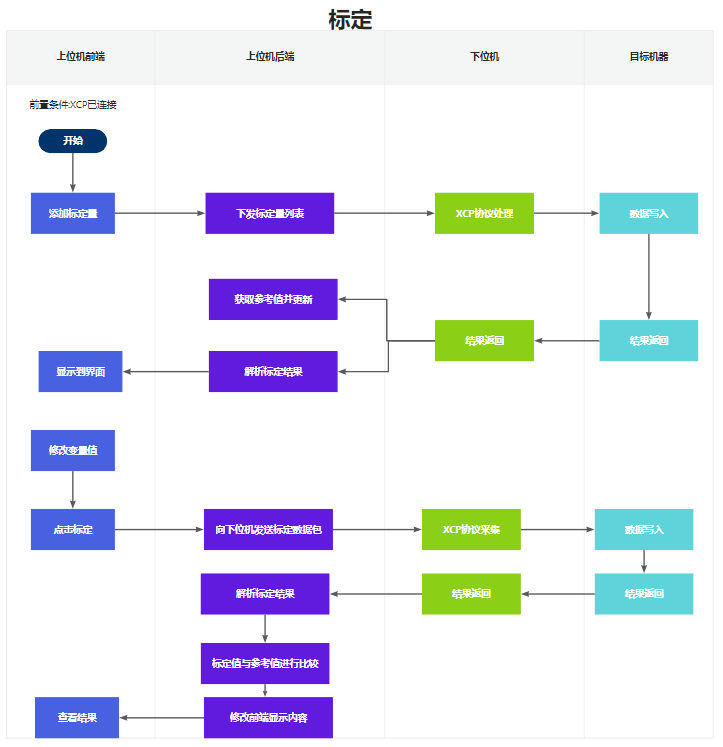
Table：a2l\_json, a2l\_characteristic, a2l\_measurement

Column: channel\_id(重要)，以及三张表其余相关字段

1. 实现思路:
   1. XCP的采集需要先连接设备，才能继续下一步操作。通过点击连接按钮，就可以下发XCP连接指令，连接成功后，界面的按钮状态就会改变，并回显历史添加的采集量和标定量。
   2. 连接成功后，就可以添加采集量和标定量，点击添加采集量按钮就可以从弹出的选择变量窗口中添加XCP采集量，点击开始采集，数据就会从边缘端发送到云端界面。

##### XCP标定功能点

1. 流程图



1. 涉及表和字段信息：

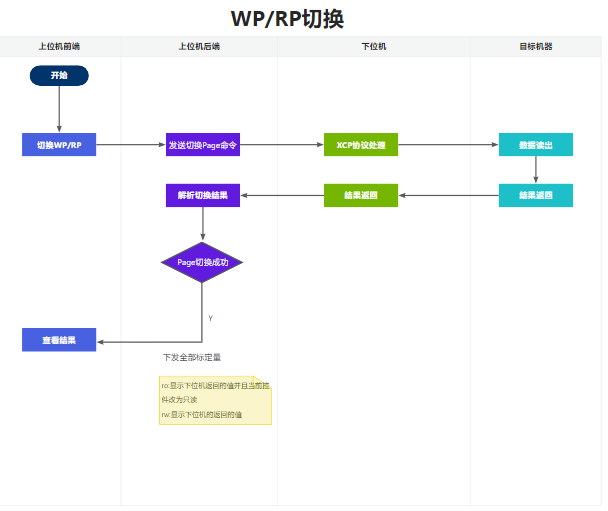
Table：a2l\_json, a2l\_characteristic, a2l\_measurement

Column: channel\_id(重要)，以及三张表其余相关字段

1. 实现思路
   1. XCP连接成功后，就可以点击添加标定量按钮，在弹出的选择变量窗口添加标定量，关闭添加标定量窗口，就会自动下发选择的所有标定量。
   2. 下发标定量列表后，边缘端会返回标定量ro,rw的属性，并且根据参考值更新界面上变量的状态，rw的变量可以编辑，且可以标定，ro的变量为不可编辑，不可以标定

##### 工作页、参考页切换

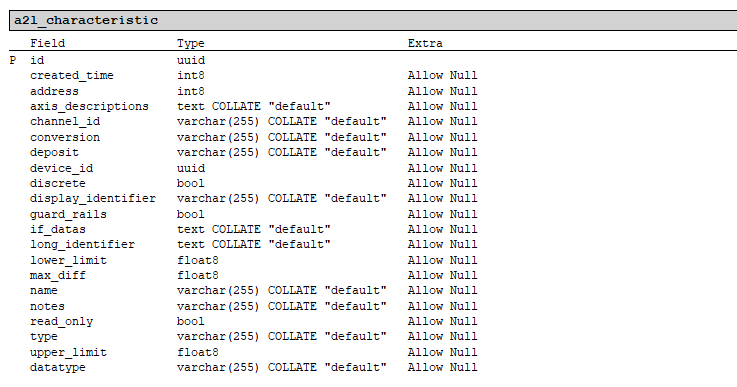
1. 流程图

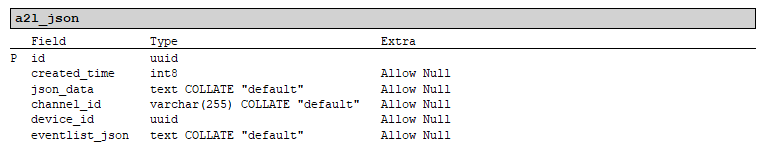


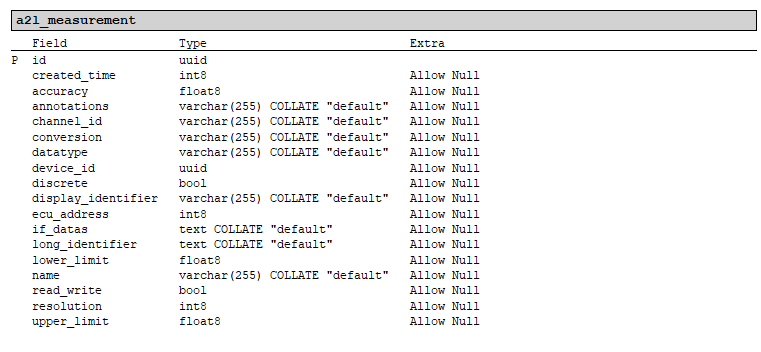
1. 涉及表和字段信息：无
2. 实现思路
   1. WP/RP的切换通过两个按钮切换，目前默认WP的pageNum为0， RP的pageNum为1下发标定量列表后，边缘端会返回标定量ro,rw的属性，并且根据参考值更新界面上变量的状态，rw的变量可以编辑，且可以标定，ro的变量为不可编辑，不可以标定
   2. 点击WP按钮会发送切换WP命令，page切换成功后，会再发下发标定量列表给边缘端，边缘端处理完成后，会返回标定量的ro,rw属性，然后界面会根据对应的变量来处理变量是可以标定的，还是不可以标定的。
   3. 点击RP按钮会发送切换RP命令， page切换成功后，同样会在下发标定量列表给边缘端，边缘端处理完成后，会返回标定量的ro,rw属性，然后界面会根据对应的变量来处理变量是可以标定的，还是不可以标定的。

### 数据库设计

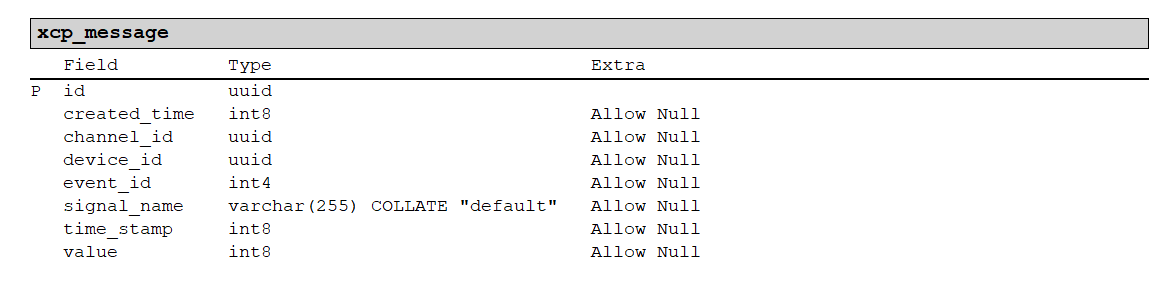
#### 数据模型

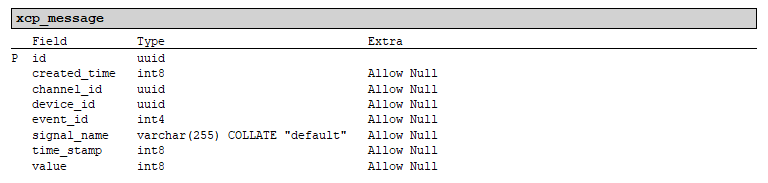












### 关键类设计

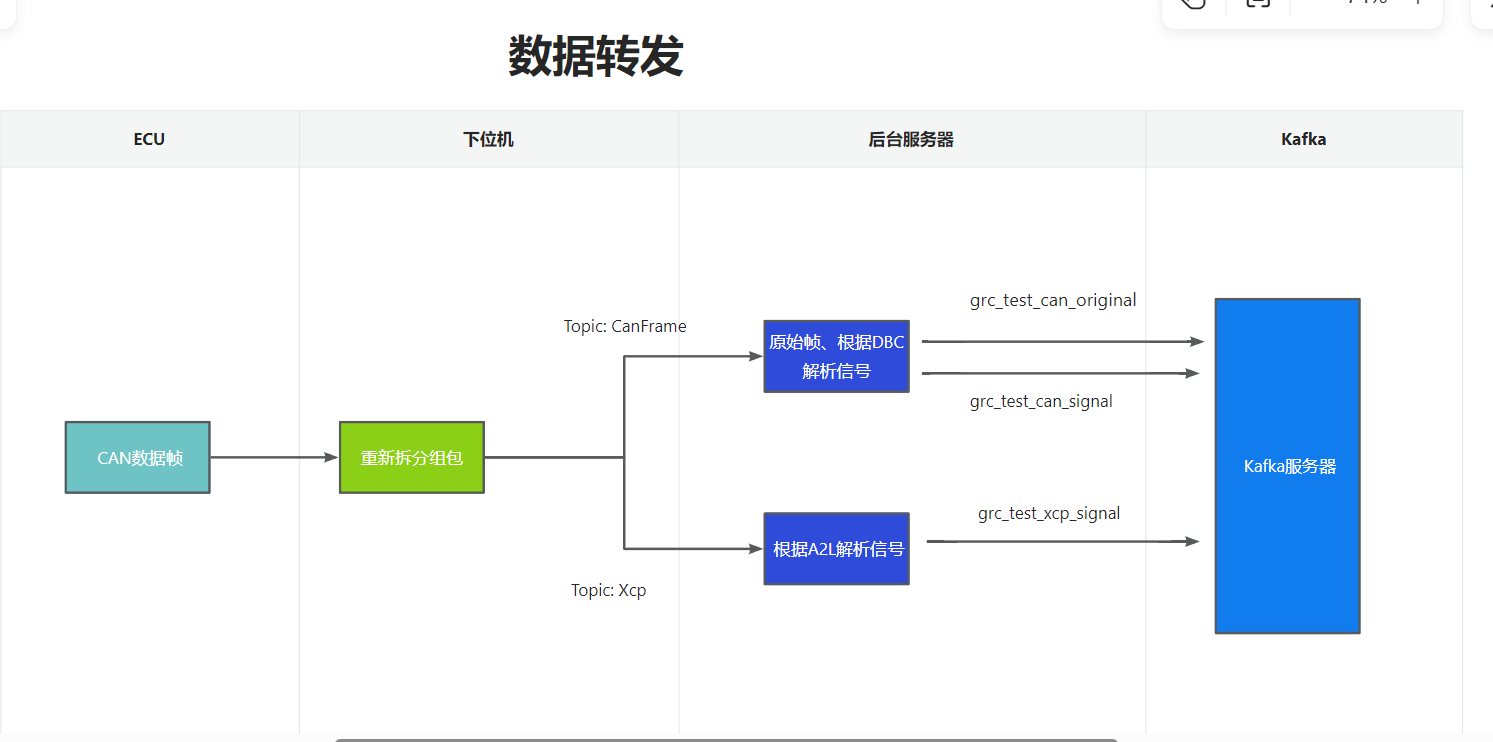
A2lCharacteristic.java, A2lCharacteristicEntity.java, A2lJson.java, A2lJsonEntity.java, A2lMeasurement.java, A2lMeasurementEntity.java,XcpMeasurementHistory.java, XcpMeasurementHistoryEntity.java,XcpMessage.java, XcpMessageEntity.java

### UI/UE设计

#### 图形用户界面, 应用程序 描述已自动生成

## 数据转发模块及功能点划分

### 数据转发模块总体设计



### 模块及功能点划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块代码**  **（编号）** | **模块名称** | **功能点编号**  **（系统需求ID）** | **功能点名称**  **（系统需求名称）** | **功能点描述** |
| MOD-TBOX-006 | 数据转发 |  | 对接Kafka | MOD-TBOX-002指定必要的Kafka地址，端口，Topic以及数据内容，自动进行传送 |

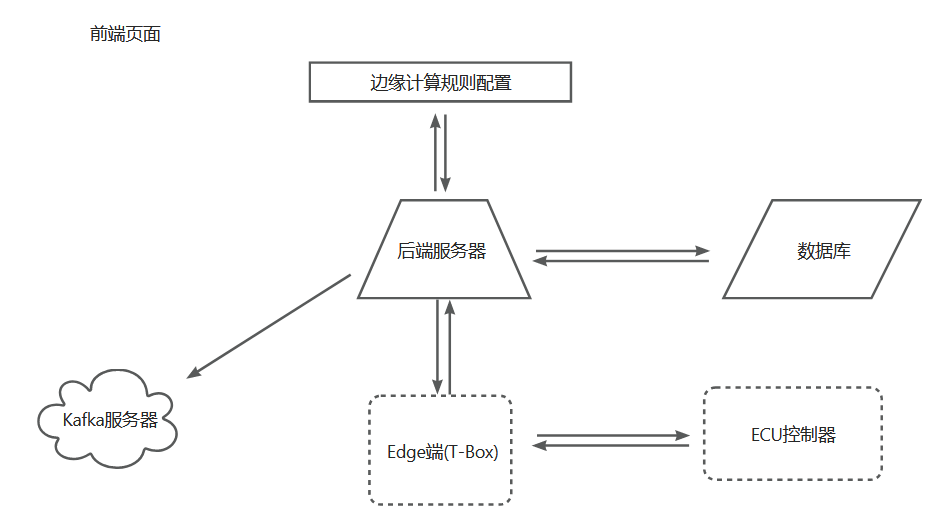
### 实现描述

根据总体设计图的数据流向， 数据从ECU中以CAN原始帧的形式发送到边缘端TBOX, TBOX接收后会对CAN原始帧的类型是CAN还是XCP进行拆分，并重新组包，然后通过MQTT转发到云端后台。云端收到数据后，首先会把CAN数据包转到Kafka; 并且将CAN数据包根据DBC的信号属性，解析成信号传到Kafka; 同时将XCP原始帧根据A2L信号量解析后转发到Kafka。

## 边缘计算应用功能设计

### 边缘计算应用总体结构

（Edge端(T-Box)和ECU控制器是辅助说明内容，非本模块功能部分）



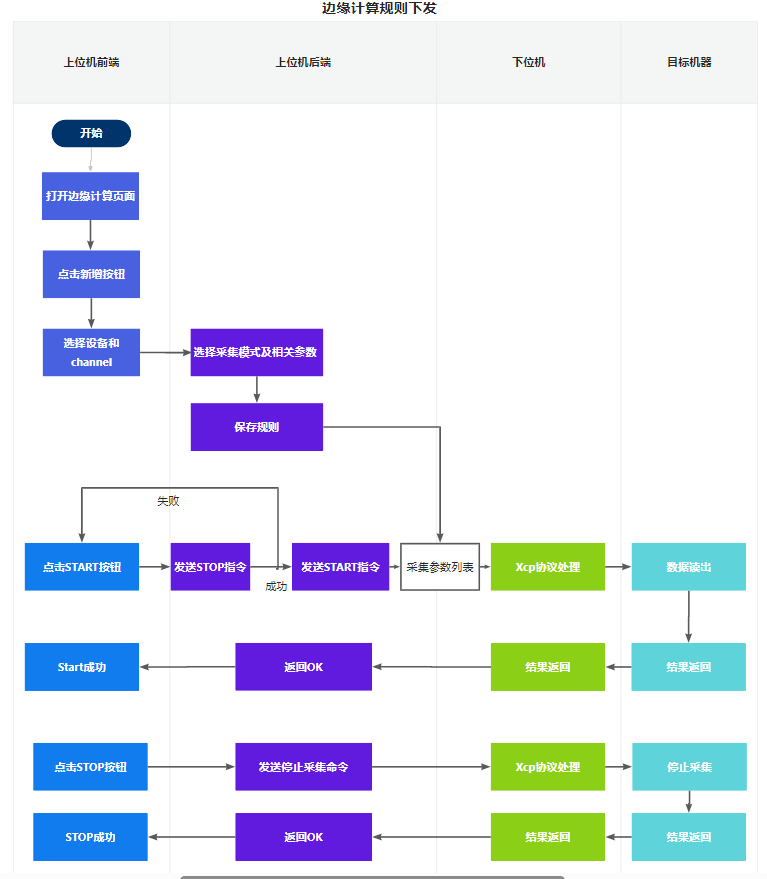
### 模块及功能点划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块代码** | **模块名称** | **功能点编号**  **（系统需求ID）** | **功能点名称**  **（系统需求名称）** | **功能点简要描述** |
| MOD-TBOX-002 | 边缘计算 |  | 规则配置 | 加载边缘计算需要的A2L/CDD文件，并保存规则 |
| MOD-TBOX-002 | 边缘计算 |  | 规则下发 | 控制开始/停止下位机按照规则采集信号 |
| MOD-TBOX-002 | 边缘计算 |  | 采集信号转发 | 用于将边缘计算采集量转发至Kafka服务器 |

#### 功能点逻辑设计

##### 边缘计算下发功能点

1. 流程图



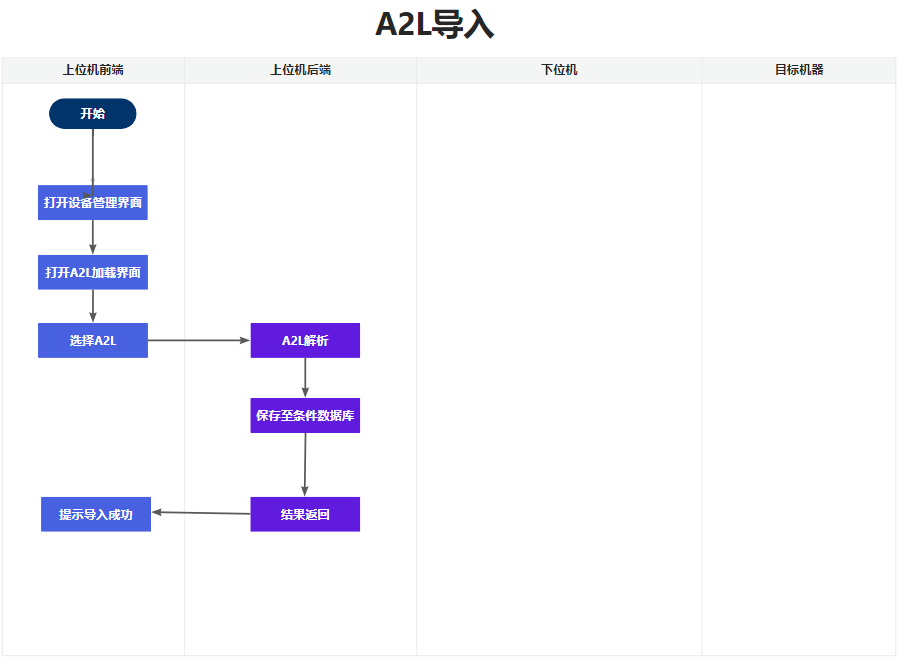
1. 涉及表和字段信息

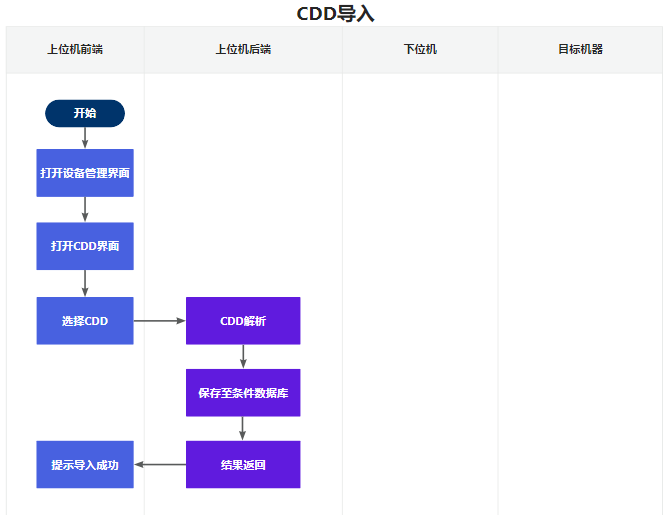
Table: edge\_computing\_rule, ecr\_measurement, ecr\_did, ecr\_dtc

1. 实现思路:
   1. 在配置完边缘计算规则后，对应设备连接后，就可以点击规则对应行上的start按钮，向下位机发送数据包，下位机确认无误后，数据就会从边缘端发送到云端，并转发至Kafka。
   2. 当用户想要终止边缘计算时，可点击Stop按钮，通知下位机停止采集发送数据至云端。

##### 边缘计算配置功能点

1. 流程图







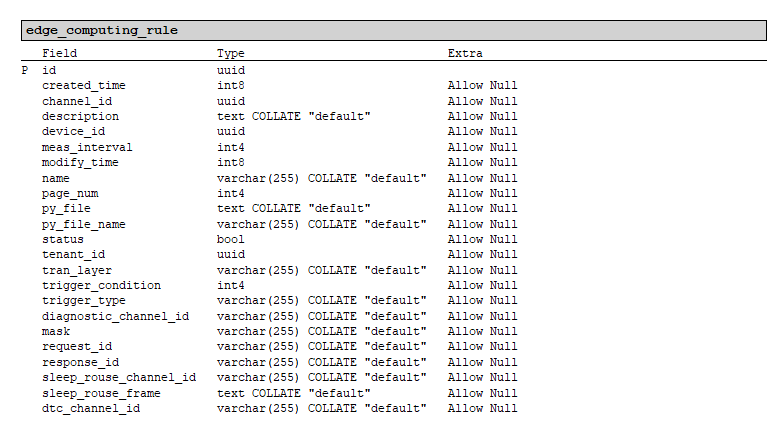
1. 涉及表和字段信息：

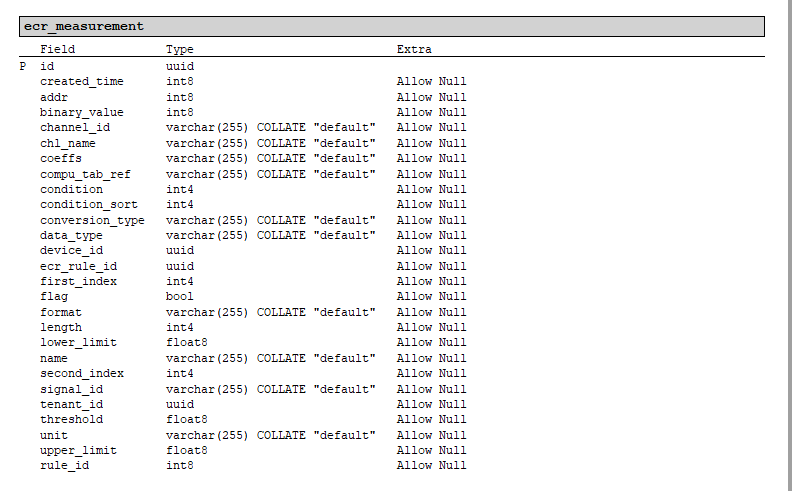
Table: edge\_computing\_rule, ecr\_measurement, ecr\_did, ecr\_dtc

1. 实现思路：
   1. 此功能点主要是 加载A2L/CDD文件到数据库 和 配置边缘计算规则。
   2. 加载A2L/CDD文件功能在设备的对应通道中，加载完文件后解析的信息就会保存到数据库中，与设备的通道信息绑定
   3. 加载A2L/CDD文件是为了准备完成边缘计算的基本信息，边缘计算采集量的数据源于A2L文件，而 边缘计算所需的DID和DTC编码源于CDD文件。
   4. 边缘计算触发类型目前已有7种类型:基本触发，手动触发，条件触发，故障码触发，代码触发，控制器休眠唤醒，诊断触发。其中，基本触发是根据用户选的设备通道下的A2L文件提供采集量供用户选择去执行采集，手动触发是基于基本触发，开始采集60S后自动停止采集。

条件触发是只采集满足用户选择的条件时的采集量。故障码触发是满足用户从CDD文件选择的DTC编码产生条件的采集量。代码触发用户可以选择采集量作为加载脚本文件的入参，当脚本文件执行成功时采集选择的采集量。控制器休眠唤醒，用户可选择任意通道作为唤醒通道去唤醒需要采集的通道。诊断触发用户可以配置诊断通道，收集间隔，DTC状态掩码，以及当前诊断通道加载的CDD文件所含有DID和DTC编码作为诊断采集时的参数。

### 数据库设计

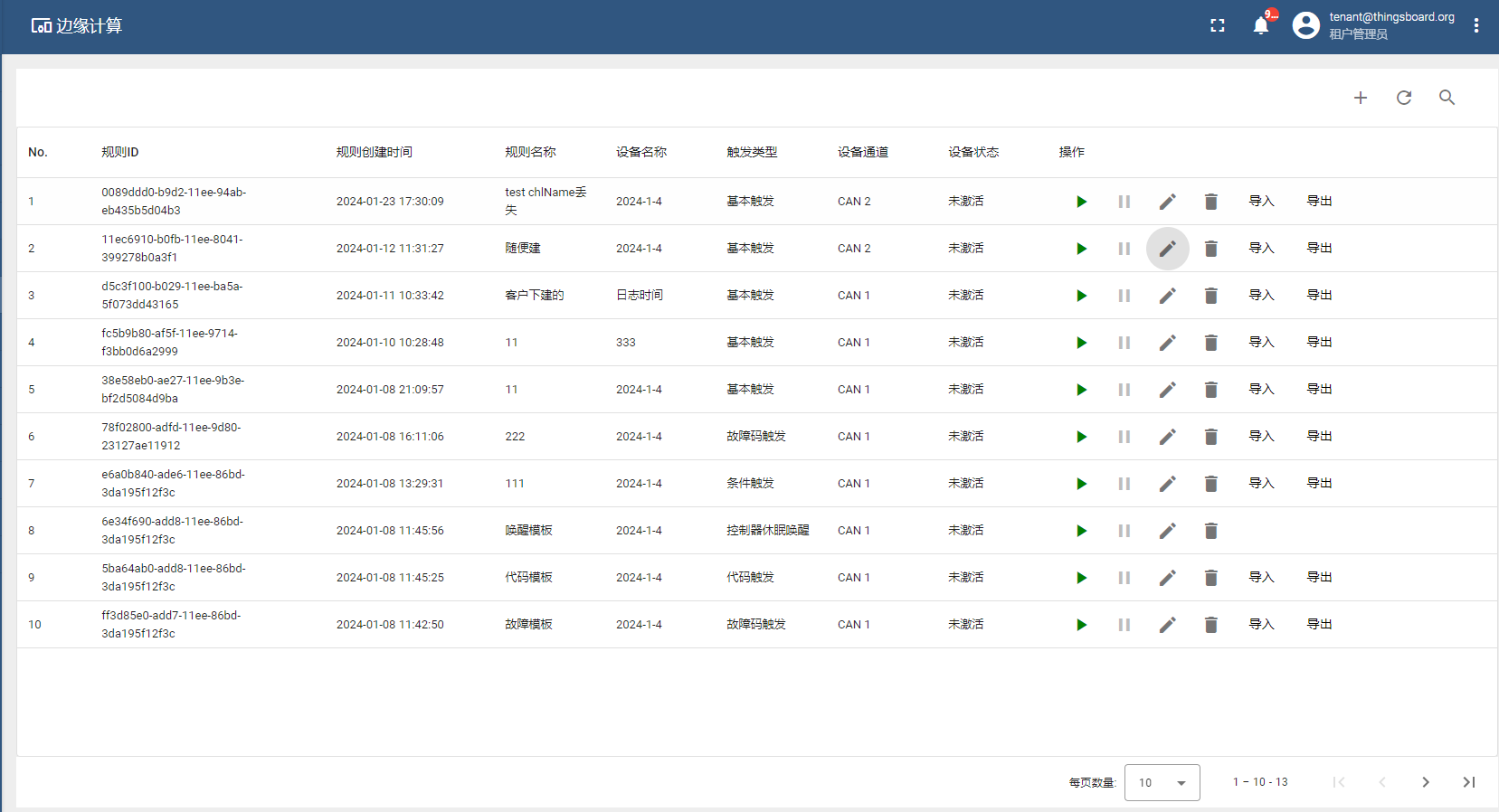


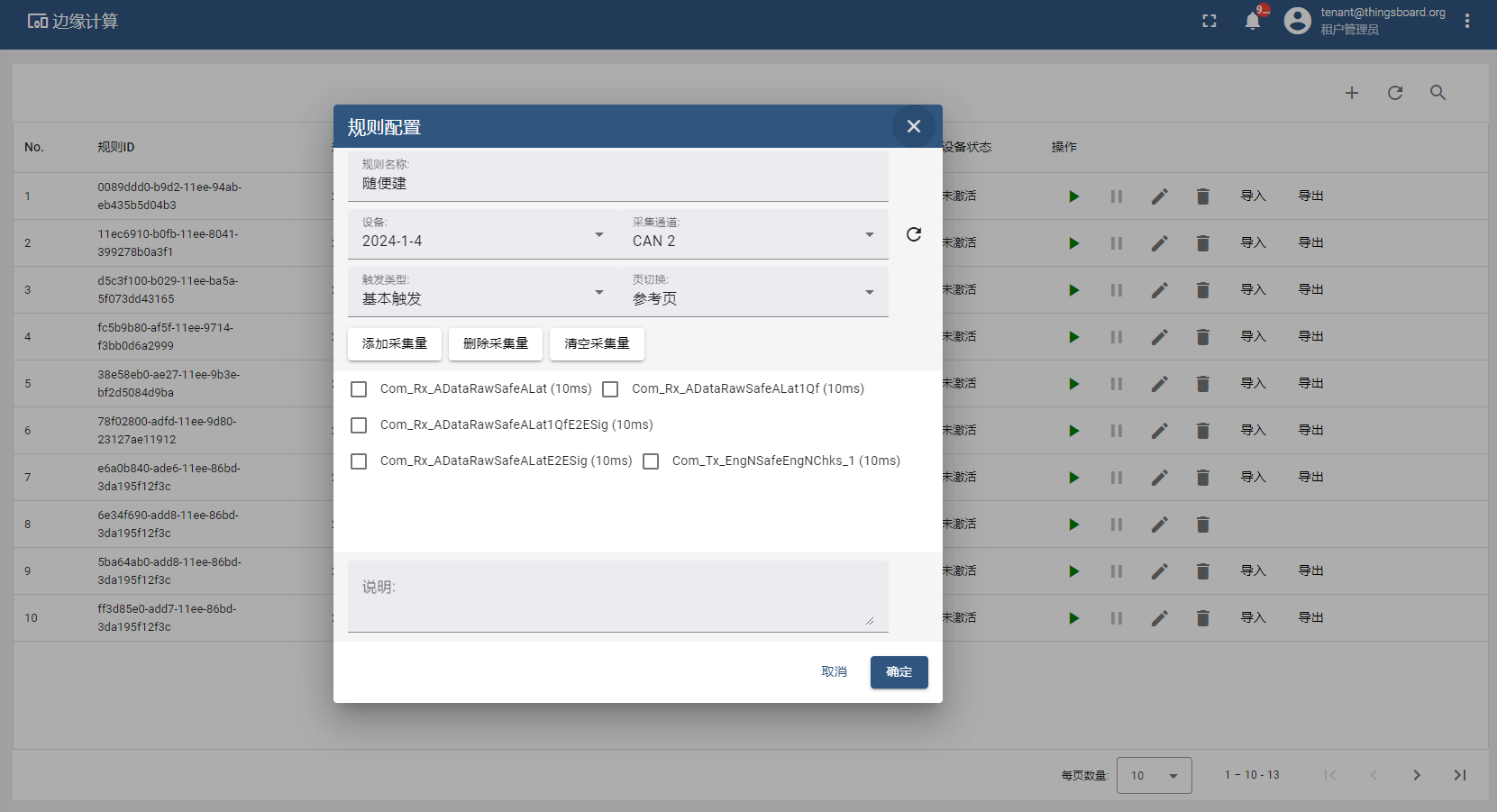


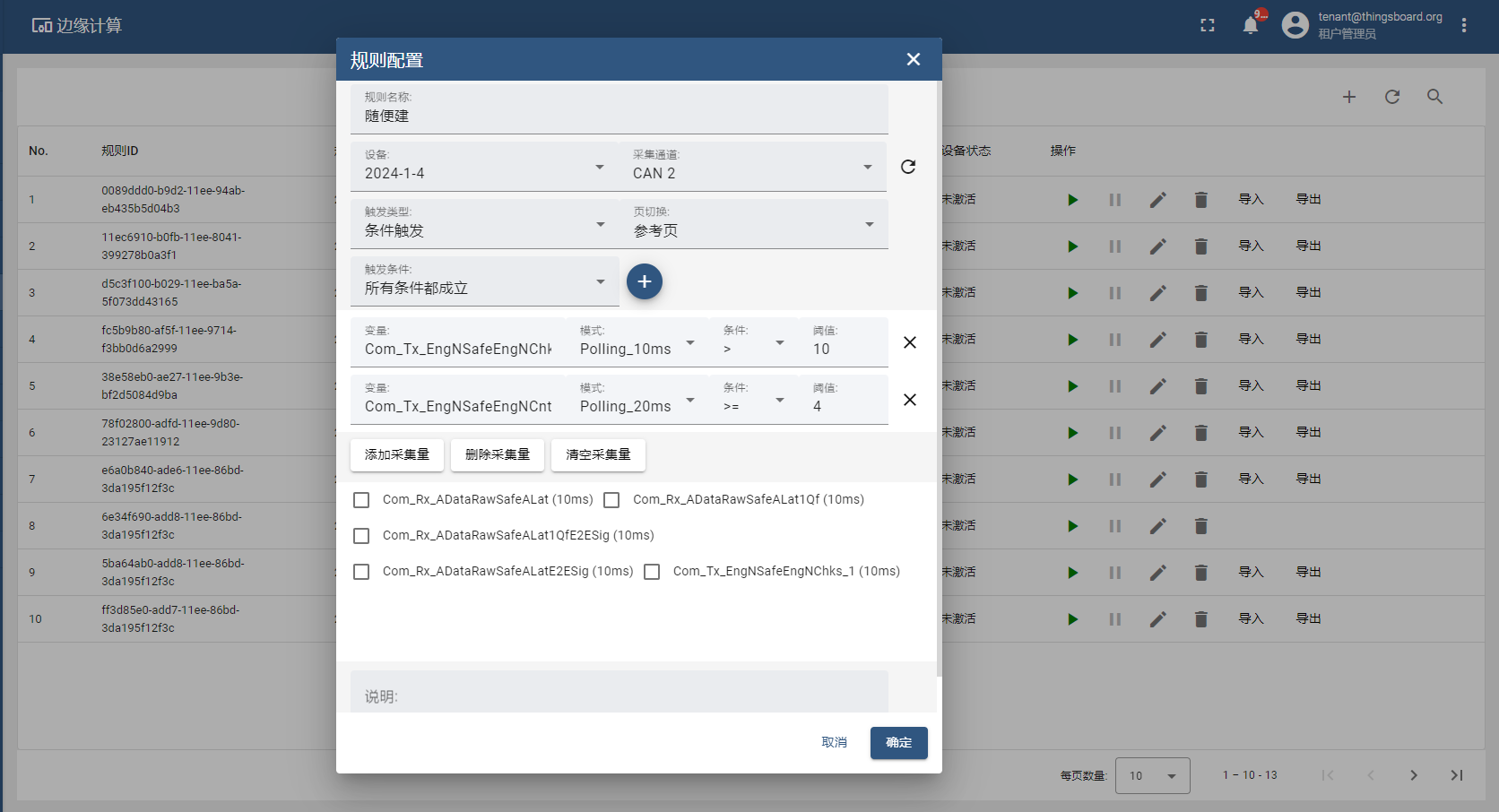
### 关键类设计

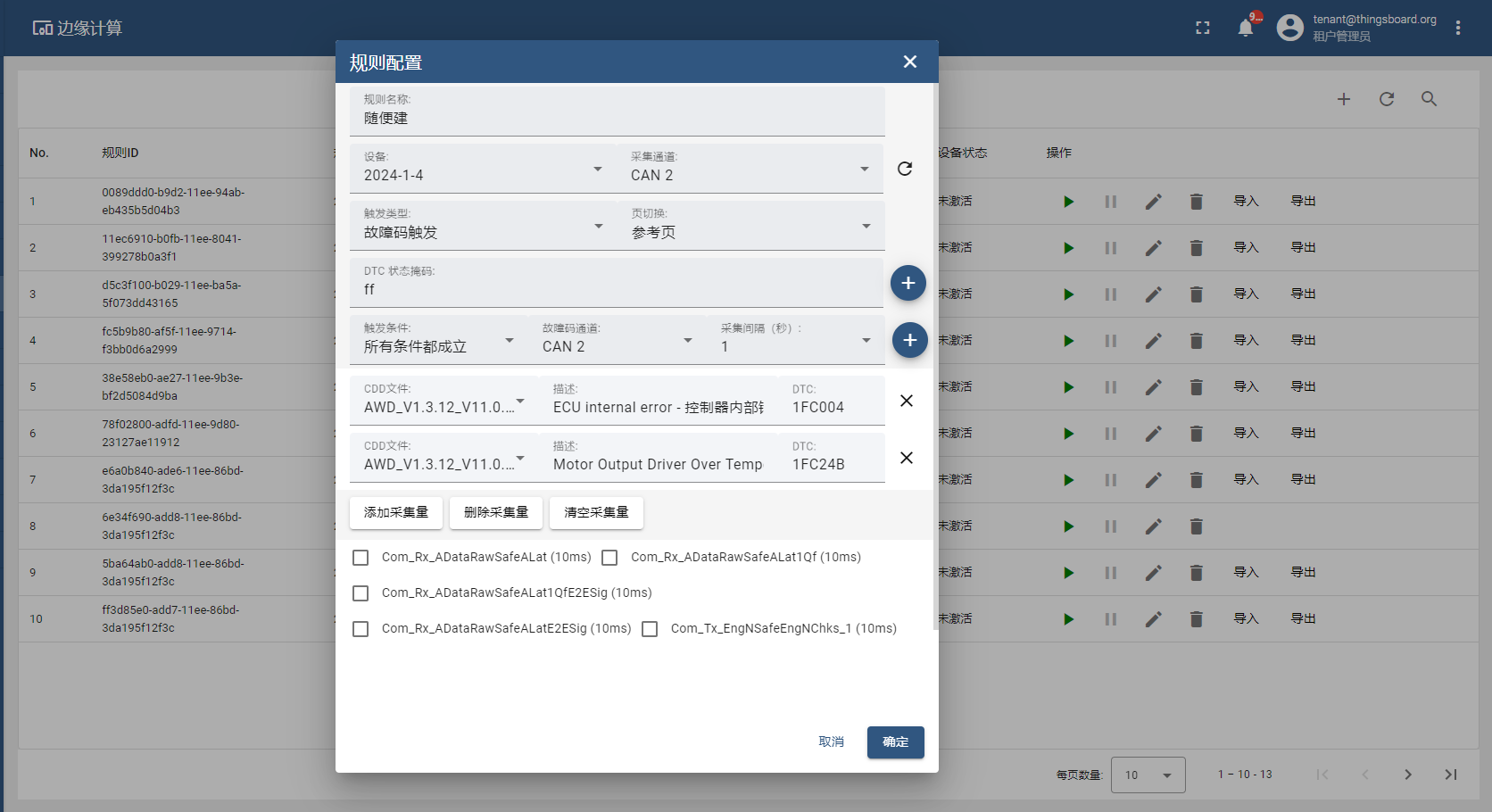
EdgeComputingRule.java, EdgeComputingRuleEntity.java,EcrMeasurement.java, EcrMeasurementEntity.java,cddInfo.java,cddInfoEntity.java

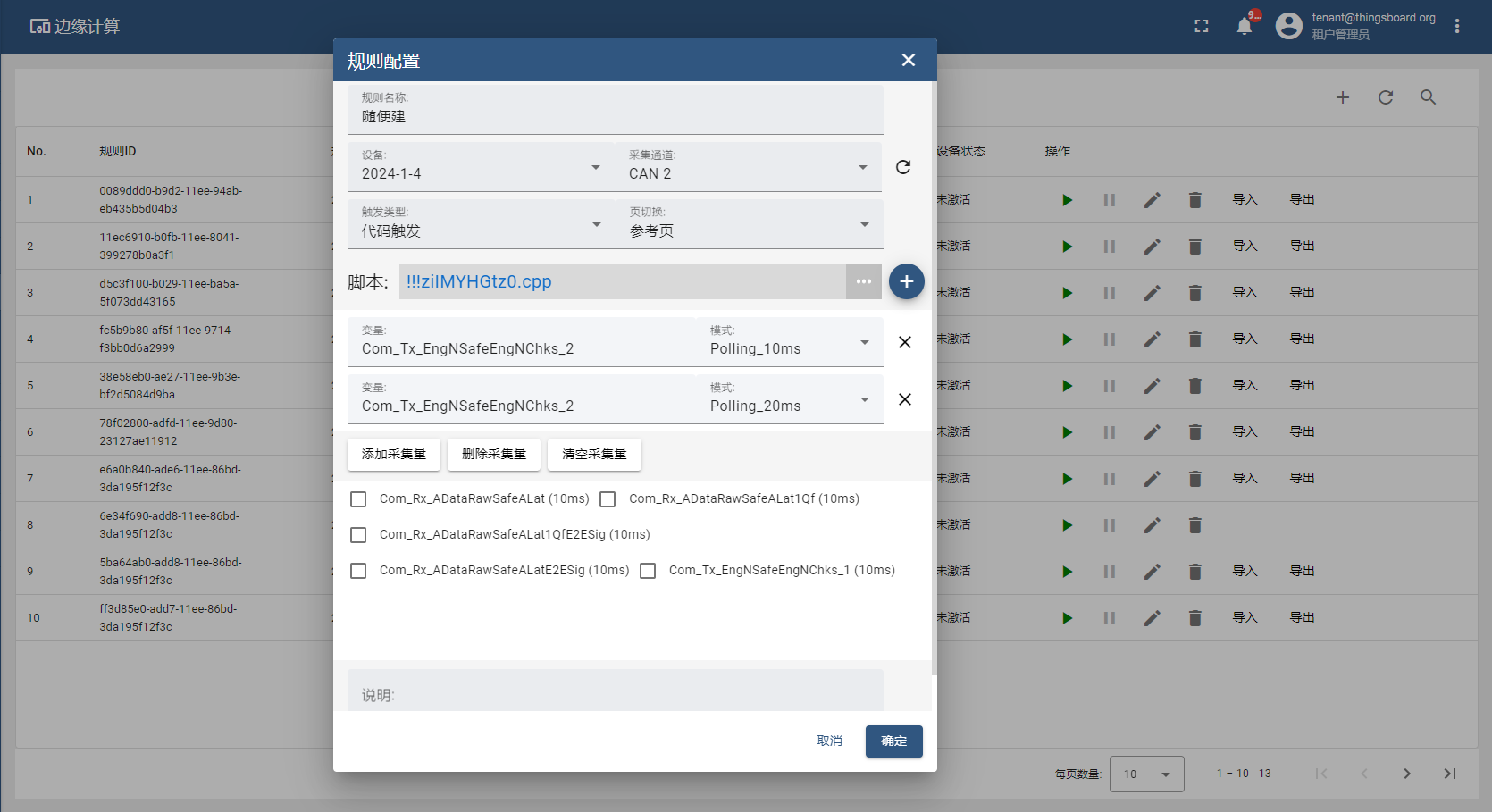
### UI/UE设计

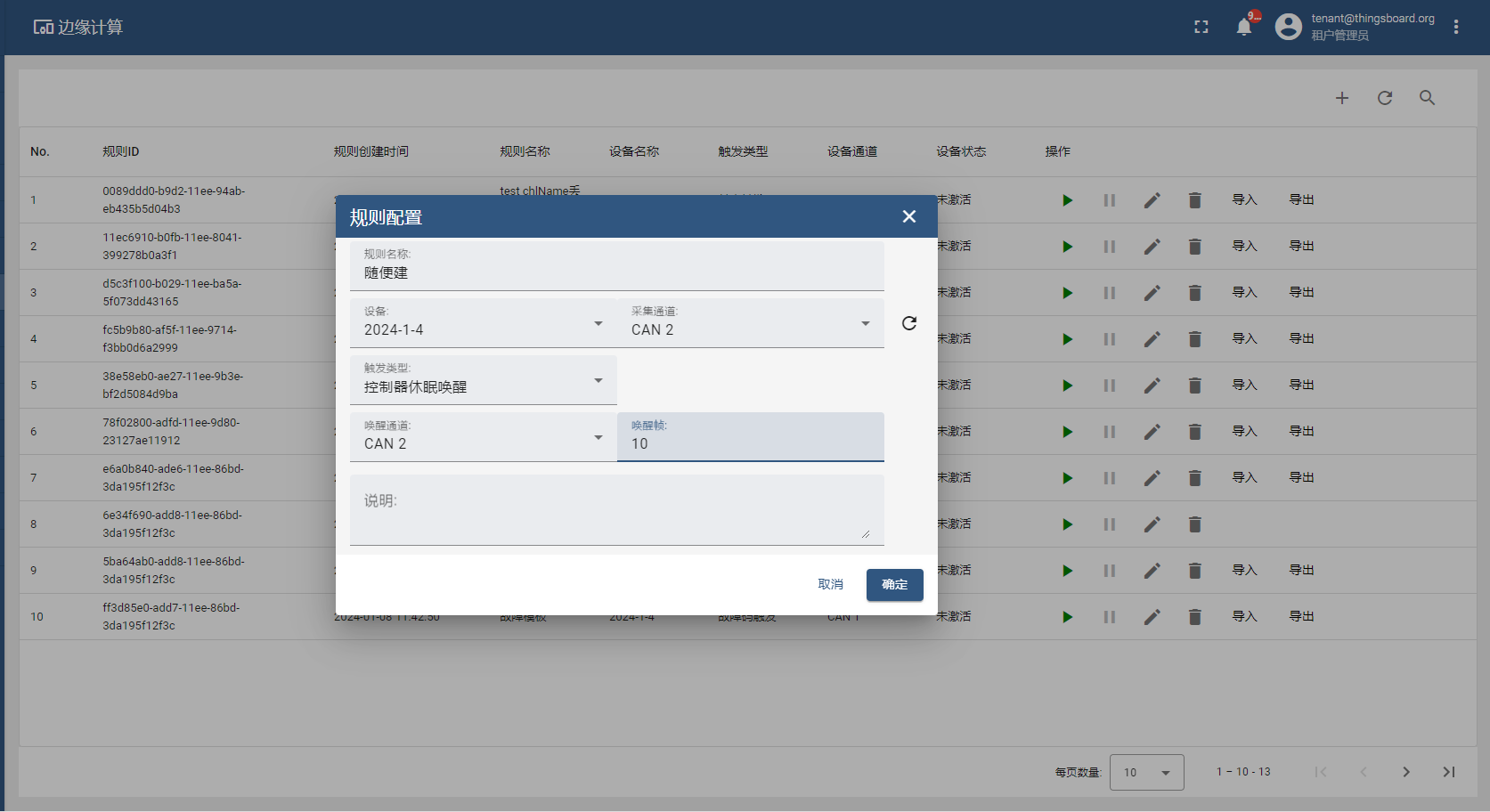










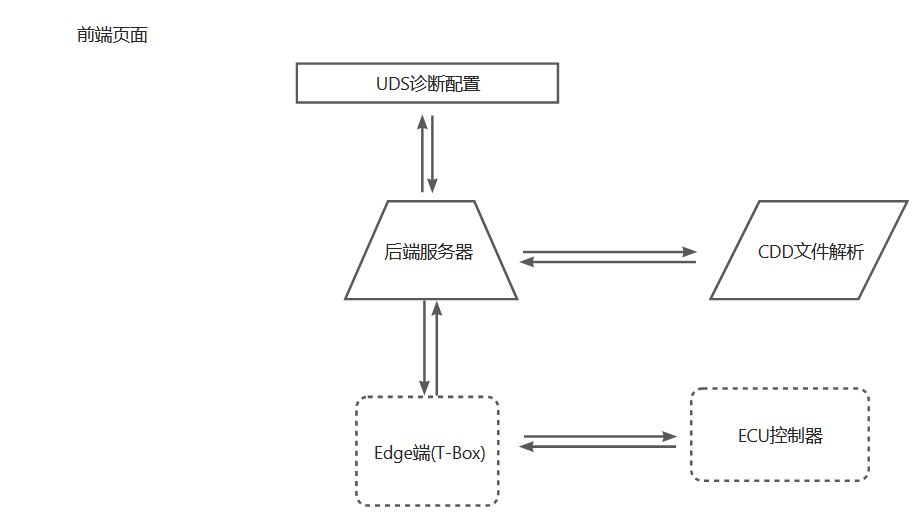




## UDS诊断应用功能设计

### UDS诊断应用总体结构

（Edge端(T-Box)和ECU控制器是辅助说明内容，非本模块功能部分）



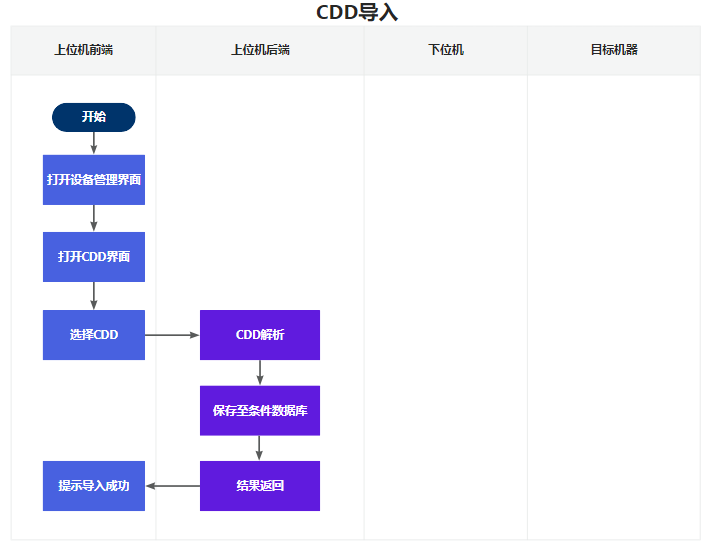
### 模块及功能点划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块代码** | **模块名称** | **功能点编号**  **（系统需求ID）** | **功能点名称**  **（系统需求名称）** | **功能点简要描述** |
| MOD-TBOX-005 | UDS诊断 |  | CDD文件加载 | 设备配置界面加载CDD文件，并解析存入数据库 |
| MOD-TBOX-005 | UDS诊断 |  | 请求服务配置下发并接收回复 | 通过请求数据控件手动配置或者通过服务单选框自动匹配编码并下发，日志栏显示请求与回复的原始数据并提供相关解释（有CDD文件时） |

#### 功能点逻辑设计

##### 加载CDD文件

1. 流程图



1. 涉及表和字段信息:

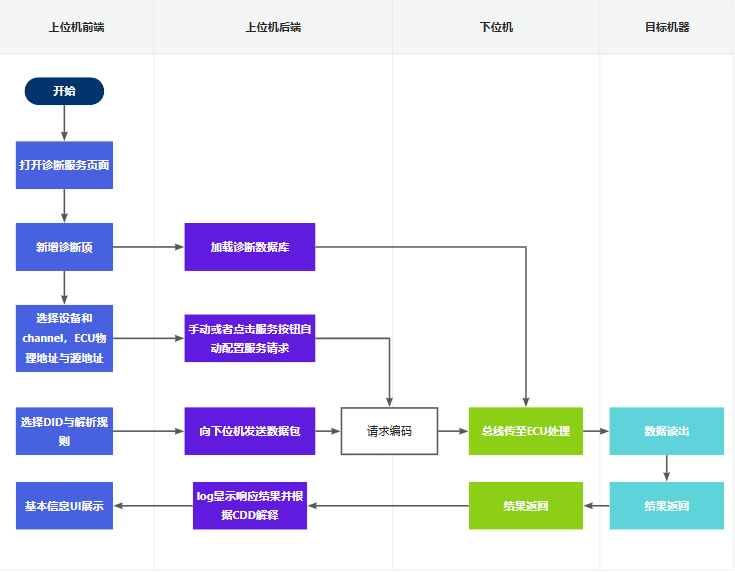
Table:cdd\_info

Column:did\_compose,did\_length

1. 实现思路:
   1. 此功能点主要是 加载CDD到数据库 和 配置边缘计算规则。
   2. 加载CDD功能在设备的对应通道中，加载完后CDD的信息就会保存到数据库中，与设备的通道信息绑定。
   3. 加载CDD是为了准备自动化UDS配置服务请求，以及在收到下位机响应时，对显示在log里的响应的原始数据进行解释。

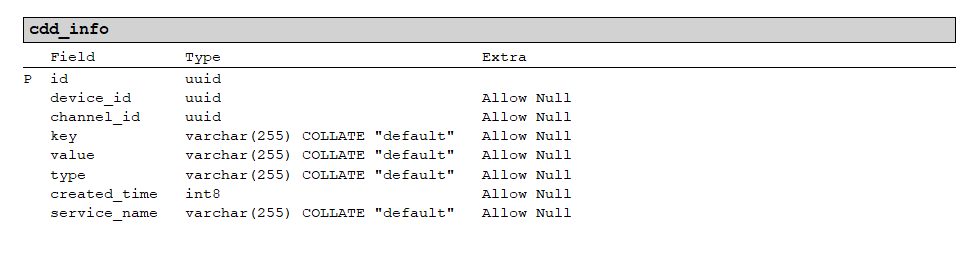
##### UDS诊断请求响应

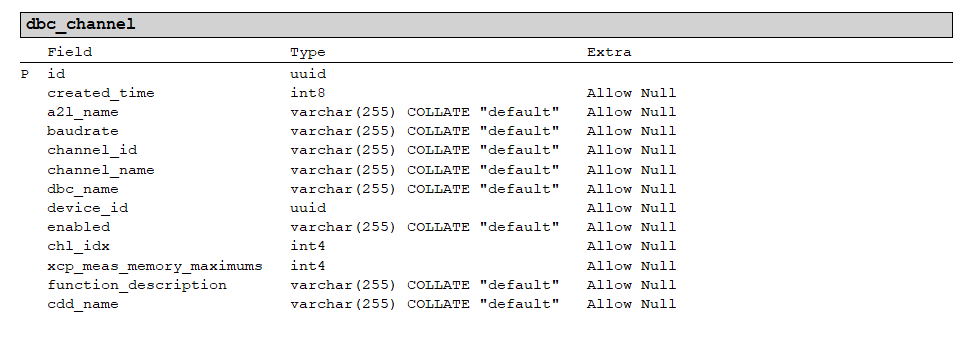
1. 流程图



1. 涉及表和字段信息:无
2. 实现思路:
3. 此功能点主要是配置用户需要的服务请求数据并下发到下位机进行UDS诊断。
4. 先添加诊断项，诊断项由诊断数据库配置，选择DID与解析规则。
5. 主要有两种方式进行配置请求数据，第一种是通过用户在请求控件中手动输入数据，第二种是通过选择界面上对应服务的单选按钮以及弹窗去自动配置请求数据，然后点击send按钮，就会将请求数据下发到下位机。
6. 之后下位机会回复一个响应数据到基本信息页面显示，若加载CDD文件，同时会对对应的响应数据进行解释。

### 数据库设计

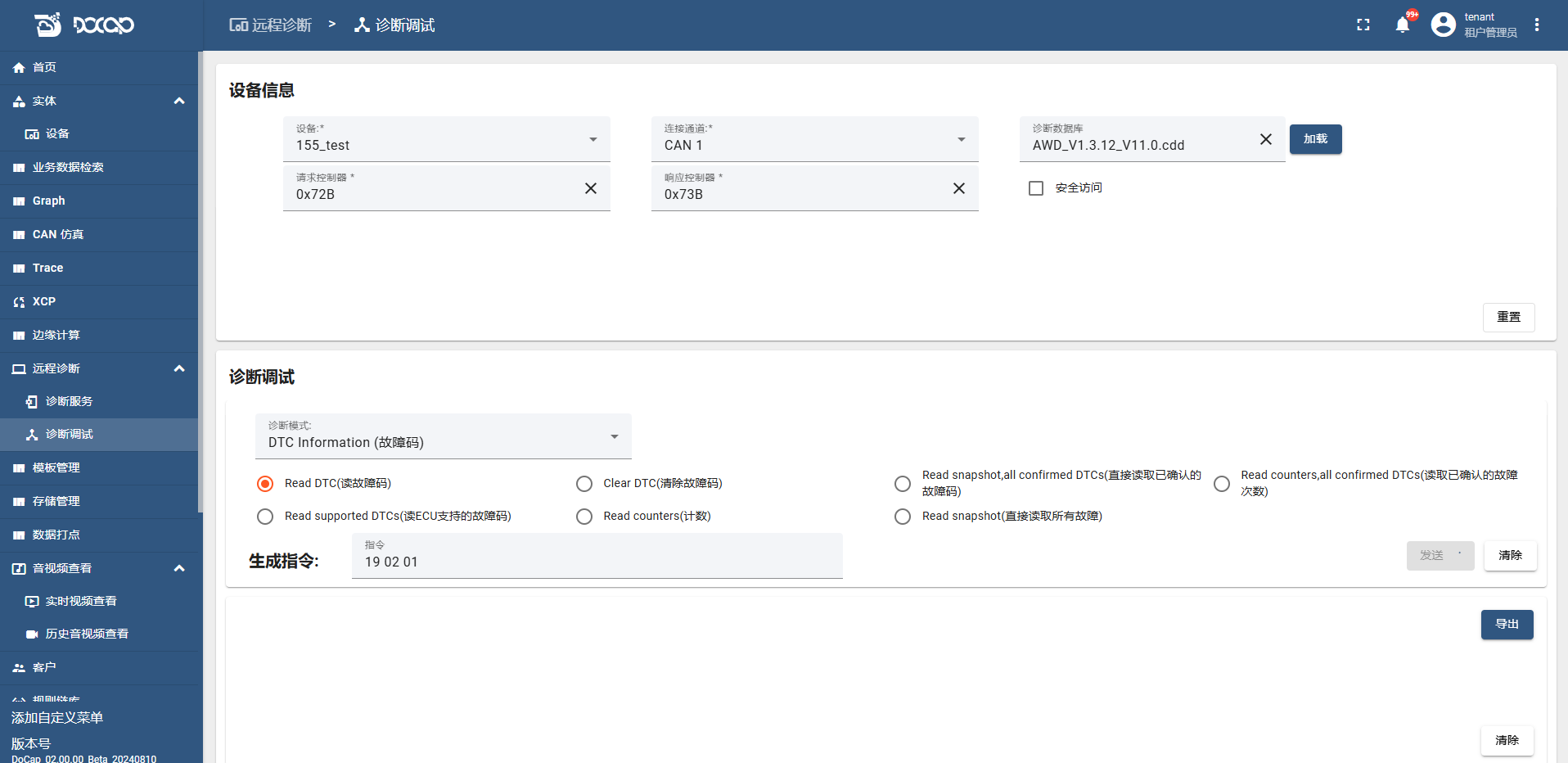


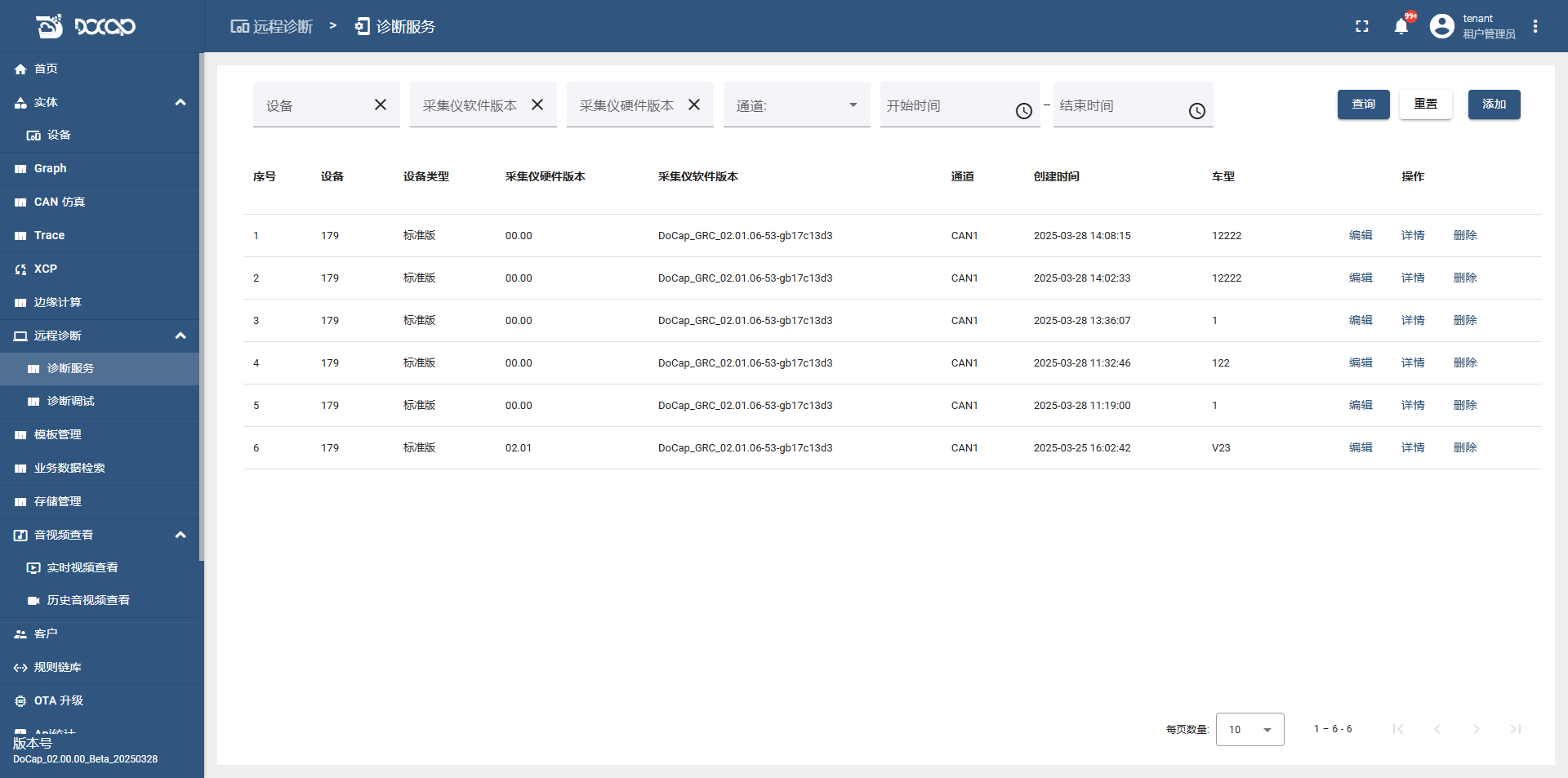


### 关键类设计

CddInfo.java, CddInfoEntity.java,DbcChannel.java, DbcChannelEntity.java

### UI/UE设计





图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 应用程序, Teams

描述已自动生成

## OTA升级应用功能设计

### OTA升级应用总体结构



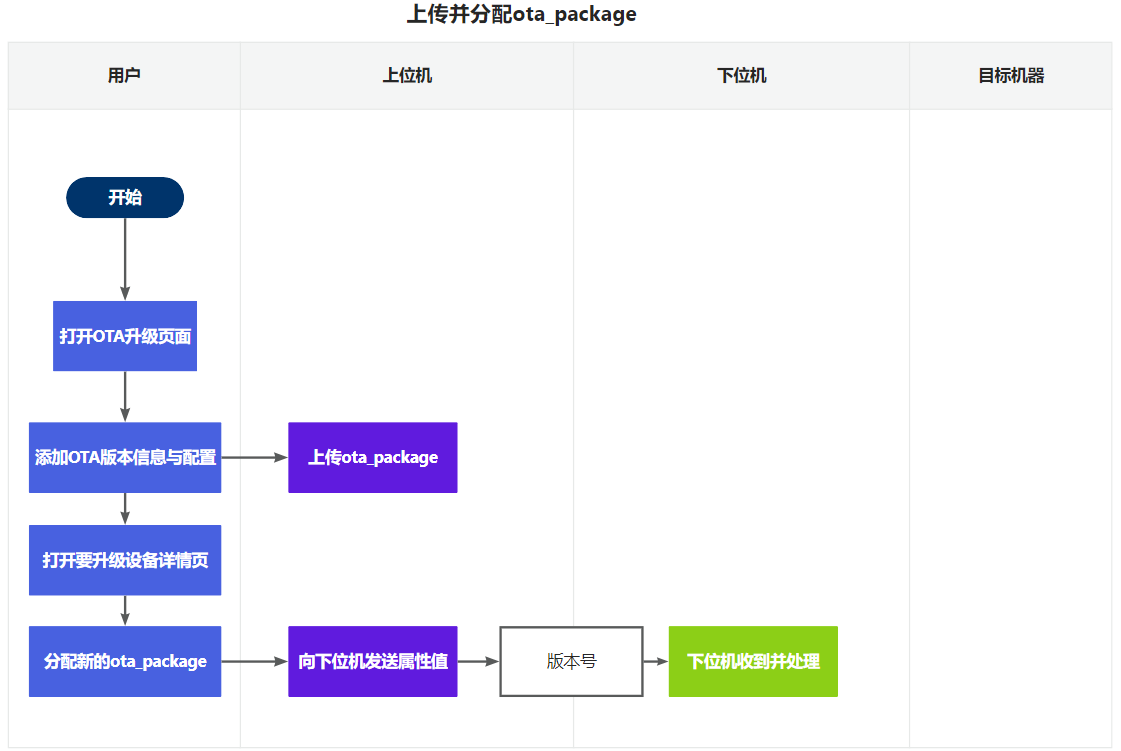
### 模块及功能点划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块代码** | **模块名称** | **功能点编号**  **（系统需求ID）** | **功能点名称**  **（系统需求名称）** | **功能点简要描述** |
| MOD-TBOX-010 | 添加并上传OTA包 |  | 新建OTA包，配置相关版本信息并上传OTA包 | OTA升级界面添加包，输入包版本号，版本名称，选择要上传的包文件 |
| MOD-TBOX-010 | 分配OTA包到要升级OTA的设备 |  | 对某个设备进行OTA升级 | 选择要升级设备详情页，分配确认好的OTA包，进行更新 |
| MOD-TBOX-010 | 下位机升级OTA包 |  | 云端和下位机收发OTA版本信息与升级OTA包 | 云端发送OTA最新的版本信息，下位机进行比较，发送升级请求，云端分chunk发送OTA包，下位机校验后进行更新，更新后返回成功或失败信息 |
| MOD-TBOX-010 | 下位机设备显示OTA更新状态 |  | 实时显示下位机升级OTA状态 | 实时显示所有设备对应下位机更新OTA的状态init到downloaded或failed |

#### 功能点逻辑设计

##### 上传并分配OTA包

1. 流程图：



1. 涉及表和字段信息：

Table : ota\_package  
Column: tenant\_id, device\_profile\_id,

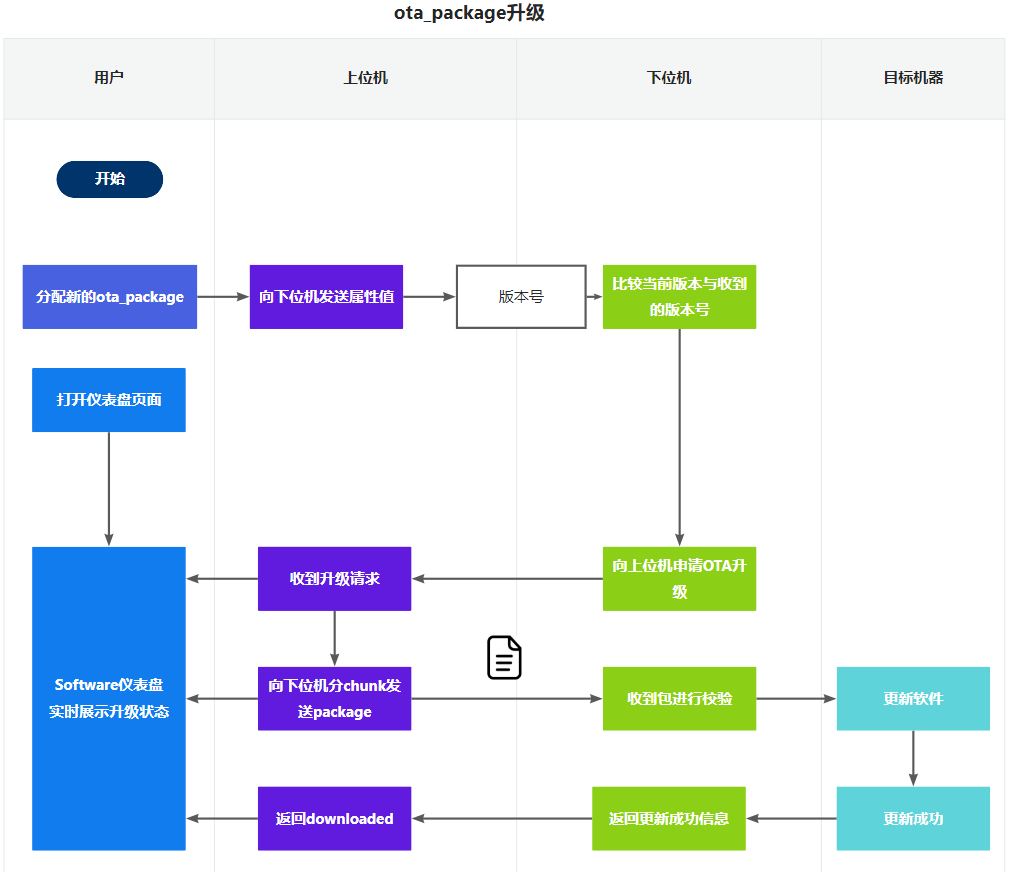
title, version, tag分别对应OTA包版本名称，版本号，标签

file\_name, checksum\_algorithm,checksum,data分别对应OTA文件名，校验和算法如SHA256, 二进制文件

1. 实现思路：
2. 此功能点主要是通过云端对边缘设备进行OTA升级。
3. 用户通过在页面上添加包输入包的版本名称，版本号，然后上传需要发送到下位机的OTA包文件完成云端OTA的配置。该包信息和文件存入ota\_package这张表，文件以二进制存入data字段。
4. 之后对需要升级的设备，选择设备详情页，编辑选择分配的软件，选择刚新添加的包或之前的包后，设备会更新属性值，在更新属性值时通过判断sw开头的属性值确定诸如sw\_version，sw\_title的信息，再通过MQTT发往对应的下位机。下位机收到后会进行处理。下位机需要提前订阅attributes/response/+和sw/response/+的MQTT topic。

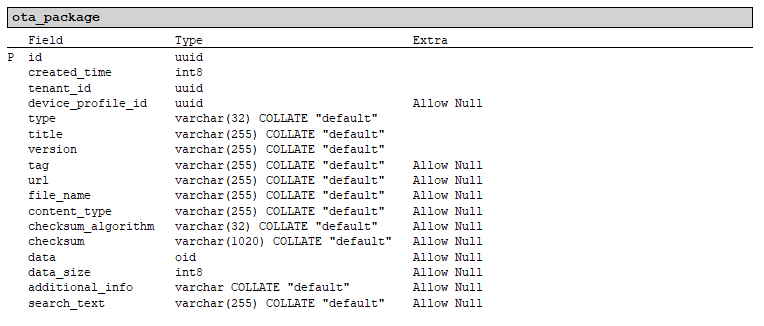
##### 升级OTA并实时展示

1. 流程图：



1. 涉及的表和字段
2. 实现思路
3. 下位机收到sw信息后和当前版本信息比较，不一致会向上位机申请下发OTA包，上位机根据存入的ota\_package分chunk推送data，下位机收到后进行文件和校验，校验通过将新的OTA包更新到本地，更新成功或失败的信息会发回上位机。这些信息都是通过telemetry的topic收发的。
4. 上位机云端会在仪表盘通过首页的仪表盘功能实时展示OTA升级状态，该仪表盘是json文件，每次创建tenant都会作为默认值插入dashboard表。该json文件包括了设备列表以及从MQTT 指定telemetry topic收到来自下位机的sw\_version, sw\_title, sw\_state = init/downloading/downloaded/failed的信息。

### 数据库设计

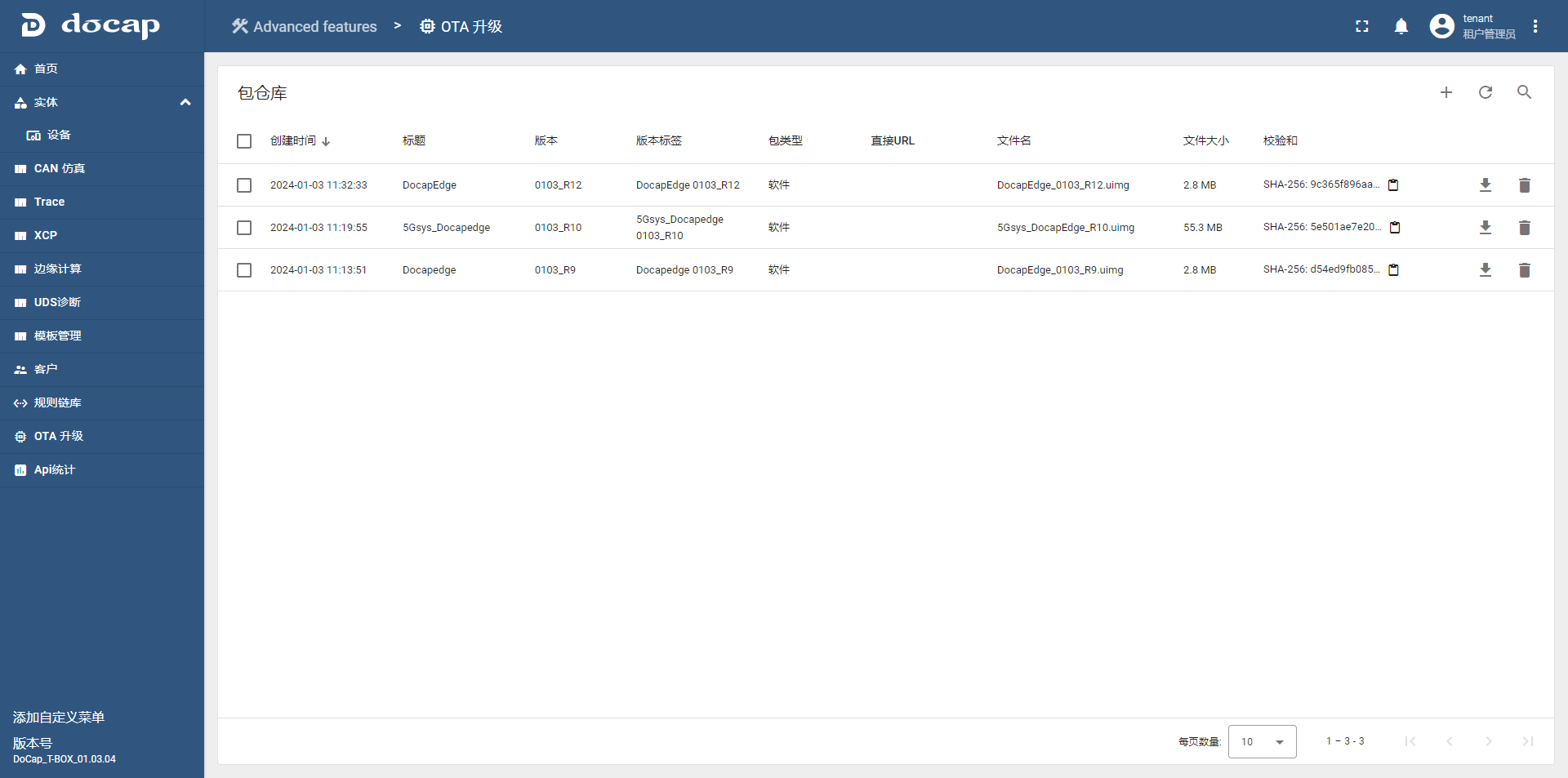


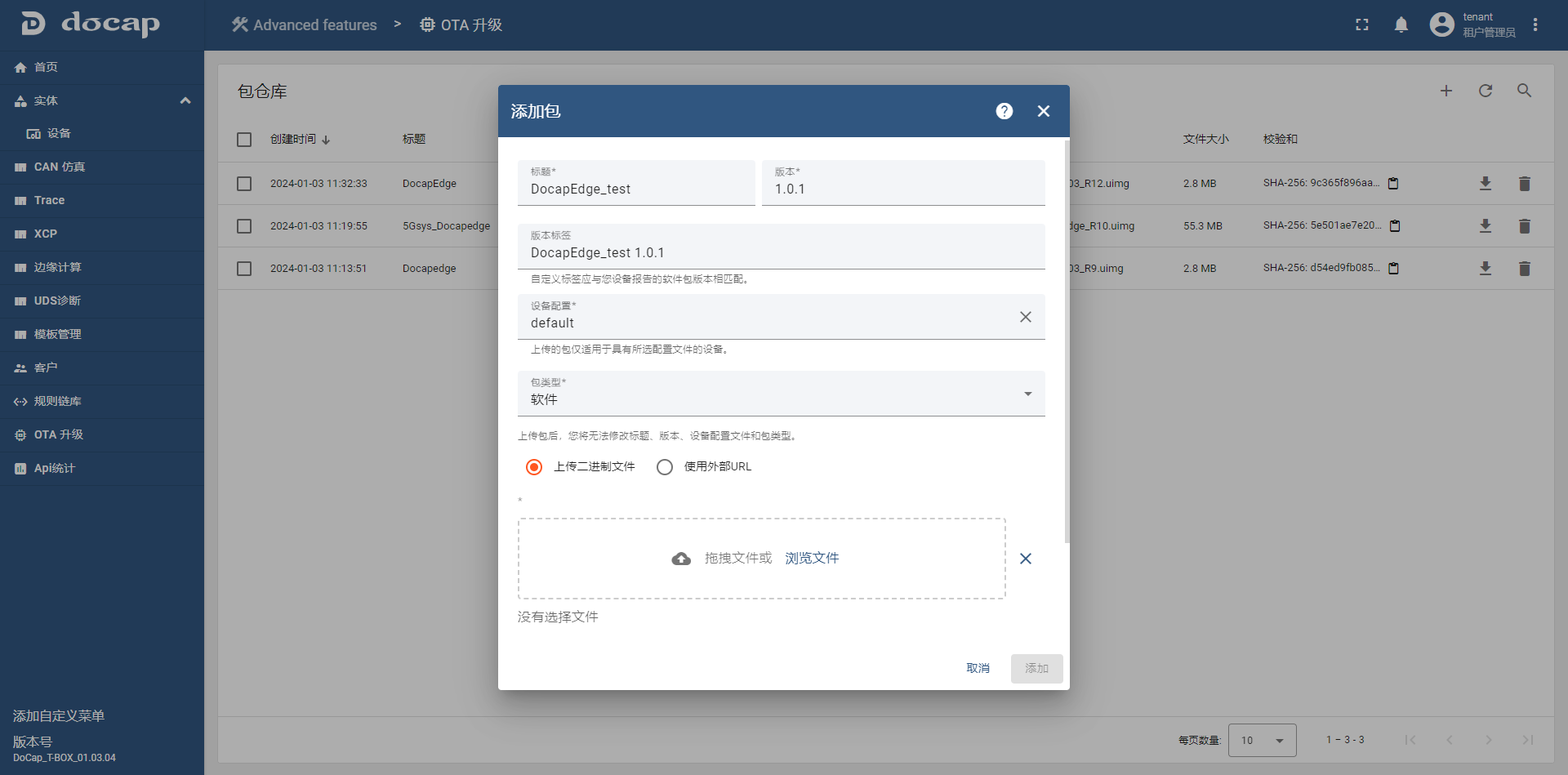
### 关键类设计

DeviceActor.java, DeviceActorProcessor.java

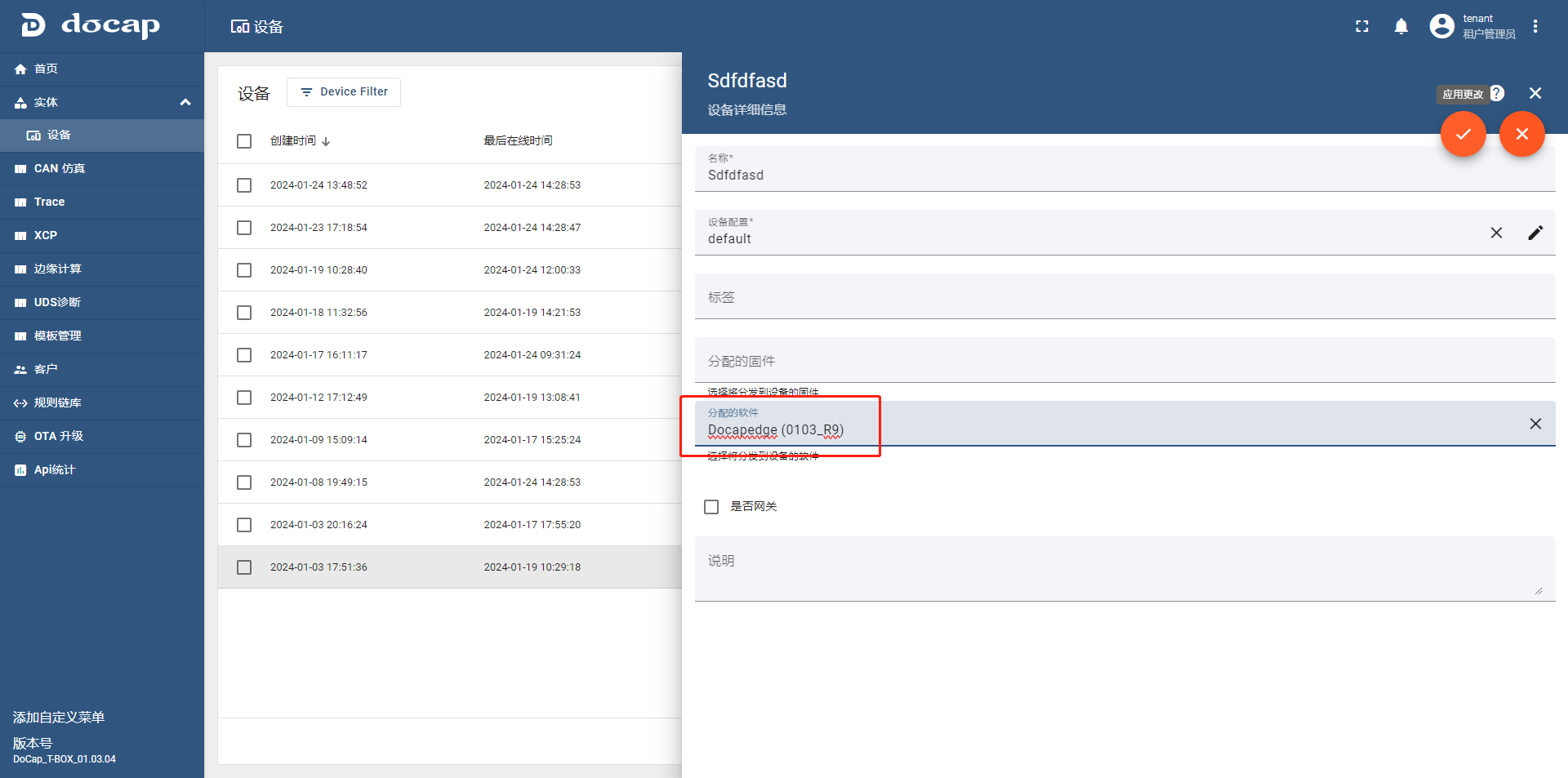
### UI/UE设计

1）上传OTA包

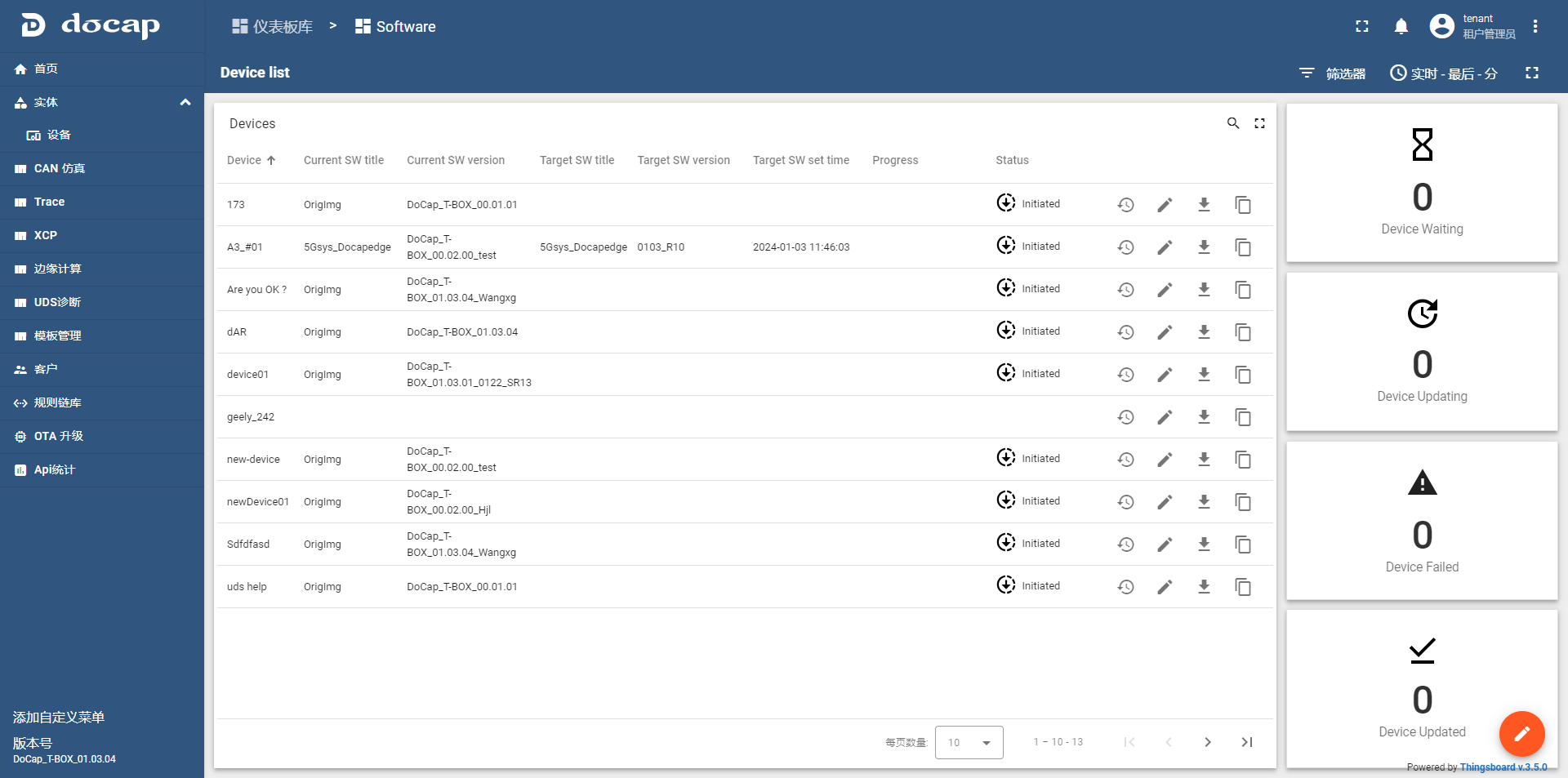




2）选择需要升级的设备，分配软件

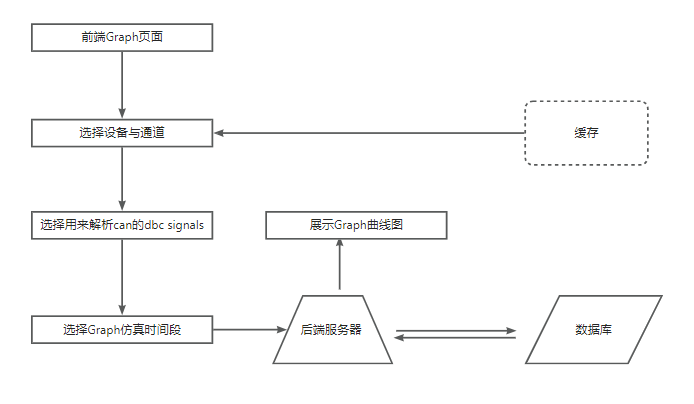


3）Software仪表盘显示升级状态



## Graph仿真应用功能设计

### Graph仿真应用总体结构



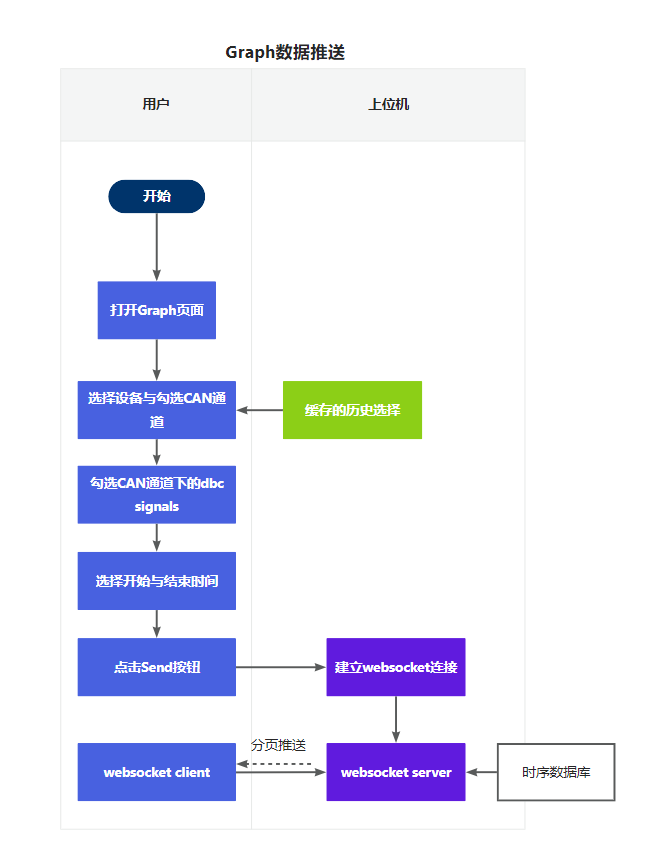
### 模块及功能点划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块代码** | **模块名称** | **功能点编号**  **（系统需求ID）** | **功能点名称**  **（系统需求名称）** | **功能点简要描述** |
| MOD-TBOX-003 | 选择设备通道与下面的signals |  | 用户可以选择设备与勾选多个CAN通道，以及通道下面的signals | 会自动记录上次的选择，树状图展示CAN通道下面的messageName与signalName |
| MOD-TBOX-003 | 选择Graph仿真的时间段 |  | 用户可以精确到毫秒级在日历上选择时间 | 通过自定义的日历组件实现毫秒级时间选择，同时限定结束时间不得超过开始时间后的24小时之后 |
| MOD-TBOX-003 | 请求后端该条件下DBC解析的值 |  | 后端按page不断推送DBC解析后的(x,y)数据 | 后端根据请求的参数查询时序数据库，对数据按页进行分割，查询请求的DBC文件对CAN数据解析获得每个信号下的值，再通过开启计划调度线程池每隔固定时间推送给前端 |
| MOD-TBOX-003 | 前端显示Graph仿真曲线图 |  | 显示每个信号解析的基于时间的值序列 | 显示横坐标为时间，纵坐标为DBC解析CAN后的值，对应多条已选择的信号，可以进行缩放显示 |

#### 功能点逻辑设计

##### Graph数据推送与展示

1. 流程图：



1. 涉及表和字段信息：

Table：grc\_can\_frame\_bytes\_cf

Column: device\_id, start\_timestamp, end\_timestamp, bytes(CAN数据包), channel\_ids(CAN数据包包含的channelId)

Table: dbc\_signal\_his

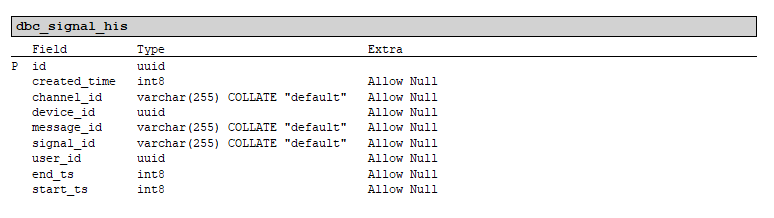
Column: device\_id, channel\_id,message\_id, signal\_id,start\_ts,end\_ts

Table: dbc\_signal, dbc\_message用来解析can数据

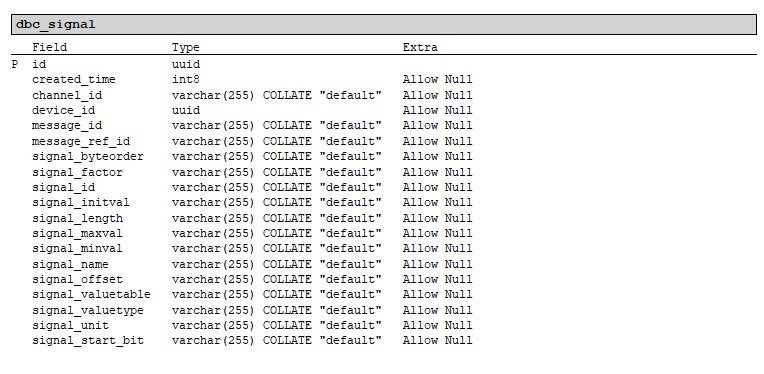
1. 实现思路：
2. 后端先提供记录历史选择的接口，接口可以按树形结构返回上次选的设备id, channelName, messageName以及下面的signalName。
3. 编写Websocket server端，规定topic以及subscription的cmd指令，指令包括指令id，和client传入的signalNameList, start\_time, end\_time, deviceId等参量。同时每次收到参量会将条件存入历史选择中。
4. Server端从时序数据库获取client请求的时间段全部数据，根据deviceId, channelName和signalNames获取相应的DBC量，对时序CAN进行解析得到时序值。通过调度线程池，周期分页地发送时序数据给client。
5. 编写Websocket client端，根据topic和cmd格式定义好下发的数据类型。
6. 前端使用Echarts曲线绘制图形，可以放大缩小预览全局。

### 数据库设计

#### 历史信号选择记录表

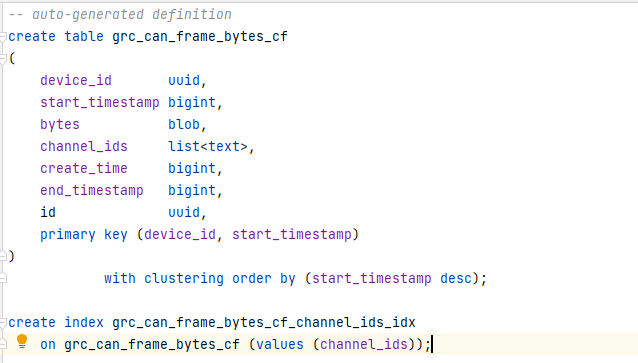


#### 从DBC中解析的信号



#### message\_id和message\_name关系映射表

#### CAN时序数据表

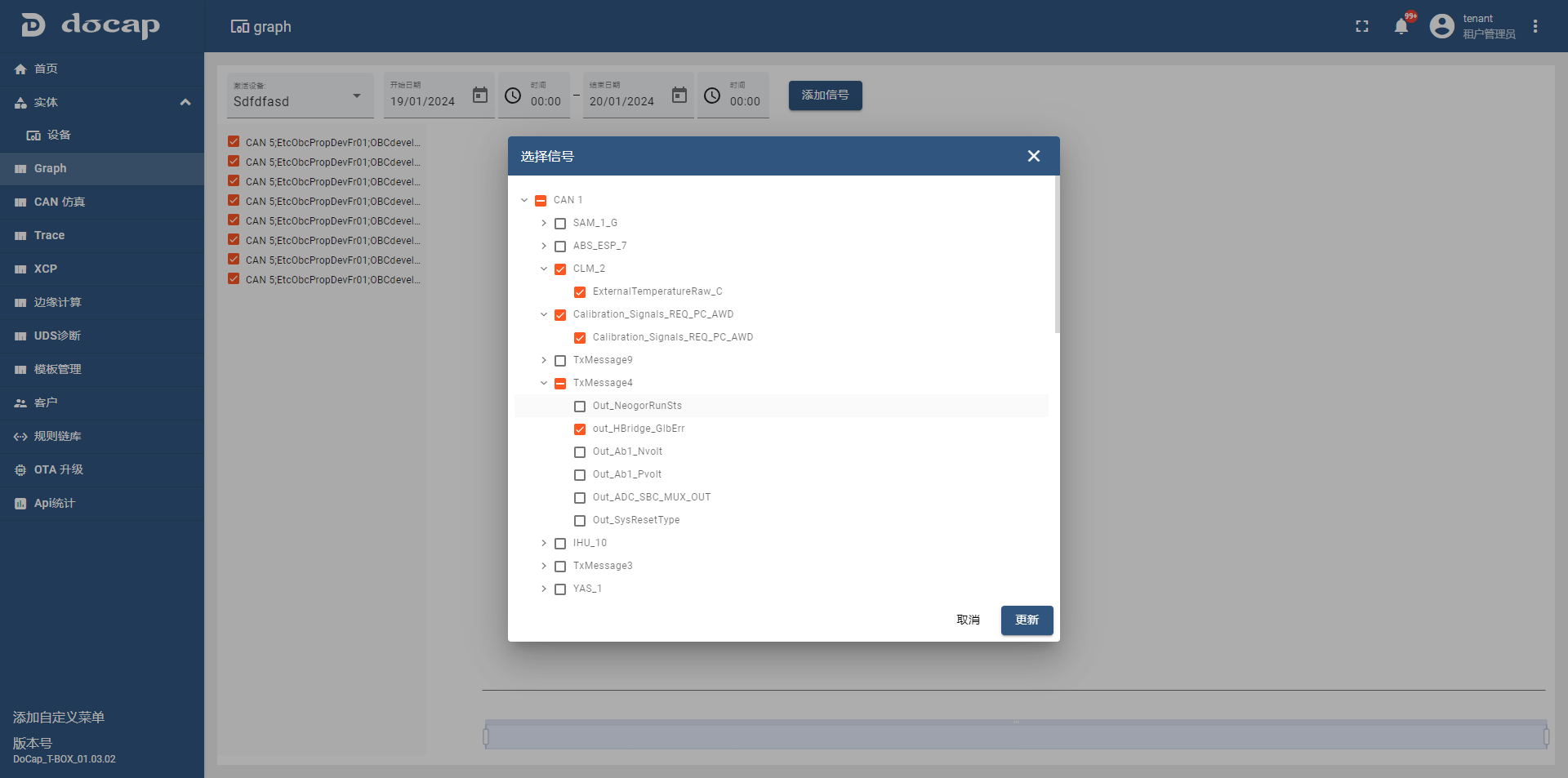


### 关键类设计

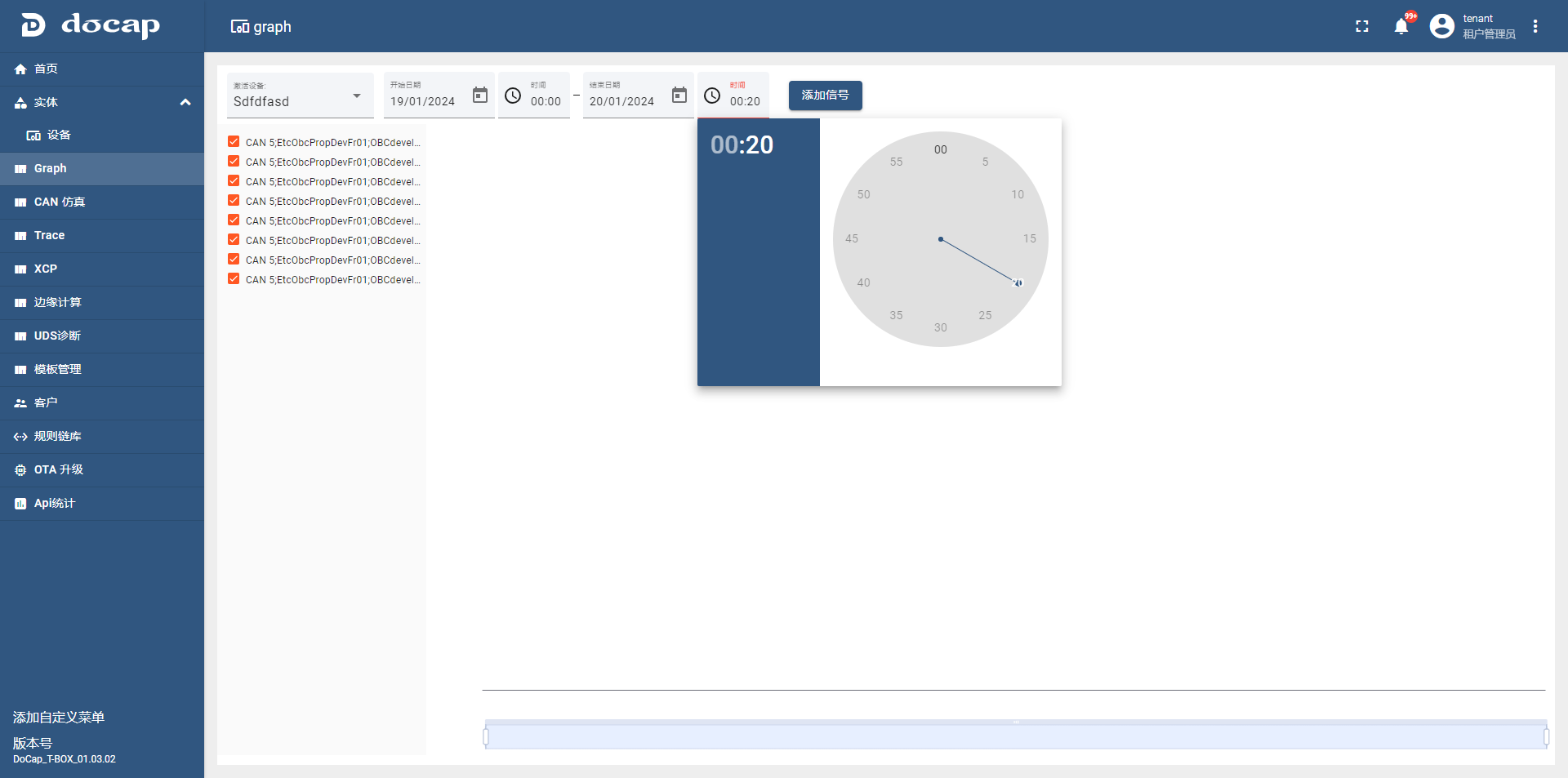
SimulationCommandsHandler.java, BaseCassandraTimeseriesService.java

### UI/UE设计

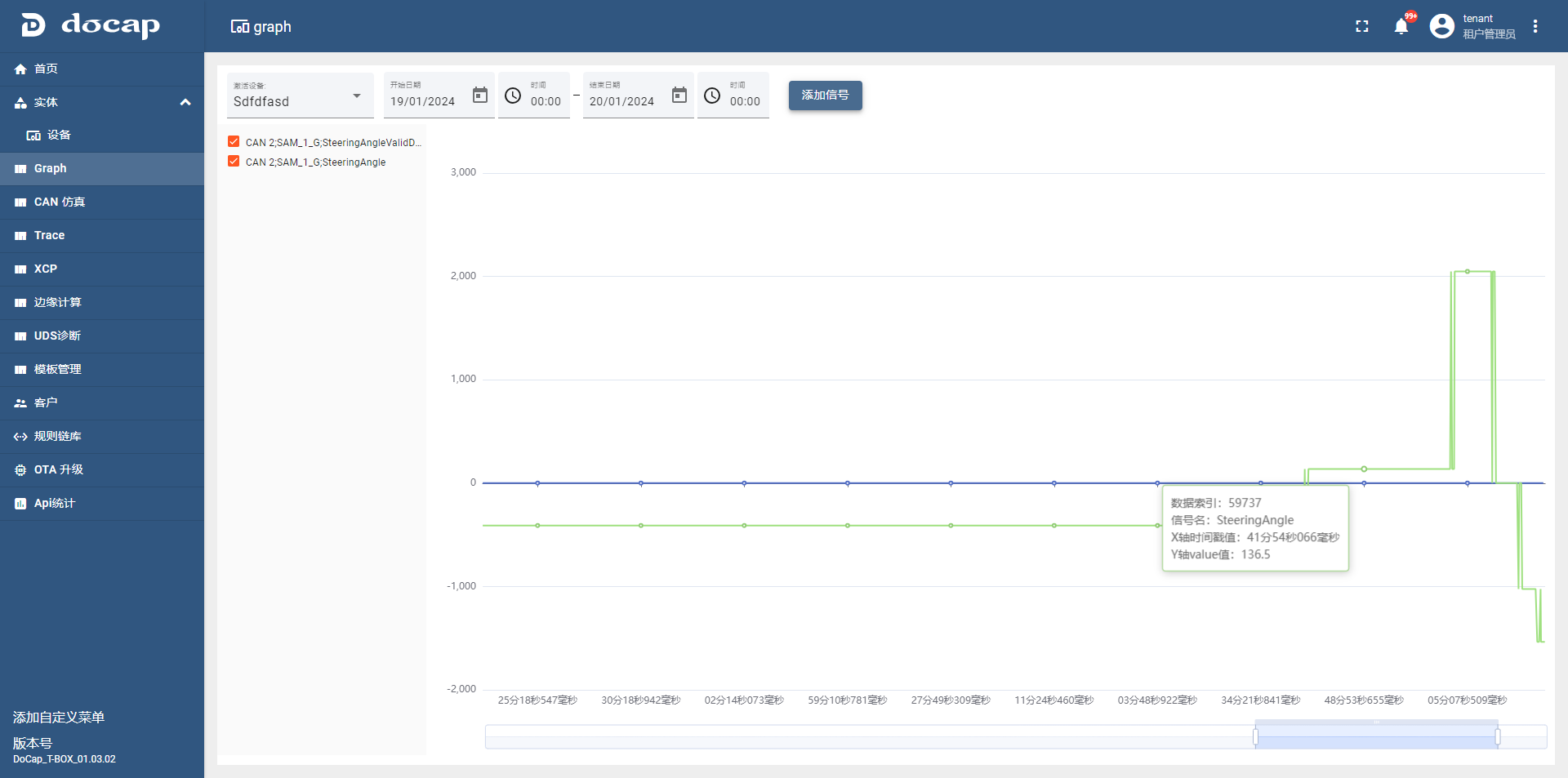
#### 选择CAN通道及其加载的信号



#### 选择时间区间，精确到毫秒（区间大小<=24h）



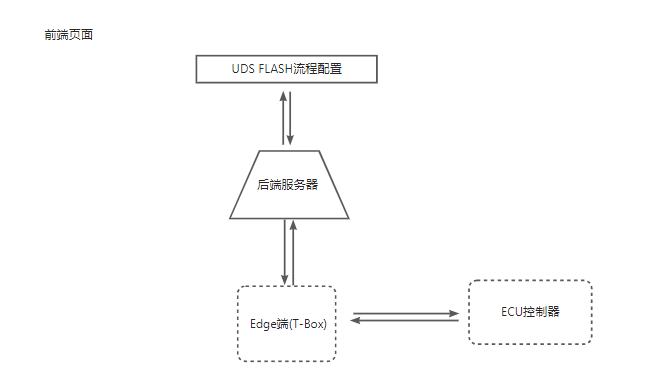
#### 展示Graph解析出来的信号数据



## UDS FLASH应用功能设计

### UDS FLASH应用总体结构

（Edge端(T-Box)和ECU控制器是辅助说明内容，非本模块功能部分）



### 模块及功能点划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块代码** | **模块名称** | **功能点编号**  **（系统需求ID）** | **功能点名称**  **（系统需求名称）** | **功能点简要描述** |
| MOD-TBOX-006 | UDS FLASH |  | 烧写流程配置 | 流程配置界面加载SeedKey文件，添加不同类型的烧写步骤 |
| MOD-TBOX-006 | UDS FLASH |  | 烧写流程下发实时显示日志进度 | 把流程下发给下位机，并实时接收下位机响应的日志进度信息。 |

#### 功能点逻辑设计

##### 烧写流程配置功能点

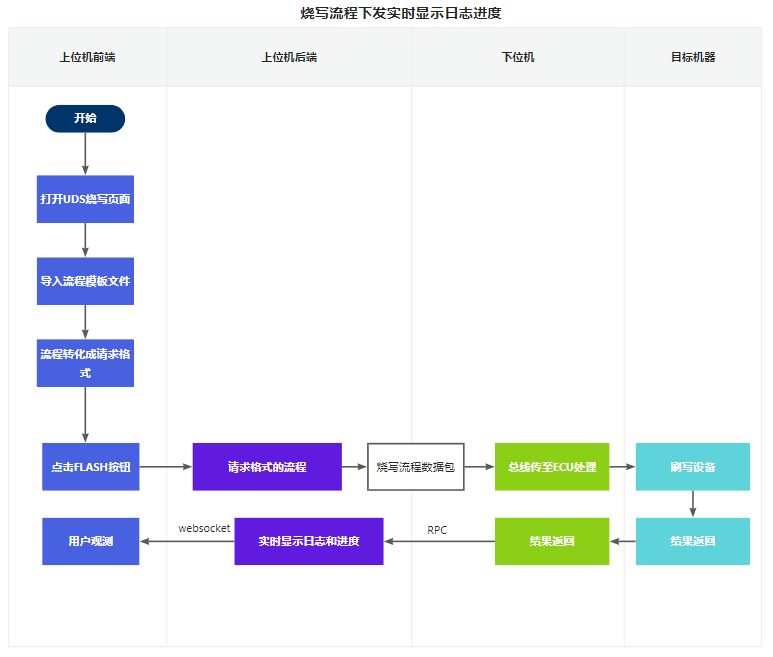
1. 流程图：



1. 涉及表和字段信息：无
2. 实现思路：
   1. 此功能点主要是 加载SeedKey文件和烧写流程配置。
   2. 其中SeedKey文件在烧写流程页面的右上角的全局设置按钮中配置，流程每一步骤点击右上角+号新增，步骤类型有三种:数据，文件，等待，每种类型有不同的控件配置参数，文件类型有一个发送设置按钮，点击后会弹出更详细的文件配置界面供用户配置，配置完毕后前端会将其转换成对应请求格式的烧写流程数据包。
   3. 导入之前保存的烧写流程模板文件可以一键配置烧写流程省去用户重复操作并将其转换成对应请求格式的烧写流程数据包。

##### 烧写流程下发实时显示日志进度功能点

1. 流程图：



1. 涉及表和字段信息：无
2. 实现思路：
   1. 此功能点主要是将一键导入或者用户手动配置的流程转换成的对应请求格式的烧写流程数据包烧写到对应的目标设备中。
   2. 烧写的同时将进度以及日志通过RPC上位机后端，并通过Websocket实时上传到前端显示。

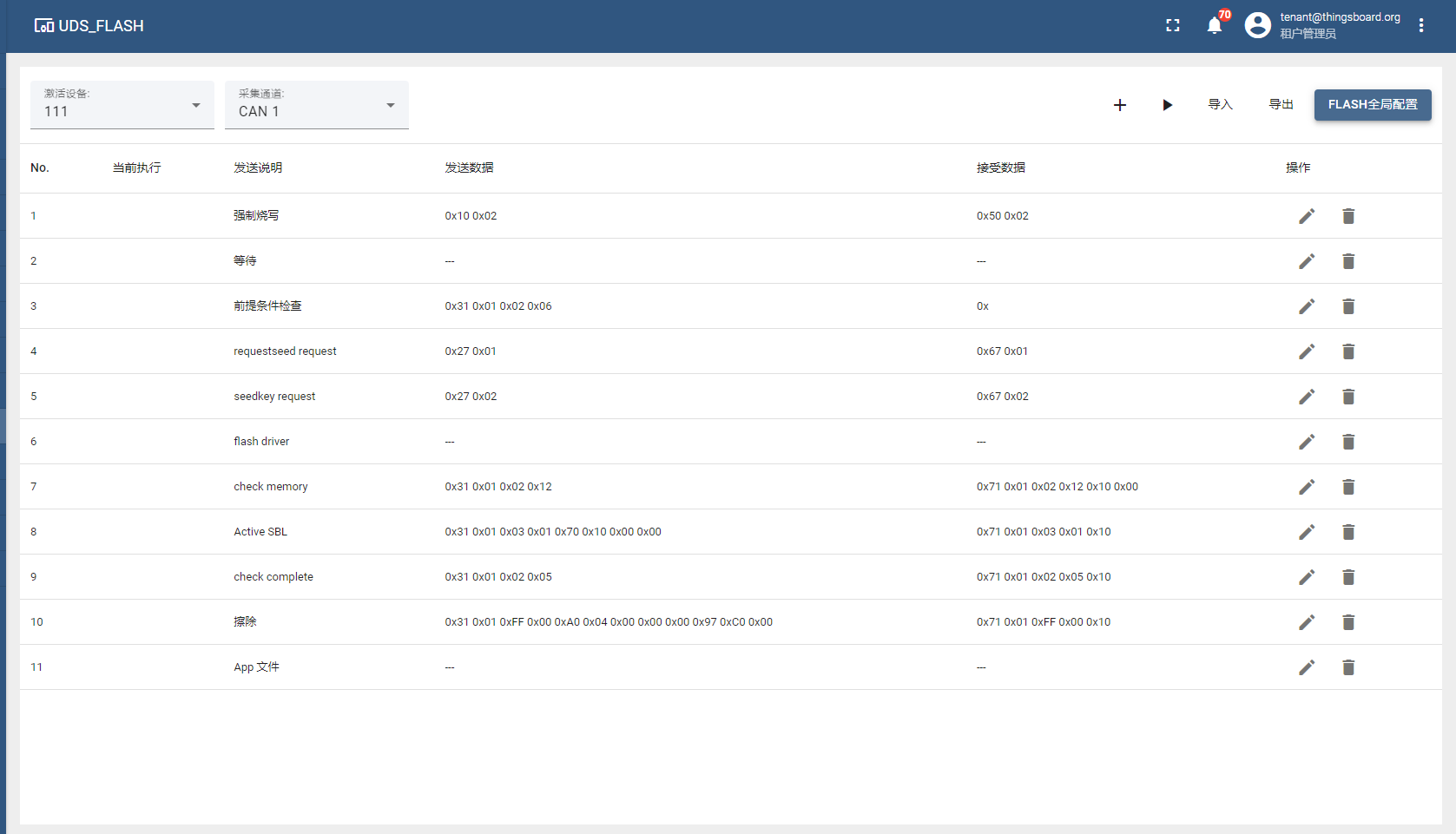
### 数据库设计

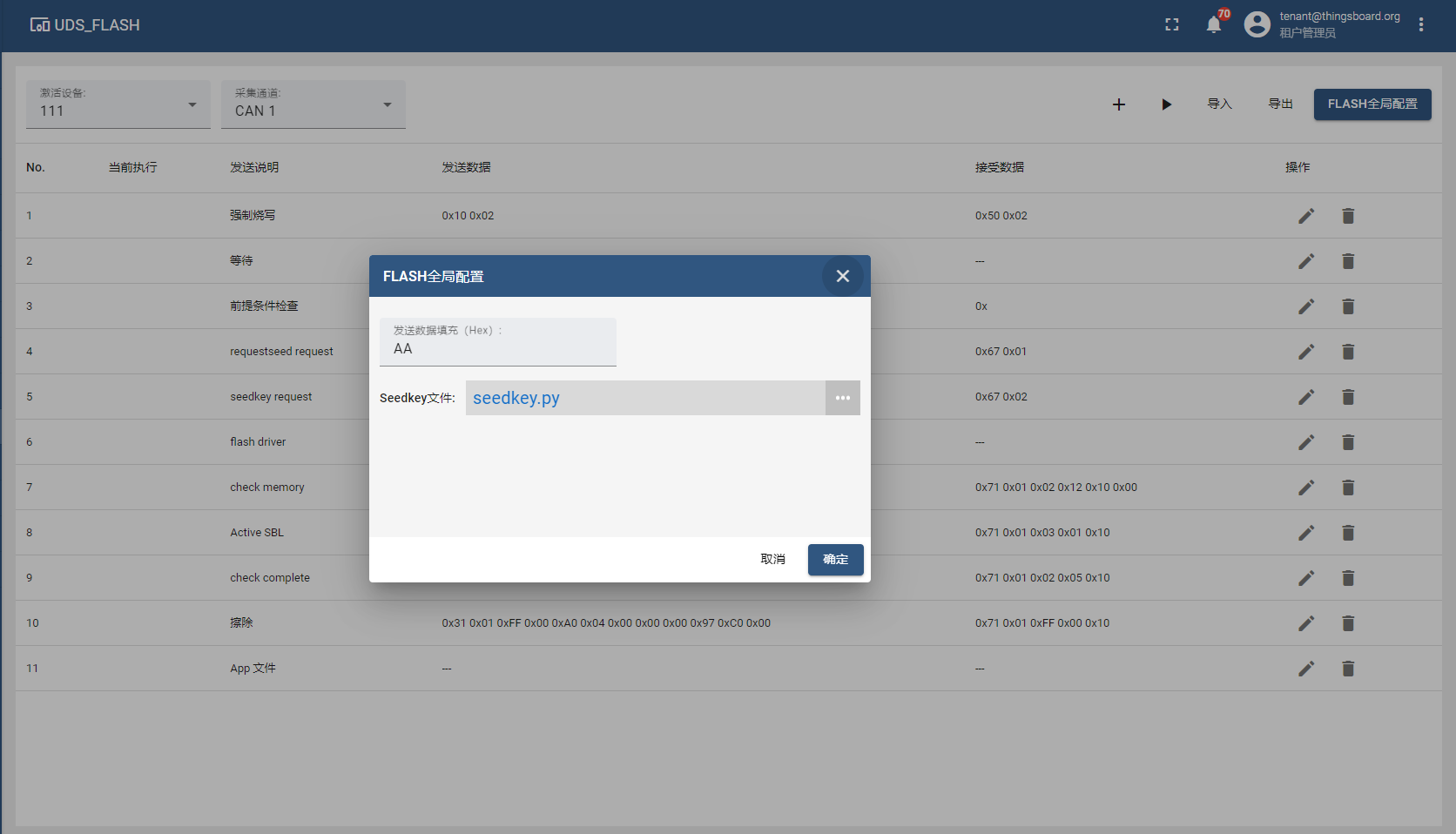
无

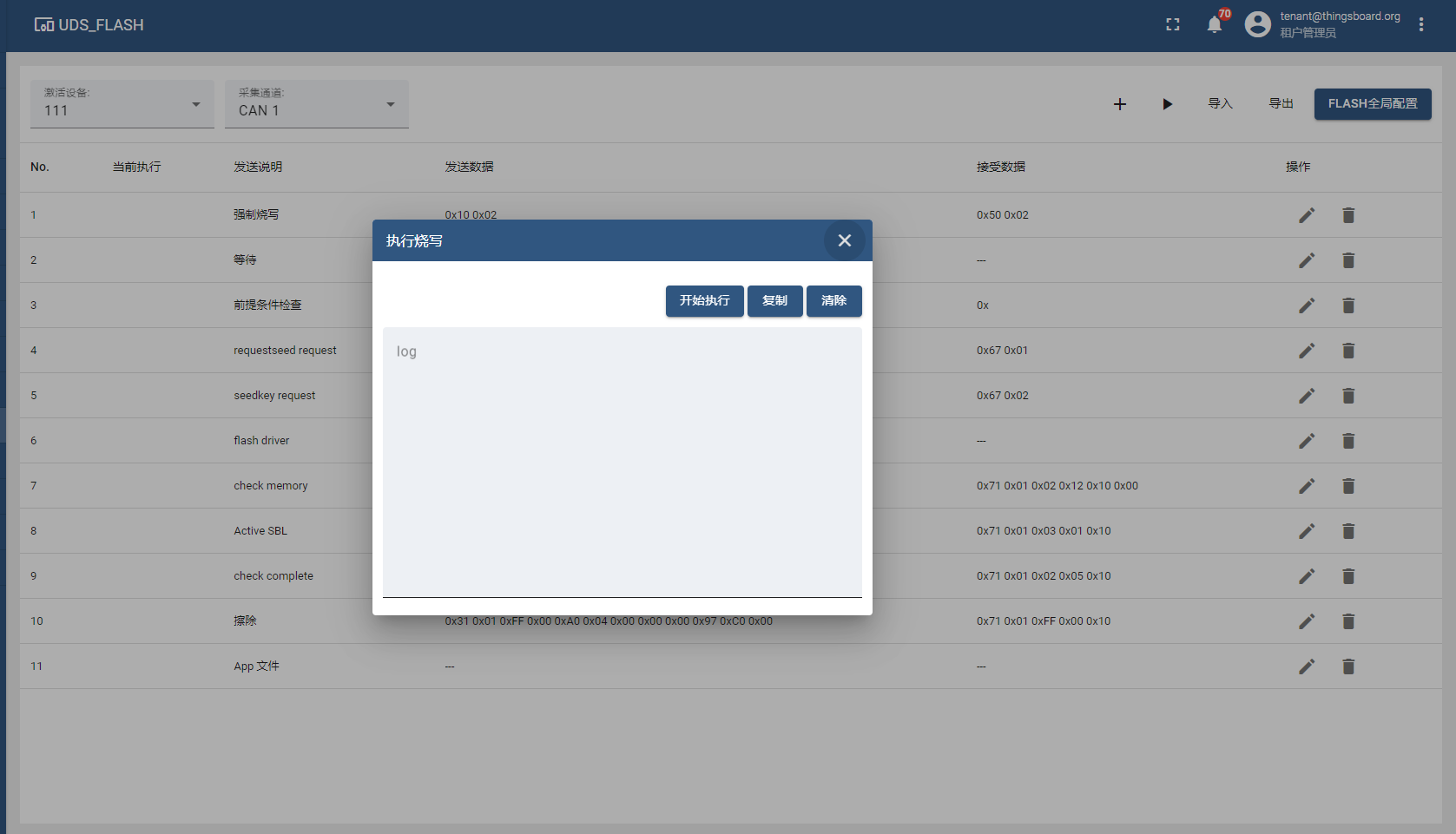
### 关键类设计

UdsFlashHandler.java，TbMsgUdsFlashTransportNode.java

### UI/UE设计



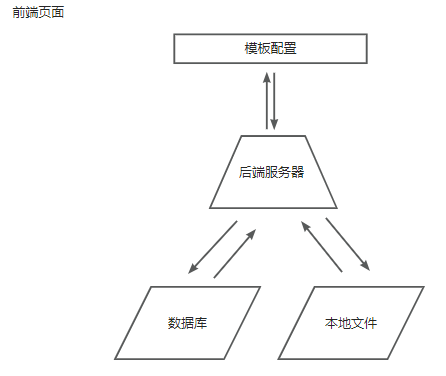




## 模板导入导出应用功能设计

### 模板导入导出应用总体结构

（Edge端(T-Box)和ECU控制器是辅助说明内容，非本模块功能部分）



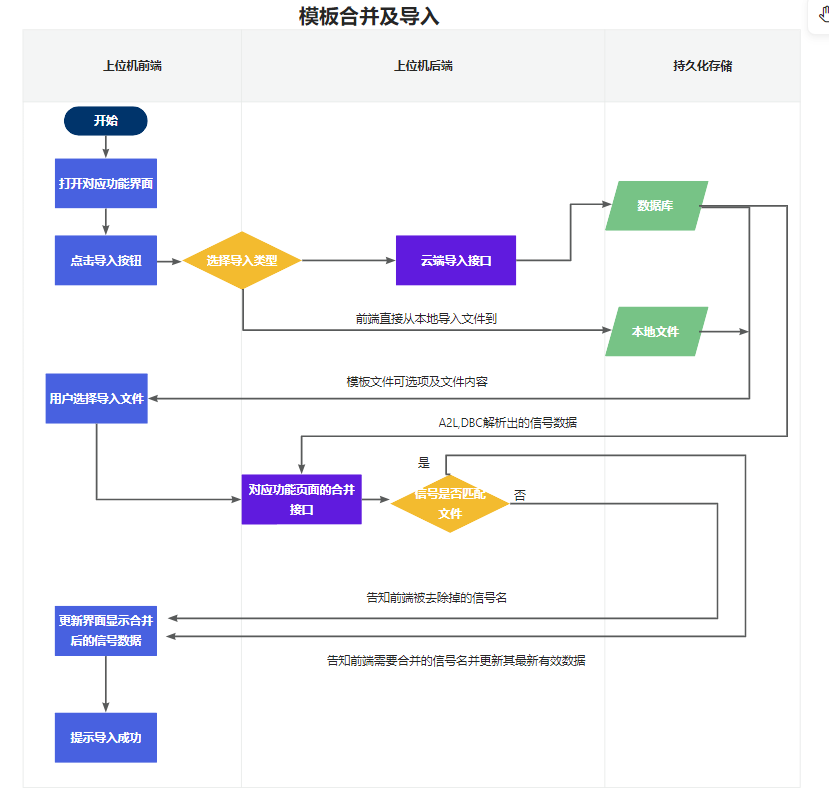
### 模块及功能点划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块代码** | **模块名称** | **功能点编号**  **（系统需求ID）** | **功能点名称**  **（系统需求名称）** | **功能点简要描述** |
| MOD-TBOX-004 | 模板导入导出 |  | 模板信号合并及导入 | 导入模板时筛选出模板当前通道下数据源文件(DBC,CDD,A2L)中含有的信号，并更新其有效字段的最新数据 |
| MOD-TBOX-004 | 模板导入导出 |  | 云端及本地导出 | 从对应功能页面(边缘计算、XCP、Kafka信号列表)导出当前页面配置至云端数据库或本地文件中，便于下次使用时导入对应功能页面中。 |

#### 功能点逻辑设计

##### 模板信号合并及导入

1. 流程图



1. 涉及表和字段信息:

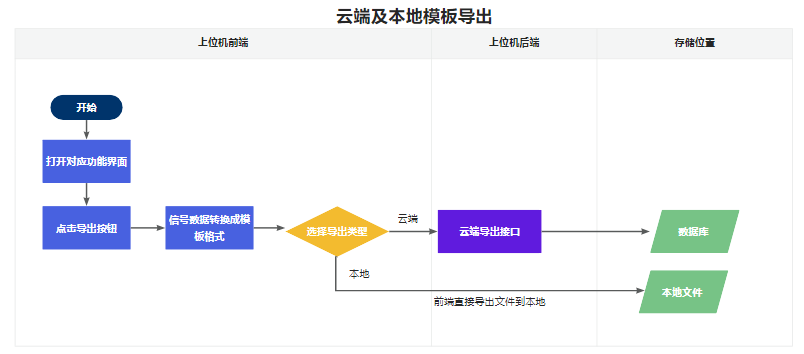
Table:template

Column:file\_content

1. 实现思路:
   1. 此功能点主要是将云端或者本地存储的模板文件导入至对应功能模块(边缘计算采集、XCP采集、XCP标定、Kafka信号列表)一键配置。
   2. 选择云端导入，前端会请求后端接口查询数据库之前导出的模板供用户选择，而本地导入则直接选取本地模板文件，选取的模板文件数据会发送给后端用于合并的接口，筛选出信号名还存在于当前数据源文件(DBC或A2L文件)的信号并更新其有效字段的最新数据，告知前端不存在于当前数据源文件的信号并提示用户，然后把筛选出的有效信号和原本存在于当前模块下的信号进行合并，显示与前端配置界面之中，并提示导入成功

##### 云端及本地导出

1. 流程图



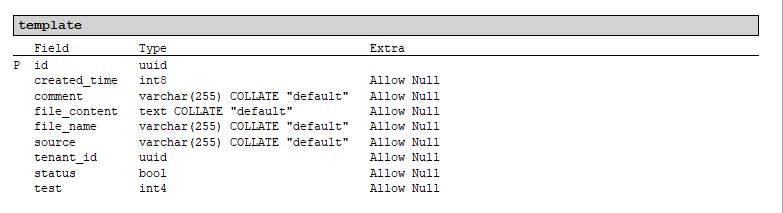
1. 涉及表和字段信息

Table:template

Column:file\_content

1. 实现思路:
   1. 此功能点主要是将对应功能模块(边缘计算采集、XCP采集、XCP标定、Kafka信号列表)配置好的数据作为模板存储到云端或者本地的文件以便下次一键导入。
   2. 点击导出按钮，对应功能模块配置好的数据会转化成模板的格式，可以选择云端导出或者本地导出，本地导出由前端直接下载到本地，而云端导出是调用后端接口存储到数据库中。

### 数据库设计

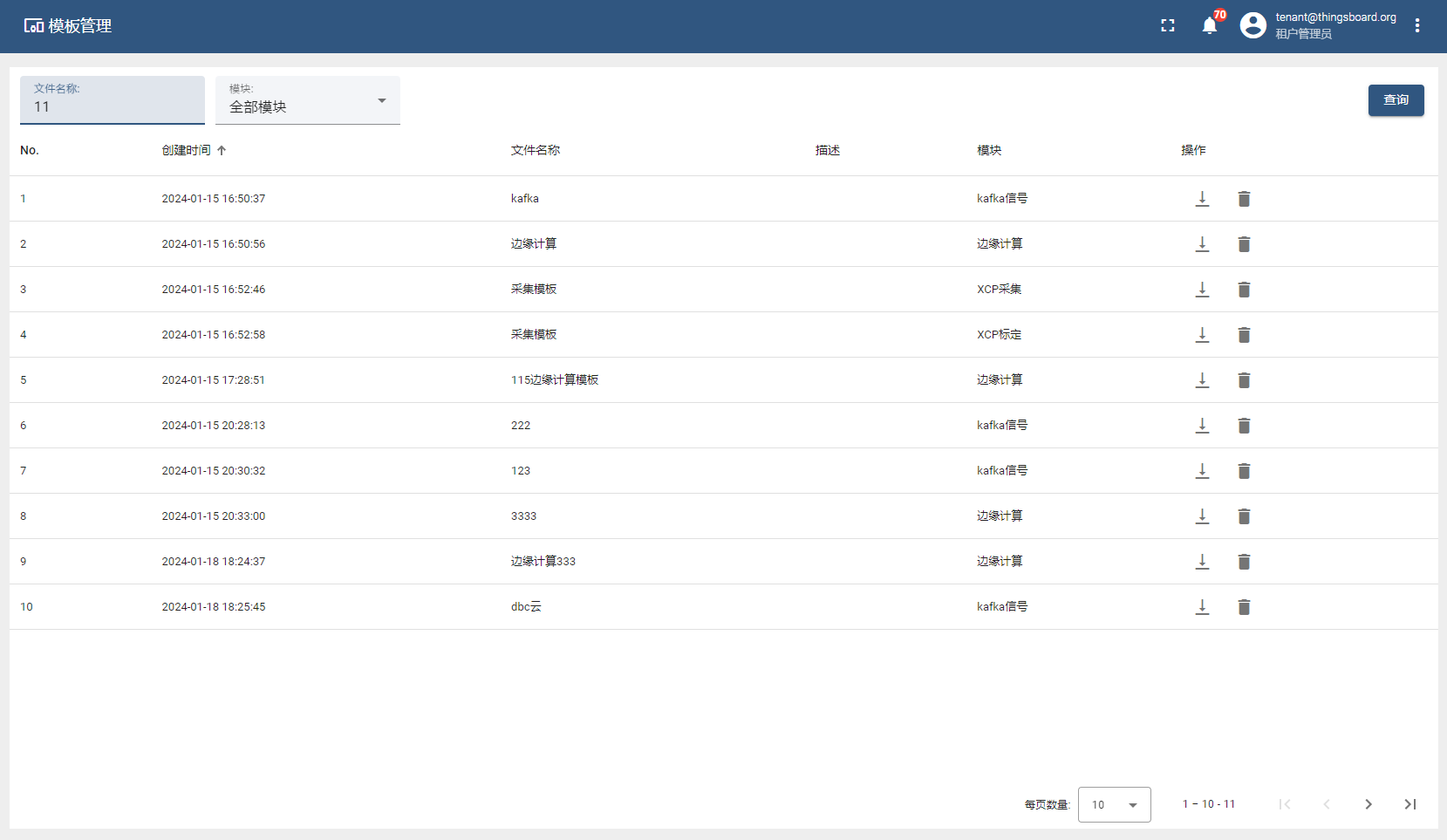


### 关键类设计

Template.java,TemplateEntity.java

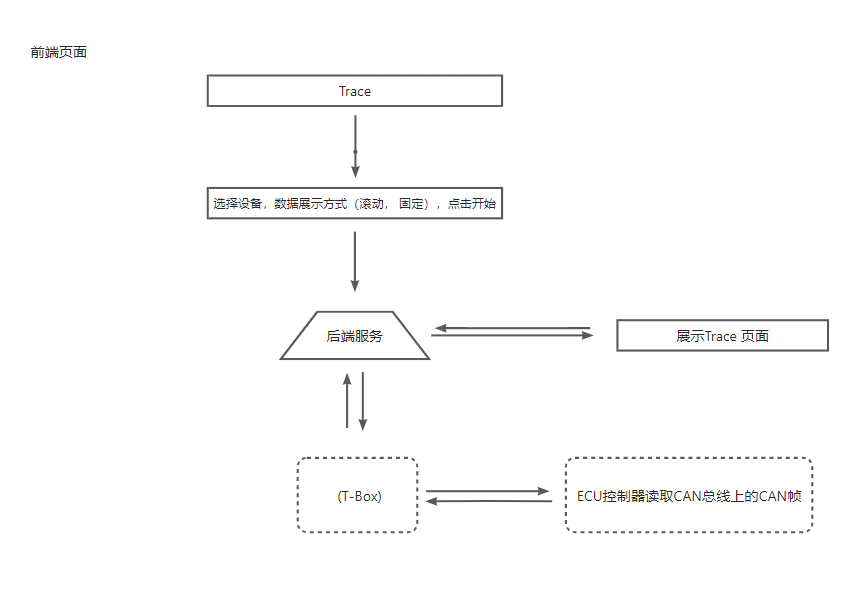
### UI/UE设计





## Trace应用功能设计

### Trace总体设计



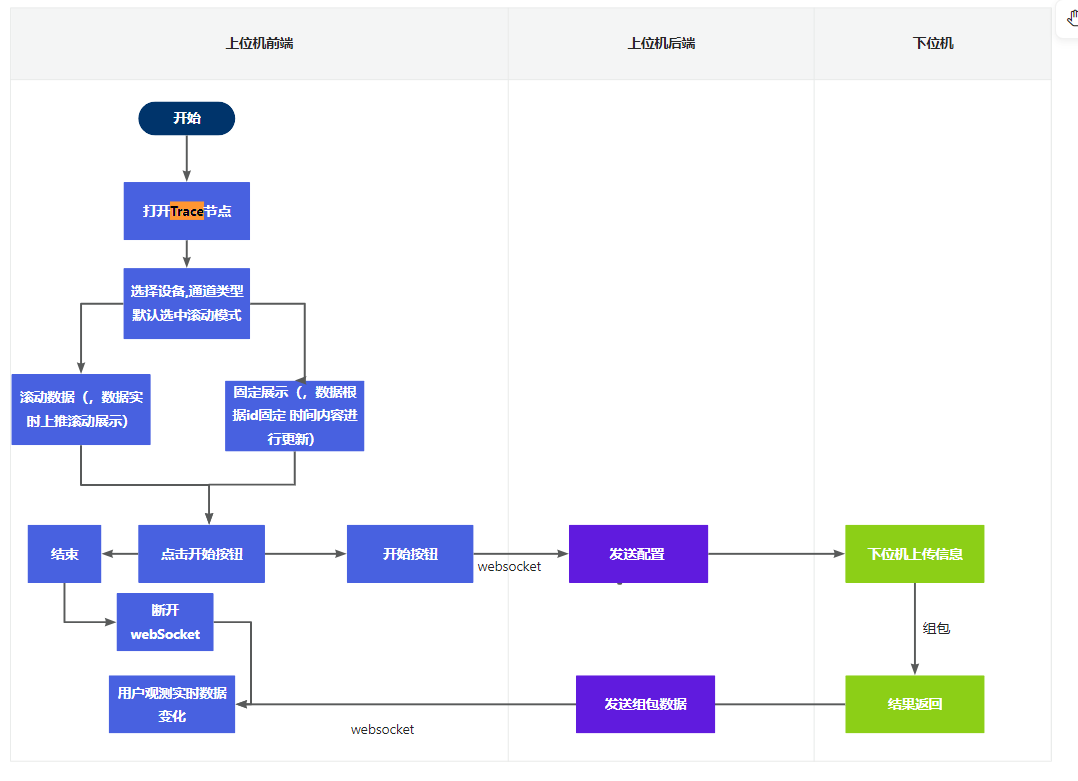
### 模块及功能点划分

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **模块代码** | **模块名称** | **功能点编号**  **（系统需求ID）** | **功能点名称**  **（系统需求名称）** | **功能点简要描述** |
| MOD-TBOX-003 | Trace |  | Trace 搭建 | 选择激活设备后默认滚动模式下，向后端通过Websocket发送请求的设备 ，读取CAN 总线上的CAN帧 |

#### 功能点逻辑设计

##### Trace搭建

1. 流程图



2 、实现思路：

1. 前端获取当前账号下所有激活设备， 默认选中，滚动触发，没有选择设备时，开始 和结束仿真按钮置灰
2. 选择设备后，点击开始按钮，通过Websocket, 向后端发送请求数据命令， 后端进行实时数据推送，前端根据服务端推送过来的数据累积记录ID 和通道数据源 ， 前端根据选择的模式差异化展示（滚动模式，前端接收数据实时展示在当前可视页面区域），（固定模式。前端根据累积的ID ，对对应ID 下数据进行实时更新）
3. （通道过滤）对用户勾选的通道， 对实时推送过来的数据，进行通道过滤， （ID过滤）类似通道过滤，对推动过来的id 实时拦截过滤

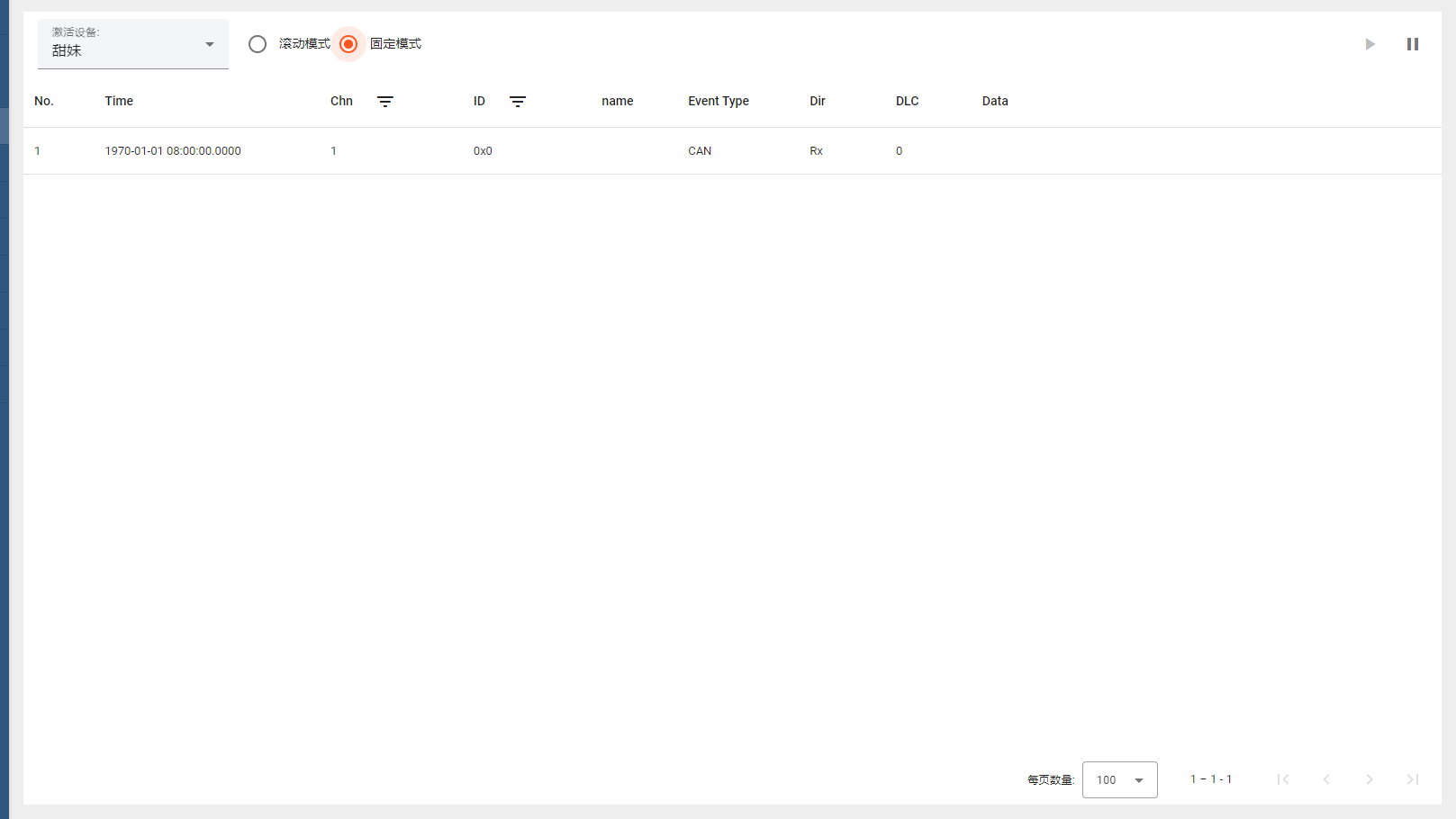
### 数据库设计

无

### UI/UE设计

图形用户界面, 应用程序

描述已自动生成



# 外部接口详细设计

## 硬件接口设计

参考《数据收集标定平台接口说明书》

## 软件接口设计

参考《数据收集标定平台接口说明书》

# 数据迁移

一、涉及历史数据范围：有哪些实体清单？

二、针对每个数据实体，定义源头及迁移清洗规则

# 非功能性设计

## 性能设计

DocapCloud需要能够处理大量的TBOX设备和数据流。因此，在性能设计方面，需要考虑以下几个方面：

* 可扩展性：设计支持水平扩展的架构，以适应不断增长的设备和数据量。
* 高吞吐量：优化数据存储和检索机制，以实现高吞吐量的数据处理。
* 快速响应时间：确保系统能够及时响应设备请求并处理数据。

## 安全性设计

|  |  |  |
| --- | --- | --- |
| **需求属性** | **需求描述** | **备注** |
| 1强认证 | 1.1系统采用对接集团统一身份认证管理平台 | 可选,内部系统优先选择此条款，可忽略第二条要求。如后台为内部、对外提供服务，内部可采用对接统一身份认证管理平台，外部采用1.2需求 |
| 1.2系统使用本身的认证机制，需满足强密码策略和锁定策略，并在保证基础用户体验的情况下，选择合适的图形验证码和短信验证码； | 内部系统如已对接统一身份认证，此条可不采用。如涉及此需求的，在选择图形验证码时优先采用集团已购置的“极验”图形验证码模块。 |
| 2 访问控制最小化 | 2.1 系统合理划分各种类型的权限管理，包括管理员、审计员、操作员，保证三权分立； | 必选 |
| 2.2 系统权限控制的管理粒度达到表单的角色级别（或单个用户的增删改查） | 明确系统权限的控制粒度，选择需求中一种或两种。 |
| 3 会话的安全性 | 3.1 系统会话标识随机并唯一，登录后重新分配，并对会话的失效周期进行定义 | 必选 |
| 4 日志安全审计 | 4.1 系统日志需以功能模块方式提供，日志记录覆盖每个用户，无论是内部用户或外部用户，记录应对重要的操作行为（登入登出、权限变更、数据导入导出等其他必要的操作）进行记录，并具有筛选功能；日志记录的内容应包含日期、时间、用户标识、行为的过程和结果等 | 必选，其他必要的操作行为记录可根据应用系统的不同业务定义，对业务审计有需求的内容，各编制人员可自行补充； |
| 4.2 日志记录的管理权限不应对基础用户提供，对开放给审计员，并禁止提供删除和修改的功能。 | 必选 |
| 5 软件容错 | 5.1 系统具备容错设计，提供可用性，对异常或错误的页面或功能进行跳转； | 必选 |
| 6 数据安全 | 6.1 应考虑系统数据的保密性和完整性，包括数据的传输加密https，敏感数据存储加密 | 必须 |
| 6.2应用系统评估隐私数据使用情况，系统涉及到哪些个人隐私数据，如姓名、手机号、地址、定位、轨迹等。 | 必须，个人隐私数据保护应考虑收集最小化、数据传输加密、存储加密、脱敏处理的安全措施 |
| 7 输入输出安全性 | 5.4.7.1 应用系统应对所有类型的用户输入做校验，尤其是那些用于数据库操作、系统命令执行、程序命令执行、显示到页面的重要参数，并记录拒绝后的错误日志。 | 必须 |
| 5.4.7.2 应用系统应对输出参数做严格的限制，保证非功能必须的参数不在输出的参数中，防止多余信息引起的数据泄露。 | 必须 |

### 强认证

系统采取对集团统一身份认证管理平台，系统会通过SSO方式，对接统一身份认证管理平台，通过内部登录系统一键登录DocapCloud系统。

### 访问控制最小化

* DocapCloud中有三种用户角色，分别为管理员、租户和客户端用户。管理员可以管理所有租户和设备，而租户只能管理自己的设备。客户端用户是指通过应用程序或API访问设备数据的用户。
* 用户组：可以将用户分组，以便在权限分配方面更加灵活。例如，可以将某些用户添加到特定的用户组中，并将该组与特定的设备或资源相关联。
* 权限分配：DocapCloud允许管理员为每个用户和用户组分配不同的权限。例如，管理员可以限制某些用户或用户组的访问范围，或者只允许他们执行特定的操作。
* 设备级别权限：在DocapCloud中，管理员和租户可以为特定的设备配置不同的权限。例如，管理员可以限制某些设备的访问权限，而租户可以为其自己的设备定义特定的访问权限
* DocapCloud允许不同的租户分别使用自己的数据存储，防止数据交叉和冲突。每个租户都有自己的数据存储空间，而且只能访问自己的数据。

### 会话的安全性

系统会话标识随机并唯一，登录后重新分配，并对会话的失效周期进行定义

* 会话管理：DocapCloud使用基于令牌的会话管理机制，通过使用JSON Web Token (JWT)来验证用户身份和保护用户会话。当用户成功登录后，服务器会生成一个包含用户身份信息的JWT令牌，并将其存储在用户的浏览器cookie中。之后，用户每次访问相关页面时，都需要将该令牌发送到服务器以进行身份验证。
* 令牌加密：为了保证令牌的安全性，DocapCloud使用RSA加密算法对JWT令牌进行加密。这样，即使黑客获取了JWT令牌，在没有正确的私钥的情况下也无法解密其中的内容。
* 会话超时：为了防止会话劫持和跨站脚本攻击，DocapCloud会自动将长时间不活动的会话关闭，并要求用户重新登录。管理员可以设置会话超时时间，以便控制用户会话的持续时间。
* HTTPS连接：DocapCloud还要求所有用户使用HTTPS连接进行通信，以确保数据在传输过程中的安全性和完整性。通过使用HTTPS，DocapCloud可以防止黑客对数据进行拦截、篡改或窃取。

### 日志安全审计

* 日志记录：DocapCloud会记录各种系统级别和用户级别的操作和事件，包括用户登录、设备连接、数据读写、规则引擎执行等。这些日志记录可以提供对系统运行状况和用户活动的全面可见性。
* 日志存储：DocapCloud将日志存储在数据库或文件中，以便后续查询和分析。您可以根据需要选择合适的存储方式，并配置日志保留时间。
* 访问控制：DocapCloud提供了访问控制机制，可以限制用户对敏感操作和功能的访问权限。通过正确配置用户角色和权限，可以确保只有授权用户才能执行重要操作，并记录其操作日志。
* 审计分析：通过分析DocapCloud的日志记录，您可以了解系统的整体运行情况、用户活动、设备连接和数据流动等信息。这些信息可用于故障排除、性能优化和安全审计。
* 日志监控：为了及时发现异常和安全事件，您可以使用监控工具或集成日志管理系统对DocapCloud的日志进行实时监控。这样可以提前发现潜在的问题，并采取适当的措施。
* 合规性要求：对于一些行业或法规标准，如GDPR、HIPAA等，安全审计是必需的。通过启用和配置DocapCloud的日志安全审计功能，可以满足合规性要求并提供相关报告。

### 软件容错

* 集群部署：DocapCloud支持将系统部署为集群，通过水平扩展来提高系统的可用性和容错性。集群中的每个节点都可以处理请求，并共享负载，当一个节点失败时，其他节点可以接管其工作，从而保持系统的连续性。
* 数据复制和同步：DocapCloud使用分布式数据库或消息代理来存储和同步数据。这些组件会自动复制数据到多个节点，确保即使某个节点发生故障，数据也不会丢失，并且其他节点可以继续提供服务。
* 故障检测和恢复：DocapCloud具有故障检测和自动恢复机制。系统会定期检测节点的健康状态，如果节点发生故障或不可用，则会自动将其标记为无效，并将请求转发到其他可用节点。一旦节点恢复正常，系统会自动重新加入集群。
* 负载均衡：DocapCloud使用负载均衡机制来分散请求的负载，以防止某个节点过载。负载均衡器会根据节点的负载情况，将请求合理地分发到可用的节点，从而保持系统的性能和稳定性。
* 容错处理：DocapCloud具备容错处理机制，可以在面对异常情况时保持系统的稳定性。例如，当设备连接中断或传感器故障时，系统会进行错误处理并采取适当的措施来修复或绕过问题，以确保数据流的连续性。
* 监控和告警：DocapCloud具备监控和告警功能，可以实时监测系统的状态和性能指标。通过设置适当的阈值和规则，系统可以自动触发告警并通知管理员或运维人员，以便及时处理潜在的故障和异常情况。

### 数据安全

* 数据加密：DocapCloud使用TLS/SSL协议对所有传输的数据进行加密，包括设备数据、用户数据、配置数据等。这种加密可以防止数据在传输过程中被窃听或篡改，从而保证数据的机密性和完整性。
* 访问控制：DocapCloud采用访问控制机制来限制用户对敏感数据和功能的访问权限。通过设置角色和权限，管理员可以控制用户对数据的读写访问，并记录其操作日志。这种访问控制机制可以保证数据的机密性和完整性。
* 身份验证和授权：DocapCloud使用OAuth2和JWT等身份验证和授权协议来管理用户身份和权限。通过验证用户身份和授权机制，系统可以确保只有经过身份验证的用户才能访问敏感数据和功能。这种身份验证和授权机制可以保证数据的机密性和完整性。
* 数据备份和恢复：DocapCloud使用数据库或文件存储数据，并定期进行数据备份。这种备份可以保证数据在系统故障或数据丢失的情况下能够恢复，从而保证数据的可用性和完整性。
* 审计和监控：DocapCloud具备审计和监控功能，可以记录和监控所有系统操作和事件。通过分析日志记录和监控数据，管理员可以监测系统的安全状况，并快速识别潜在威胁和异常情况。这种审计和监控机制可以保证数据的机密性、完整性和可用性。
* 合规性要求：在一些行业或法规标准，如GDPR、HIPAA等，对数据安全性有着严格的要求。DocapCloud的数据安全设计满足这些要求，并提供相关的报告和证明，以确保数据的机密性、完整性和可用性。

### 输入输出安全性

* 输入安全性：DocapCloud具有输入安全性机制，可以防止恶意用户对系统进行攻击或注入伪造数据。具体来说，DocapCloud使用数据验证和过滤机制来确保输入数据的合法性、完整性和准确性。例如，系统会验证设备标识符、传感器数据格式、用户权限等等，以防止非法数据的输入。
* 输出安全性：DocapCloud具有输出安全性机制，可以确保系统输出的数据和信息是准确、完整、机密和可信的。具体来说，系统会通过访问控制机制、身份验证和授权机制、加密机制等等，来保护输出数据的机密性、完整性和可信性。例如，系统会限制用户对敏感数据的访问权限、使用TLS/SSL协议加密输出数据、记录操作日志等等，以确保输出数据的安全性。
* 数据备份和恢复：DocapCloud采用数据库或文件存储数据，并定期进行数据备份。这种备份可以保证数据在系统故障或数据丢失的情况下能够恢复，从而保证数据的可用性和完整性。

## 其他非功能性设计

* 可靠性设计：以下是Docap在可靠性设计方面的实现：
* 容错性：设计容错机制，以应对设备故障或通信中断等异常情况，并确保系统的连续性。
* 数据冗余和备份：为了保证数据的安全性和可靠性，可以考虑使用数据冗余和备份策略，防止数据丢失。
* 可用性设计：为了提供良好的用户体验，可用性设计方面做了以下几点：
* 用户界面设计：设计直观、易用的用户界面，使用户能够轻松地管理和监控物联网设备。
* 错误处理和用户反馈：提供有效的错误处理机制，并及时向用户提供反馈信息，以帮助用户解决问题。

# 附录

## 【模板】接口清单



## 【模板】数据库清单

