

# 小米电池预警系统

## 1. 需求分析

### 1.1 项目背景

- BMS系统（电池管理系统）是智能化管理及维护各个电池单元，防止电池出现过充电和过放电、延长电池的使用寿命、监控电池状态的系统。在BMS系统中存在大量电池各种信号的规则管理以及监控，良好的信号处理，并且根据规则，生成相关预警信息，能够极大地提升用户体验。

### 1.2 功能需求

#### 1.2.1 车辆信息存储

- 支持车辆信息（vid, 车架编号, 电池类型, 总里程(km), 电池健康状态(%)）的录入。
- 车辆信息录入的原因是：先有车才有电池，最后才会在车行驶中产生电流信号。
- 需要设计车辆信息存储的表结构，并将给定的信息存储到汽车信息表中。

#### 1.2.2 规则配置

- 支持规则（包括：序号，规则编号，名称，预警规则，电池类型）的配置。
- 预警规则包含预警规则描述以及预警等级（0级最高响应）。
- 电池类型：不同类型电池对应规则不同。
- 信号：Mx（最高电压），Mi（最小电压）、Ix（最高电流），Ii（最小电流）。
- 需要设计车辆上报信号和规则的存储的表结构，并将给定的数据存储在规则表中。

#### 1.2.3 上报电池信号功能

- 能通过接口上报电池信号状态，完成数据库的增删改查。

### 1. 查询电池信号功能

- 查询电池信号状态，要求接口使用Redis做缓存，且保证缓存和数据库数据的一致性。

#### 1.2.4 预警功能

- 通过定时任务扫描电池信号数据，通过发送MQ消息，消费MQ消息生成预警信息。
- 支持通过预警接口查询指定车辆的预警信息。

## 1.3 技术需求

### 1.3.1 技术栈

- Java技术栈，SpringCloudAlibaba，SpringBoot 2.0+，Mybyties，Mysql 5.7+，Redis，http/https，MQ。

### 1.3.2 技术方案

- 必须包含系统设计、数据库表设计、接口设计、缓存和数据库数据一致性、单元测试。
- 所有接口必须100%单测覆盖。

## 1.4 性能需求

- 系统每天处理信号量为百万甚至千万数据级别：考虑数据量对系统性能的影响，给出合理设计数据存储和查询方案。

## 2. 可行性分析

### 2.1 技术可行性

#### 2.1.1 Java技术栈

- Java技术栈成熟稳定，SpringBoot框架提供了快速开发和部署的能力，Mysql数据库广泛应用于企业级应用，Redis作为缓存解决方案能够有效提升查询性能，MQ（如RabbitMQ、Kafka等）能够实现高效的异步消息处理。

#### 2.1.2 性能优化

- 通过Redis缓存电池信号状态，减少数据库的直接访问，提高查询效率。
- 使用MQ进行异步处理，避免同步调用带来的性能瓶颈。
- 对规则解析和预警计算进行性能测试和优化，确保P99响应时间在1s以内。

### 2.2 经济可行性

#### 2.2.1 开发成本

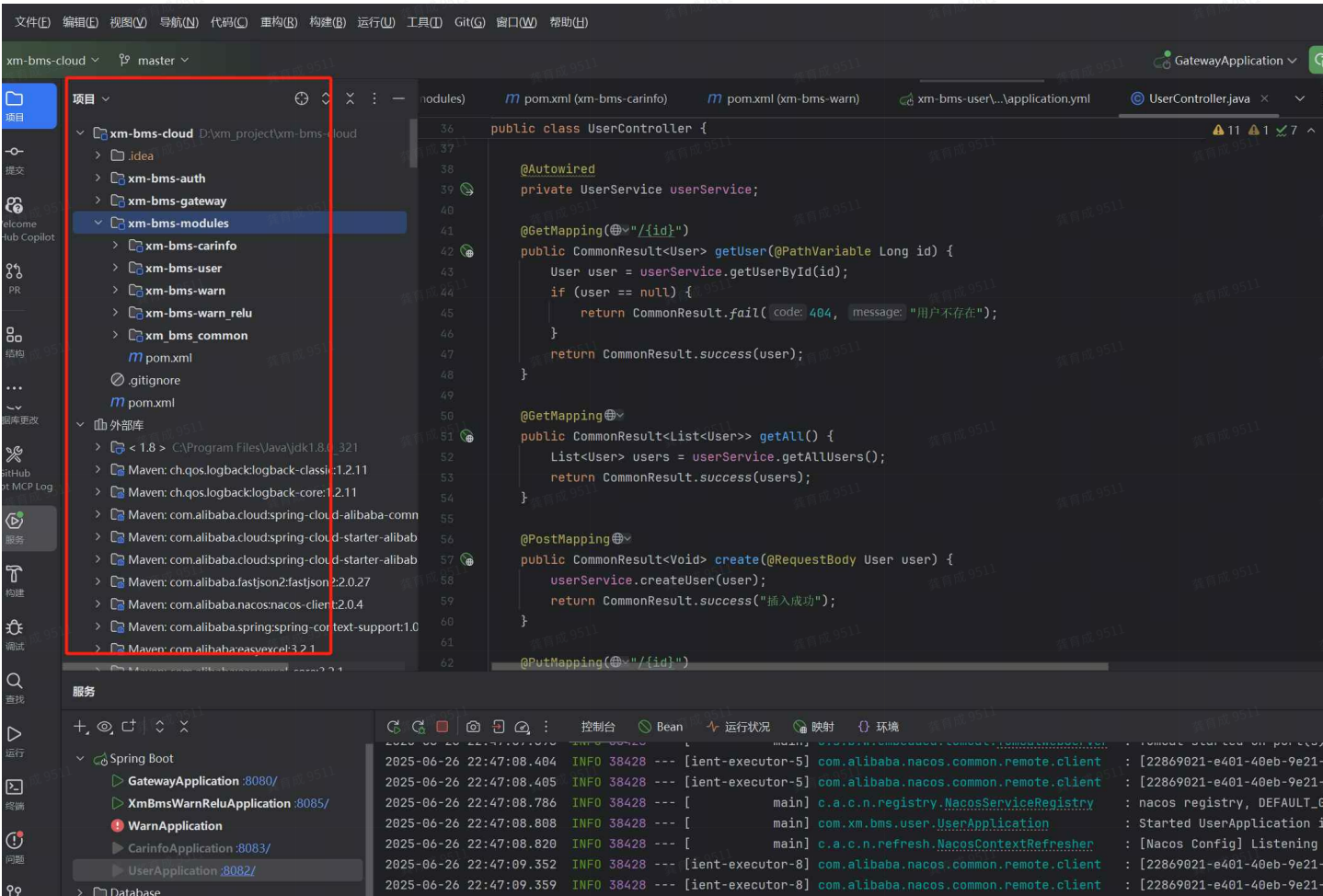
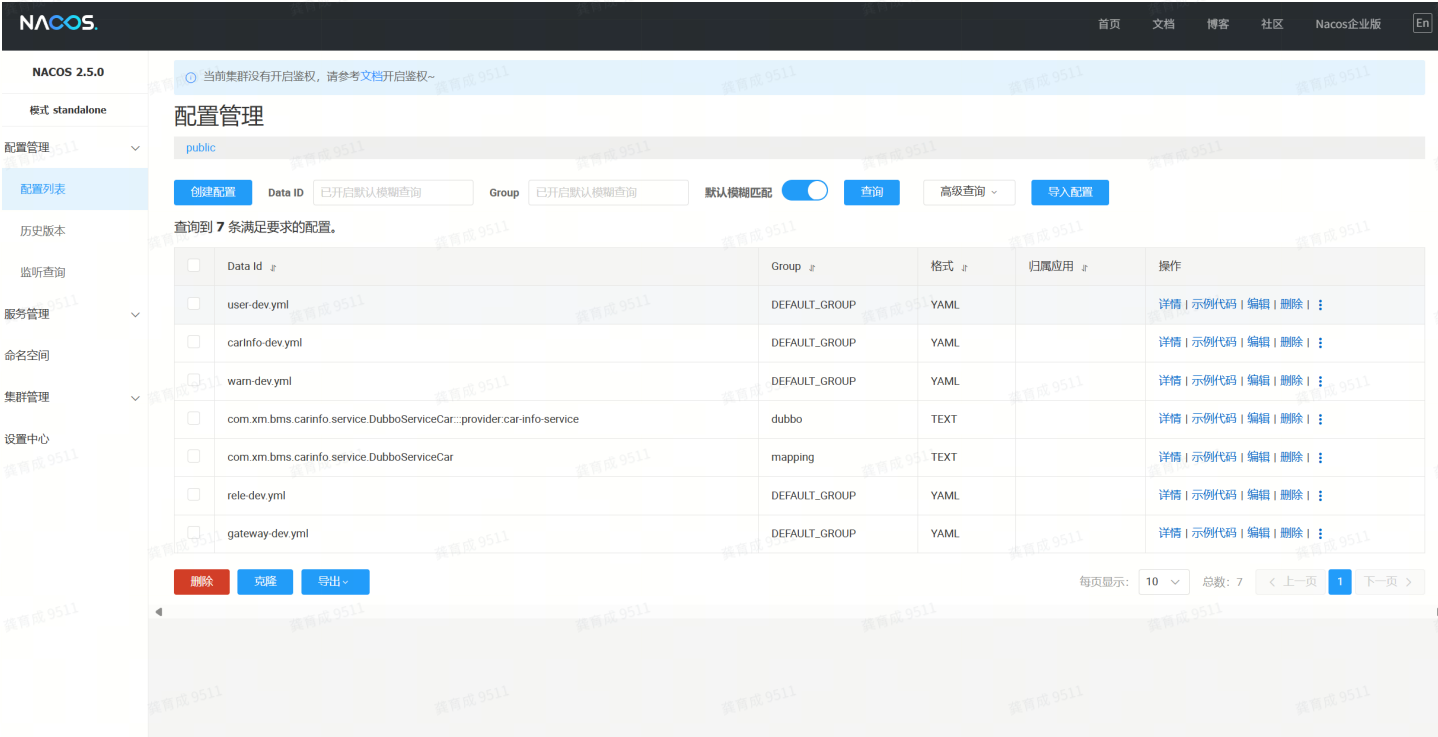
- 使用开源技术栈（SpringBoot、Mysql、Redis、MQ），开发成本较低。
- 项目规模适中，开发周期可控，人力成本在合理范围内。

#### 2.2.2 运营成本

- 系统基于成熟的开源技术栈，运维成本较低。
- 数据存储和查询方案合理，能够有效应对大规模数据量，减少硬件投入。

### 3. 总体设计

- 使用nacos作为注册中心和配置中心。



## 3.1 系统架构

### 3.1.1 分层架构

- 数据层：Mysql数据库存储车辆信息、用户信息、规则配置、电池信号数据、预警信息等。
- 缓存层：Redis缓存电池信号状态和用户信息，提高查询性能。
- 业务逻辑层：SpringBoot框架实现业务逻辑，包括用户信息管理、车辆信息管理、车辆预警信息管理、电池预警规则管理等。
- 消息队列层：MQ实现异步消息处理，提高系统性能。
- 接口层：提供RESTful API接口，供前端或其他系统调用。
- 网关层：使用Spring Cloud Gateway实现API网关，进行请求路由、权限控制等。
- 公共模块：用于统一异常处理和结果返回包装。

## 3.2 系统模块

### 3.2.1 网关模块（Gateway）

- 负责请求路由、权限验证、限流等功能。

### 3.2.2 权限控制模块（Auth）

- 负责用户认证、授权、权限管理等功能。

### 3.2.3 用户信息管理模块（User Management）

- 负责用户信息的录入、查询、更新和删除。（批量导入或更新，导出列表，导入模板下载）

### 3.2.4 车辆信息管理模块（Vehicle Management）

- 负责车辆信息和车辆状态信息的录入、查询、更新和删除。（批量导入或更新，导出列表，导入模板下载）

### 3.2.5 车辆预警信息管理模块（Vehicle Warn Management）

- 负责车辆预警信息的生成、查询和管理。（批量导入或更新，导出列表，导入模板下载）

### 3.2.6 电池预警规则管理模块（Battery Warn Rule Management）

- 负责电池预警规则的添加、查询、更新和删除。

### 3.2.7 公共模块（Common）

- 提供统一异常处理、结果返回包装等通用功能。

## 4. 详细设计

### 4.1 数据库表设计

#### 4.1.1 用户信息表 (user\_info)

- user\_id (INT, 主键): 用户ID。
- username (VARCHAR(50)): 用户名。
- password (VARCHAR(100)): 密码 (加密存储)。
- role (VARCHAR(20)): 用户角色 (如管理员、工程师、车主等)。
- create\_time (DATETIME): 创建时间。

#### 4.1.2 车辆信息表 (vehicle\_info)

- vid (VARCHAR(16), 主键): 车辆识别码, 每辆车唯一。
- car\_id (INT): 车架编号。
- battery\_type (VARCHAR(20)): 电池类型 (三元电池、铁锂电池)。
- total\_km (INT): 总里程 (km)。
- battery\_health (INT): 电池健康状态 (%)。
- user\_id (INT): 所属用户ID。

#### 4.1.3 预警规则配置表 (rule\_config)

- rule\_id (INT, 主键): 规则编号。
- rule\_name (VARCHAR(50)): 规则名称。
- battery\_type (VARCHAR(20)): 电池类型。
- warn\_rule (TEXT): 预警规则描述。
- warn\_level (INT): 预警等级。

#### 4.1.4 车辆信号表 (battery\_signal)

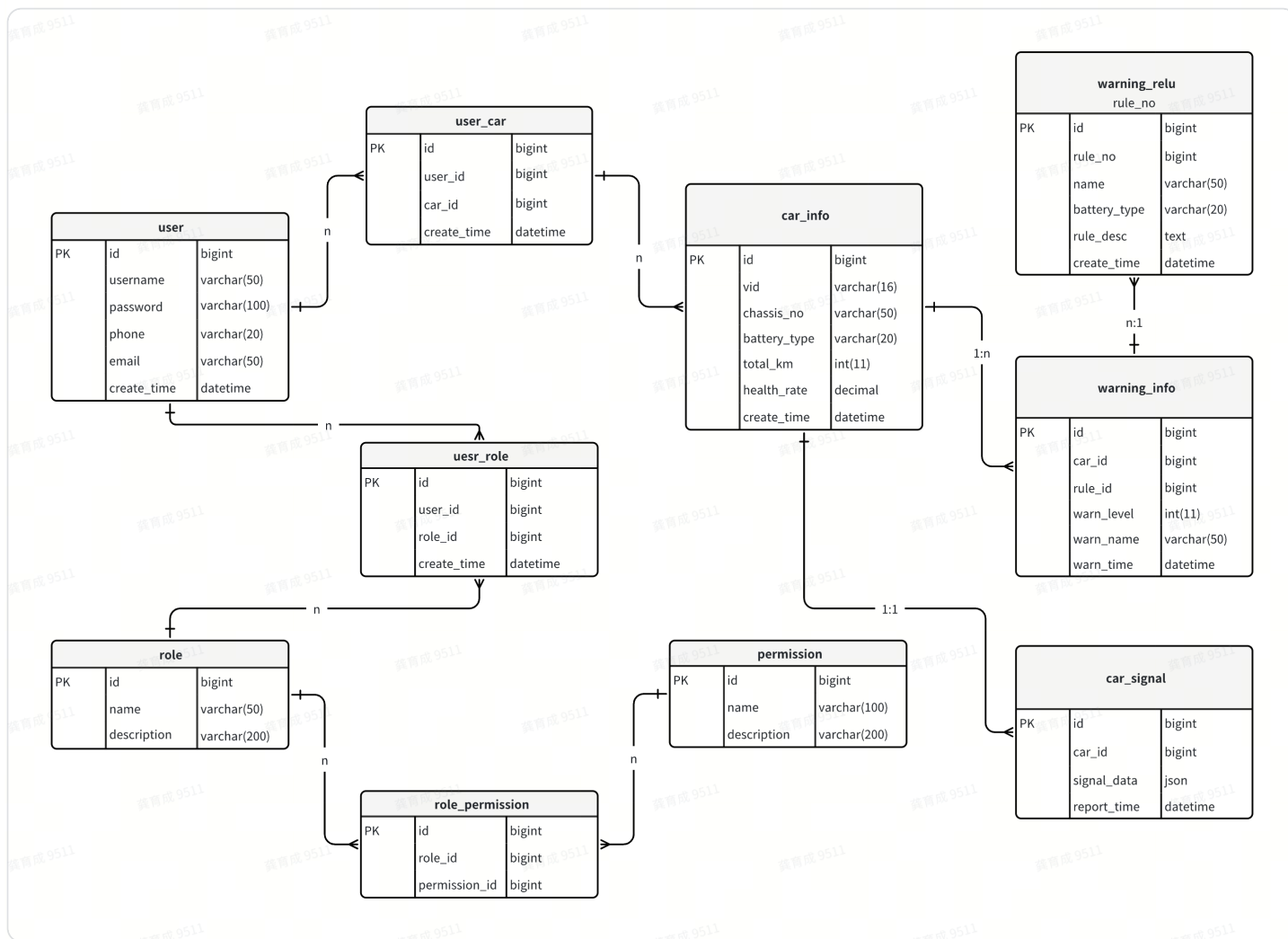
- id (INT, 主键): 信号编号。
- car\_id (INT): 车架编号。
- signal\_data (TEXT): 信号数据 (JSON格式)。
- create\_time (DATETIME): 信号上报时间。

#### 4.1.5 预警信息表 (warn\_info)

- warn\_id (INT, 主键): 预警编号。

- `car_id` (INT): 车架编号。
- `warn_name` (VARCHAR(50)): 预警名称。
- `warn_level` (INT): 预警等级。
- `create_time` (DATETIME): 预警生成时间。

## 4.2 MYSQL放在云服务器上，减少本地主机压力



## 4.3 接口设计

### 4.3.1 网关模块（Gateway）

接口名: `/api/gateway`

接口方法: `POST`

Body:

`path` (String): 请求路径。

`token` (String): 用户令牌。



返回信息：

`code` (int): 状态码。

`message` (String): 请求成功或失败信息。

`data` (JSON): 路由后的响应数据。

## 4.3.2 用户信息管理模块 (User Management)

### 1. 用户信息查询接口

接口名: `/api/users/{id}` 接口方法: `GET` 功能描述:

根据用户ID查询用户信息。

如果用户存在，返回用户详细信息；如果用户不存在，返回404错误。

返回信息：

`code` (int): 状态码

`message` (String): 请求成功或失败信息。

`data` (User): 用户信息对象。

### 2. 用户信息列表查询接口

接口名: `/api/users` 接口方法: `GET` 功能描述:

查询所有用户信息。

返回信息：

`code` (int): 状态码。

`message` (String): 请求成功或失败信息。

`data` (List<User>

### 3. 用户信息创建接口

接口名: `/api/users` 接口方法: `POST` Body:

`username` (String): 用户名。

`password` (String): 密码。

`role` (String): 用户角色。

### 4. 根据用户ID更新用户信息。

返回信息：

`code` (int): 状态码。

`message` (String): 请求成功或失败信息。

### 用户信息删除接口

接口名: `/api/users/{id}` 接口方法: `DELETE` 功能描述:

5.根据用户ID删除用户信息。

如果用户存在，删除成功；如果用户不存在，返回404错误。

返回信息：

`code` (int): 状态码。

`message` (String): 请求成功或失败信息。

6.用户数据导出接口

接口名: `/api/users/export` 接口方法: `POST` 功能描述:

导出所有用户信息为Excel文件。

设置响应头，使浏览器下载文件。

返回信息：

7.Excel文件流。

用户数据导入接口

接口名: `/api/users/import` 接口方法: `POST` 功能描述:

导入用户信息Excel文件。

读取Excel文件内容，批量插入或更新用户信息。

返回信息：

`code` (int): 状态码。

`message` (String): 请求成功或失败信息。

8.用户导入模板下载接口

接口名: `/api/users/importTemplate` 接口方法: `GET` 功能描述:

提供用户信息导入模板的下载。

设置响应头，使浏览器下载文件。

返回信息：

Excel模板文件流。

### 4.3.3 车辆信息管理模块 (Vehicle Management)

车辆信息录入接口

接口名: `/api/carinfos/add`

接口方法: `POST`

Body:

`vid` (String): 车辆识别码。

`car_id` (int): 车架编号。



`battery_type` (String): 电池类型。

`total_km` (int): 总里程。

`battery_health` (int): 电池健康状态。

`user_id` (int): 所属用户ID。

返回信息:

`code` (int): 状态码。

`message` (String): 请求成功或失败信息。

车辆信息查询接口

接口名: `/api/vehicle/query`

接口方法: `GET`

参数:

`car_id` (int): 车架编号。

返回信息:

`code` (int): 状态码。

`message` (String): 请求成功或失败信息。

`data` (JSON): 车辆信息。

#### 4.3.4 车辆预警信息管理模块 (Vehicle Warn Management)

预警信息生成接口

接口名: `/api/warns/generate`

接口方法: `POST`

Body:

`car_id` (int): 车架编号。

`signal` (String): 信号数据 (JSON格式)。

返回信息:

`code` (int): 状态码。

`message` (String): 请求成功或失败信息。

预警信息查询接口

接口名: `/api/warn/query`

接口方法: `GET`

参数:

`car_id` (int): 车架编号。

返回信息:

`code` (int): 状态码。

`message` (String): 请求成功或失败信息。

`data` (JSON): 预警信息。

## 4.4 公共模块 (Common)

### 4.4.1 统一异常处理

- 捕获系统中所有的异常，返回统一的错误信息。

### 4.4.2 结果返回包装

- 将接口的返回结果进行统一包装，包括状态码、消息和数据。

## 4.5 缓存设计

- 使用Redis缓存用户信息和电池信号状态，减少数据库的直接访问，提高查询效率。

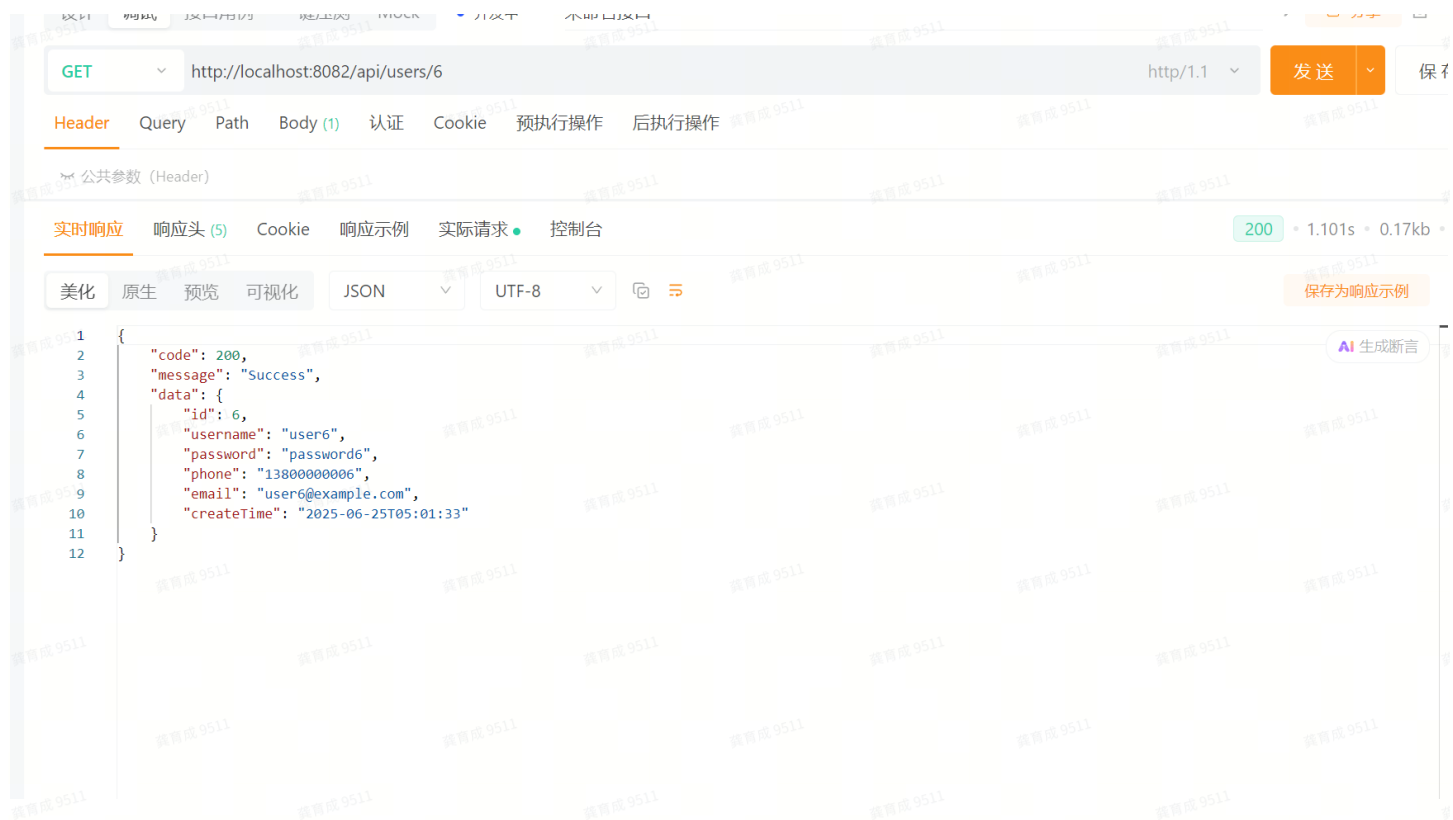
## 5. 编码和测试

- 编码见github: <https://github.com/gongyucheng0505/xm-bms-cloud.git>

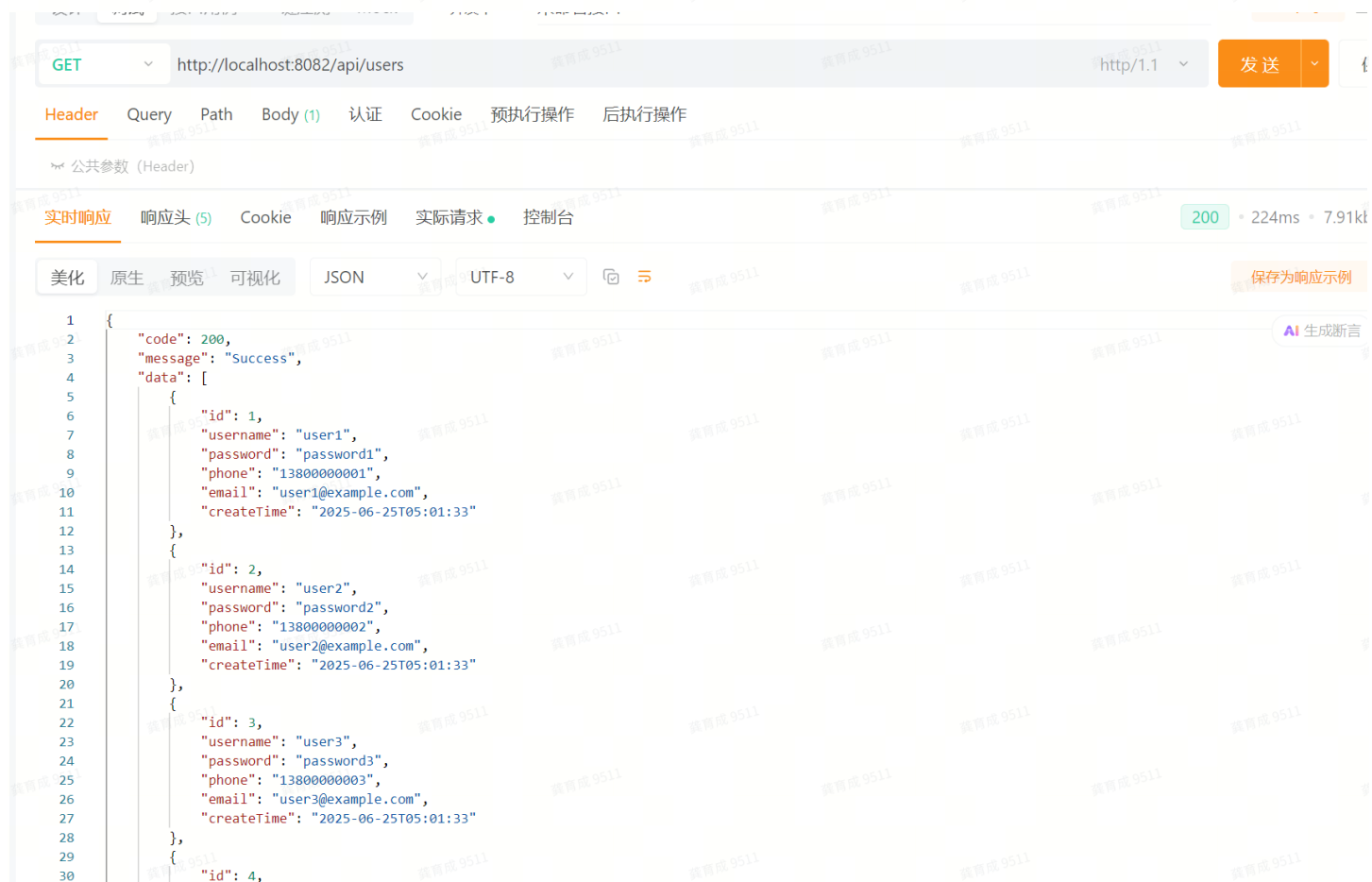
### 5.1 测试截图

#### 5.1.1 用户信息模块

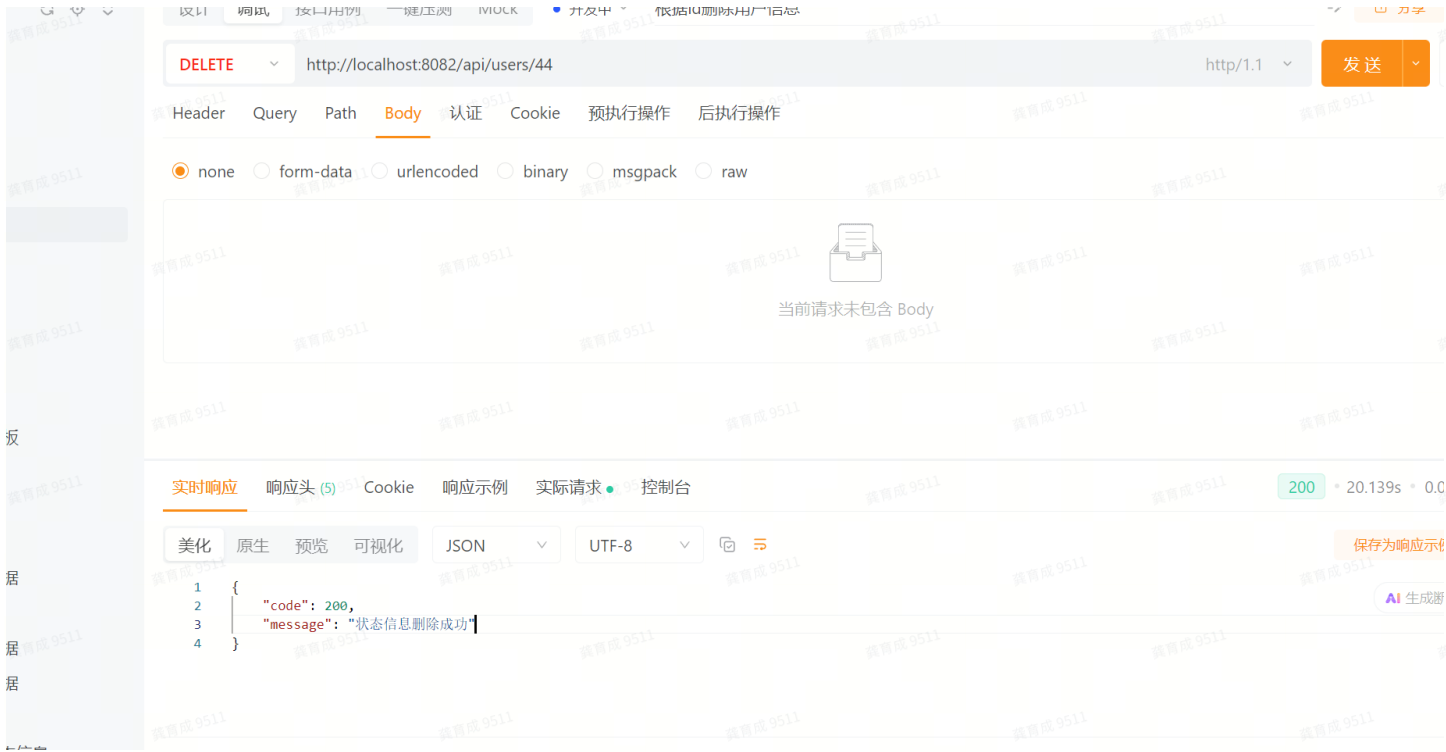
##### 1. 根据id查询用户信息



## 2.查询所有用户信息



## 3.根据id删除用户信息（删除成功不存在就返回记录不存在）



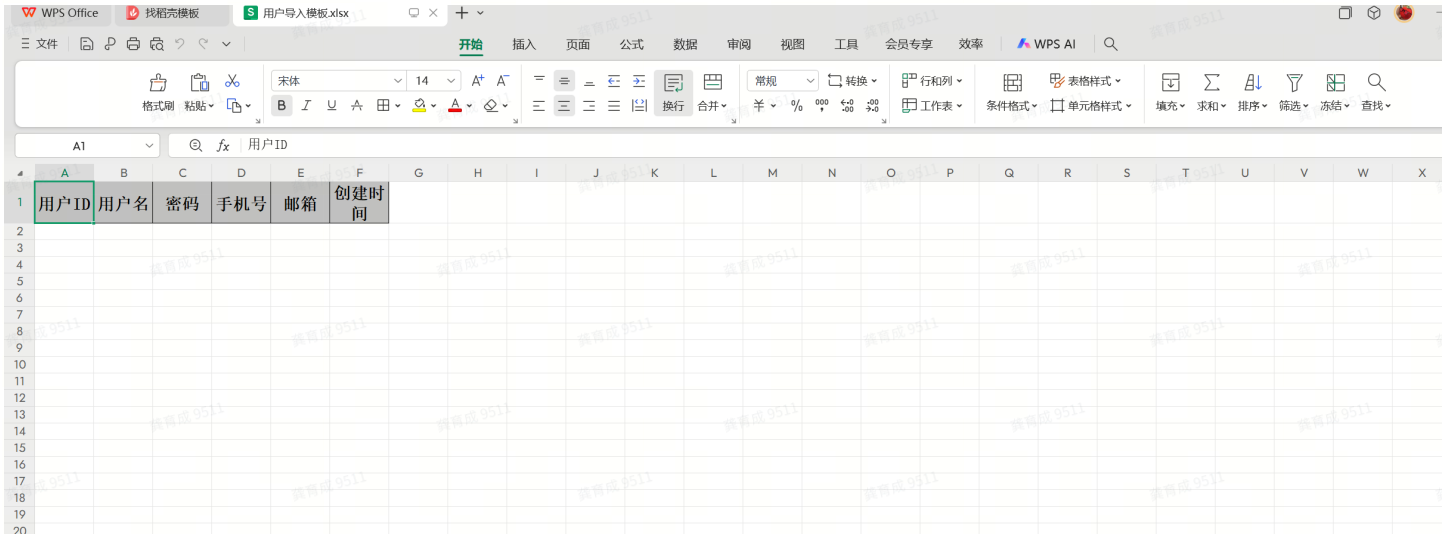
#### 4.根据id修改用户信息



#### 5.单条插入用户信息



## 6. 下载用户信息模板



## 7. 批量导入用户信息



## 8. 导出车辆用户信息



用户ID	用户名	密码	手机号	邮箱	创建时间
3	u32w	password:13800000	user3@ex		2025-06-25 05:01:33
5	user5	password:13800000	user5@ex		2025-06-25 05:01:33
6	user6	password:13800000	user6@ex		2025-06-25 05:01:33
7	user7	password:13800000	user7@ex		2025-06-25 05:01:33
8	user8	password:13800000	user8@ex		2025-06-25 05:01:33
9	user9	password:13800000	user9@ex		2025-06-25 05:01:33
10	user10	password:13800000	user10@e		2025-06-25 05:01:33
11	user11	password:13800000	user11@e		2025-06-25 05:01:33
12	user12	password:13800000	user12@e		2025-06-25 05:01:33
13	user13	password:13800000	user13@e		2025-06-25 05:01:33
14	user14	password:13800000	user14@e		2025-06-25 05:01:33
15	user15	password:13800000	user15@e		2025-06-25 05:01:33
16	user16	password:13800000	user16@e		2025-06-25 05:01:33
17	user17	password:13800000	user17@e		2025-06-25 05:01:33
18	user18	password:13800000	user18@e		2025-06-25 05:01:33
19	user19	password:13800000	user19@e		2025-06-25 05:01:33
20	user20	password:13800000	user20@e		2025-06-25 05:01:33
21	user21	password:13800000	user21@e		2025-06-25 05:01:33
22	user22	password:13800000	user22@e		2025-06-25 05:01:33
23	user23	password:13800000	user23@e		2025-06-25 05:01:33
24	user24	password:13800000	user24@e		2025-06-25 05:01:33
25	user25	password:13800000	user25@e		2025-06-25 05:01:33

## 5.1.2 车辆信息模块

### 1.根据id查询车辆信息

设计 调试 接口用例 一键压测 Mock 开发中 未命名接口

GET http://localhost:8083/api/carInfos/2 http/1.1

Header Query Path Body (1) 认证 Cookie 预执行操作 后执行操作

公共参数 (Header)

参数名	参数值	描述
-----	-----	----

实时响应 响应头 (2) Cookie 响应示例 实际请求 控制台

美化 原生 预览 可视化 JSON UTF-8

```
1 {
2   "code": 200,
3   "message": "Success",
4   "data": {
5     "id": 2,
6     "vid": "2025009283746501",
7     "chassisNo": null,
8     "batteryType": "三元电池",
9     "totalKm": 9800,
10    "healthRate": 91.8,
11    "createTime": "2025-06-25T20:41:00"
12  }
13 }
```

## 2.查询所有车辆信息

设计 调试 接口用例 一键压测 Mock 开发中 未命名接口

GET http://localhost:8083/api/carInfos http/1.1 发送

请求区

实时响应 响应头 (2) Cookie 响应示例 实际请求 控制台 200 • 87ms

美化 原生 预览 可视化 JSON UTF-8 保存为响应

```
1 {
2   "code": 200,
3   "message": "Success",
4   "data": [
5     {
6       "id": 2,
7       "vid": "2025009283746501",
8       "chassisNo": null,
9       "batteryType": "三元电池",
10      "totalKm": 9800,
11      "healthRate": 91.8,
12      "createTime": "2025-06-25T20:41:00"
13    },
14    {
15      "id": 3,
16      "vid": "2025004567893210",
17      "chassisNo": null,
18      "batteryType": "三元电池",
19      "totalKm": 15700,
20      "healthRate": 89.9,
21      "createTime": "2025-06-25T20:42:00"
22    },
23    {
24      "id": 4,
25      "vid": "2025008374650921",
26      "chassisNo": null,
27      "batteryType": "铁锂电池",
28      "totalKm": 7400,
29      "healthRate": 95.6,
30      "createTime": "2025-06-25T20:43:00"
31    },
32    {
33      "id": 5,
```

读取缓存，不存在就查询数据库，再写入redis缓存





### 3.根据id删除车辆信息



首先删除 与车id关联的车辆状态信息。再实现延迟双删，先删除缓存，再更新数据库，等100ms删除缓存（两个缓存都要删掉，防止脏数据）

```
1 @Transactional
2 @Override
3 public int deleteCarInfo(Long id) {
4     // 1. 删除缓存，防止脏数据
5     try (Jedis jedis = jedisClient.getJedis()) {
6         jedis.del("car:info:" + id); // 删除车辆信息缓存
7         jedis.del("car:signal:" + id); // 删除车辆状态信息缓存
8     } catch (Exception e) {
9         e.printStackTrace(); // 打印异常堆栈信息
10    }
11
12    // 2. 延时删除缓存，防止缓存未及时更新
13    try {
14        Thread.sleep(200); // 延时100ms确保缓存删除操作完成
15    } catch (InterruptedException ignored) {
16        ignored.printStackTrace(); // 打印异常
17    }
18
19    // 3. 删除与车辆ID相关的状态信息
20    carSignalMapper.deleteCarSignal(id); // 确保你已在 Mapper 中定义该方法
21}
```

```

22 // 4. 删除车辆信息
23 int result = carInfoMapper.deleteById(id);
24
25 // 5. 再次删除缓存 (如果需要的话)
26 try (Jedis jedis = jedisClient.getJedis()) {
27     jedis.del("car:info:" + id); // 再次删除缓存
28     jedis.del("car:signal:" + id); // 删除车辆信息缓存
29
30 } catch (Exception e) {
31     e.printStackTrace(); // 打印异常堆栈信息
32     // 或者使用日志记录, 比如: logger.error("延迟删除缓存失败", e);
33 }
34
35 return result; // 返回删除结果
36 }

```

#### 4. 根据id修改车辆信息

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://localhost:8083/api/carInfos/3
- Body Type:** raw (selected), JSON (format)
- Request Body (JSON):**

```

1 {
2   "vid": "2025004567893210",
3   "chassisNo": null,
4   "batteryType": "铁锂电池",
5   "totalKm": 15700,
6   "healthRate": 89.9,
7   "createTime": "2025-06-25T20:42:00"

```
- Response:**
  - Status:** 200
  - Message:** "车辆信息修改成功"

先删除缓存, 等100ms, 再更新数据库,

```

1 public void updateCarInfo(CarInfo carInfo) {
2     // 1. 更新数据库
3     carInfoMapper.update(carInfo);
4     // 2. 删除旧缓存
5     try (Jedis jedis = jedisClient.getJedis()) {
6         jedis.del("car:info:" + carInfo.getId()); // 删除缓存

```

```

7         } catch (Exception e) {
8             // 如果删除缓存失败, 记录日志
9         }
10        // 3. 等待缓存清理完成后再更新缓存
11        try {
12            Thread.sleep(100); // 延迟100ms, 确保缓存被清理
13        } catch (InterruptedException e) {
14            e.printStackTrace();
15        }
16        // 4. 将新数据写入缓存
17        try (Jedis jedis = jedisClient.getJedis()) {
18            String key = "car:info:" + carInfo.getId();
19            jedis.setex(key, 1800, objectMapper.writeValueAsString(carInfo)); // 设置30分钟过期时间
20        } catch (Exception e) {
21            // 如果写入缓存失败, 记录日志
22        }
23    }

```

## 5.单条插入车辆信息

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8083/api/carInfos
- Header:** http/1.1
- Body:**

```

1  {
2      "batteryType": "铁锂电池",
3      "totalKm": 25700,
4      "healthRate": 99.9
5  }

```
- Response:**

```

1  {
2      "code": 200,
3      "message": "车辆信息插入成功"
4  }

```

```

1 public void createCarInfo(CarInfo carInfo) {
2     LocalDateTime now = LocalDateTime.now();
3     carInfo.setCreateTime(now);
4     // 生成16位VID: 以"XM2025"开头 + 10位随机数字
5     String vid = "XM2025" + RandomStringUtils.randomNumeric(10);
6     carInfo.setVid(vid);
7
8     carInfoMapper.insert(carInfo);

```



```

1 LocalDateTime now = LocalDateTime.now();
2 for (CarInfo carInfo : carInfoList) {
3     if (carInfo.getCreateTime() == null) {
4         carInfo.setCreateTime(now);
5     }
6     // 生成16位vid, 比如 "XM2025" + 10位数字随机数
7     String vid = "XM2025" + RandomStringUtils.randomNumeric(10);
8     carInfo.setVid(vid);
9 }

```

## 8.导出车辆信息列表

WPS Office

找稻壳模板

车辆信息数据.xlsx

开始

插入

页面

公式

数据

审阅

视图

工具

会员专享

效率

WPS AI

格式刷

粘贴

宋体

14

常规

行和列

表格样式

条件格式

单元格式样

填充

求和

排序

筛选

A1

fx 车辆ID

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	车辆ID	车辆唯一标识VID	车架号	电池类型	总行驶公里数	健康率(%)	创建时间															
2		3	2025004567893210	铁锂电池	15700	89.9	#####															
3		4	2025008374650921	铁锂电池	7400	95.6	#####															
4		5	2025001029384756	铁锂电池	11200	92.1	#####															
5																						
6																						
7																						
8																						
9																						
10																						
11																						
12																						
13																						
14																						
15																						
16																						

## 5.1.3 车辆状态信息模块（改查删都用车id，新增需要判断车id是否存在）

### 1.根据id查询车辆状态信息



```

GET http://localhost:8083/api/carSignals/1

```

Header Query Path Body 认证 Cookie 预执行操作 后执行操作

none form-data urlencoded binary msgpack raw

实时响应 响应头 (2) Cookie 响应示例 实际请求 控制台 200 • 12ms • 0.16kb

美化 原生 预览 可视化 JSON UTF-8 保存为响应示例

```

1 {
2     "code": 200,
3     "message": "Success",
4     "data": {
5         "id": 1,
6         "carId": 6,
7         "signalData": "{\"Mx\": 220.5, \"Mi\": 110.2, \"Ix\": 5.3, \"Ii\": 0.8}",
8         "reportTime": "2025-06-25T14:16:24"
9     }
10 }

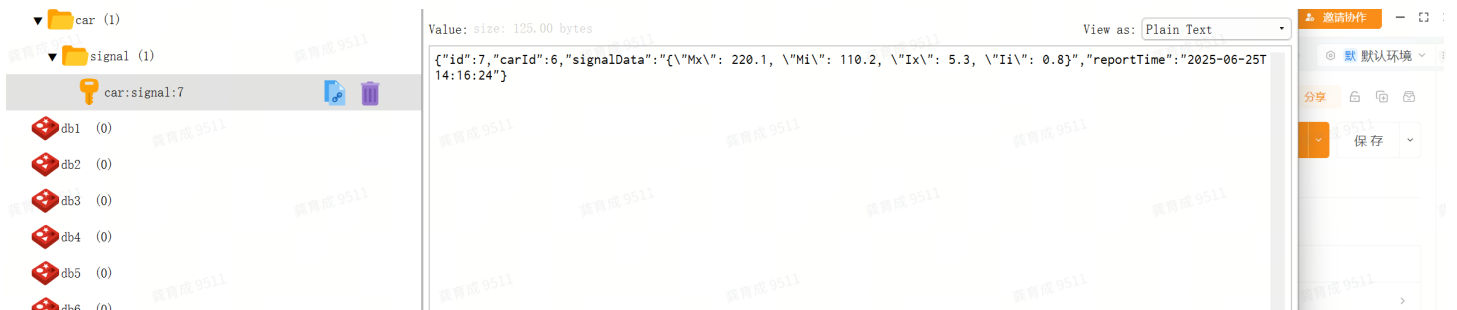
```

读取缓存，不存在就查询数据库，再写入redis缓存

```

1
2 public CarSignal getCarSignalById(Long id) {
3     String key = CACHE_PREFIX id;
4     try (Jedis jedis = jedisClient.getJedis()) {
5         // 1. 读缓存
6         String json = jedis.get(key);
7         if (json != null) {
8             return objectMapper.readValue(json, CarSignal.class);
9         }
10        // 2. 未命中, 查数据库
11        CarSignal carSignal = carSignalMapper.selectById(id);
12        if (carSignal != null) {
13            // 3. 写缓存, 设置30分钟过期
14            jedis.setex(key, 1800, objectMapper.writeValueAsString(carSignal));
15        }
16        return carSignal;
17    } catch (Exception e) {
18        // Redis 出错, 降级查询数据库
19        return carSignalMapper.selectById(id);
20    }
21 }

```



## 2.查询所有车辆状态信息

GET http://localhost:8083/api/carSignals http/1.1 发送 保

Header Query Path Body 认证 Cookie 预执行操作 后执行操作

☒ none ☐ form-data ☐ urlencoded ☐ binary ☐ msgpack ☐ raw

实时响应 响应头 (2) Cookie 响应示例 实际请求 控制台 200 • 75ms • 0.66kb

美化 原生 预览 可视化 JSON UTF-8 保存为响应示例 生成断言

```
1 {
2   "code": 200,
3   "message": "Success",
4   "data": [
5     {
6       "id": 1,
7       "carId": 6,
8       "signalData": "{\"Mx\": 220.5, \"Mi\": 110.2, \"Ix\": 5.3, \"Ii\": 0.8}",
9       "reportTime": "2025-06-25T14:16:24"
10    },
11    {
12      "id": 2,
13      "carId": 8,
14      "signalData": "{\"Mx\": 230.1, \"Mi\": 120.4, \"Ix\": 6.0, \"Ii\": 1.0}",
15      "reportTime": "2025-06-25T14:16:24"
16    },
17    {
18      "id": 3,
19      "reportTime": "2025-06-25T14:16:24"
20    }
21  ]
22 }
```

### 3.根据id删除车辆状态信息

DELETE http://localhost:8083/api/carSignals/1 http/1.1 发送 保

Header Query Path Body 认证 Cookie 预执行操作 后执行操作

☒ none ☐ form-data ☐ urlencoded ☐ binary ☐ msgpack ☐ raw

实时响应 响应头 (2) Cookie 响应示例 实际请求 控制台 200 • 215ms • 0.05kb

美化 原生 预览 可视化 JSON UTF-8 保存为响应示例

```
{ "code": 200, "message": "状态信息删除成功" }
```

延迟双删，先删除缓存，再更新数据库，等100ms删除缓存

```
1 @Override
2 public void deleteCarSignal(Long id) {
3     carSignalMapper.deleteById(id);
4     // 再次删除缓存
5     try (Jedis jedis = jedisClient.getJedis()) {
6         jedis.del(CACHE_PREFIX + id);
7     } catch (Exception e) {
8         e.printStackTrace(); // 打印异常堆栈信息
9     }
10    try {
11        Thread.sleep(100);
12    } catch (InterruptedException ignored) {
```



```

13
14     }
15     // 再次删除缓存
16     try (Jedis jedis = jedisClient.getJedis()) {
17         jedis.del(CACHE_PREFIX + id);
18     } catch (Exception e) {
19         e.printStackTrace(); // 打印异常堆栈信息
20         // 或者使用日志记录, 比如: logger.error("延迟删除缓存失败", e);
21     }
22
23 }

```

#### 4.根据id修改车辆状态信息

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://localhost:8083/api/carSignals/7
- Headers:** Content-Type: application/json
- Body (JSON):**

```

{
  "carId": 6,
  "signalData": "{\"Mx\": 220.0, \"Mi\": 110.2, \"Ix\": 5.3, \"Ii\": 0.8}",
  "reportTime": "2025-06-25T14:16:24"
}

```
- Response:** 200 OK. The response body is:

```

{
  "code": 200,
  "message": "状态信息修改成功"
}

```

先删除缓存，等100ms，再更新数据库，

```

1 @Override
2 public void updateCarSignal(CarSignal carSignal) {
3     // 1. 更新数据库
4     carSignalMapper.update(carSignal);
5     // 2. 删除旧缓存
6     try (Jedis jedis = jedisClient.getJedis()) {
7         jedis.del(CACHE_PREFIX + carSignal.getId()); // 删除缓存
8     } catch (Exception e) {
9         // 如果删除缓存失败, 记录日志
10    }

```

```

11
12 // 3. 等待缓存清理完成后再更新缓存
13 try {
14     Thread.sleep(100); // 延迟100ms, 确保缓存被清理
15 } catch (InterruptedException e) {
16     e.printStackTrace();
17 }
18 // 4. 将新数据写入缓存
19 try (Jedis jedis = jedisClient.getJedis()) {
20     String key = CACHE_PREFIX + carSignal.getId();
21     jedis.setex(key, 1800, objectMapper.writeValueAsString(carSignal));
22 // 设置30分钟过期时间
23 } catch (Exception e) {
24     // 如果写入缓存失败, 记录日志
25 }

```

## 5. 单条插入车辆状态信息

车id存在, 允许插入

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8083/api/carSignals
- Body Type:** raw (JSON)
- Request Body:**

```

1 {
2   "carId": 6,
3   "signalData": "{\"Mx\": 221.6, \"Mi\": 110.1, \"Ix\": 5.1, \"Ii\": 0.8}"
4 }

```
- Response:**

```

1 {
2   "code": 200,
3   "message": "状态信息新增成功"
4 }

```
- Status:** 200 • 388ms • 0.05kb

```

1 @Transactional
2 @Override
3 public int createCarSignal(CarSignal carSignal) {
4     // 查询车id
5     CarInfo carInfo = carInfoMapper.selectById(carSignal.getCarId());
6     // 判断车信息是否存在

```

```

7     if (carInfo != null) {
8         // 设置报告时间
9         LocalDateTime now = LocalDateTime.now();
10        carSignal.setReportTime(now);
11
12        // 插入新的 carSignal 数据
13        carSignalMapper.insert(carSignal);
14        // 返回 1 表示操作成功
15        return 1;
16    } else {
17        // 返回 0 表示车 ID 不存在
18        return 0;
19    }
20 }

```

车id不存在，无法插入车状态信息

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8083/api/carSignals
- Header:** none
- Body:** JSON
 

```

{
  "carId": 116,
  "signalData": "{\\\"Mx\\\": 220.6, \\\"Mi\\\": 110.1, \\\"Ix\\\": 5.1, \\\"Ii\\\": 0.8}"
}
```
- Response:** 200 (Status bar shows 200, 1.734s, 0.04s)
- Response Body:**

```

{
  "code": 404,
  "message": "车id不存在"
}
```

## 5.1.4 预警信息模块

1.生产者使用定时器每六十秒定时扫描所有车辆信息，取出signal\_desc里的Mx Mi，Ix，li。

```

1
2 @Slf4j
3 @Component

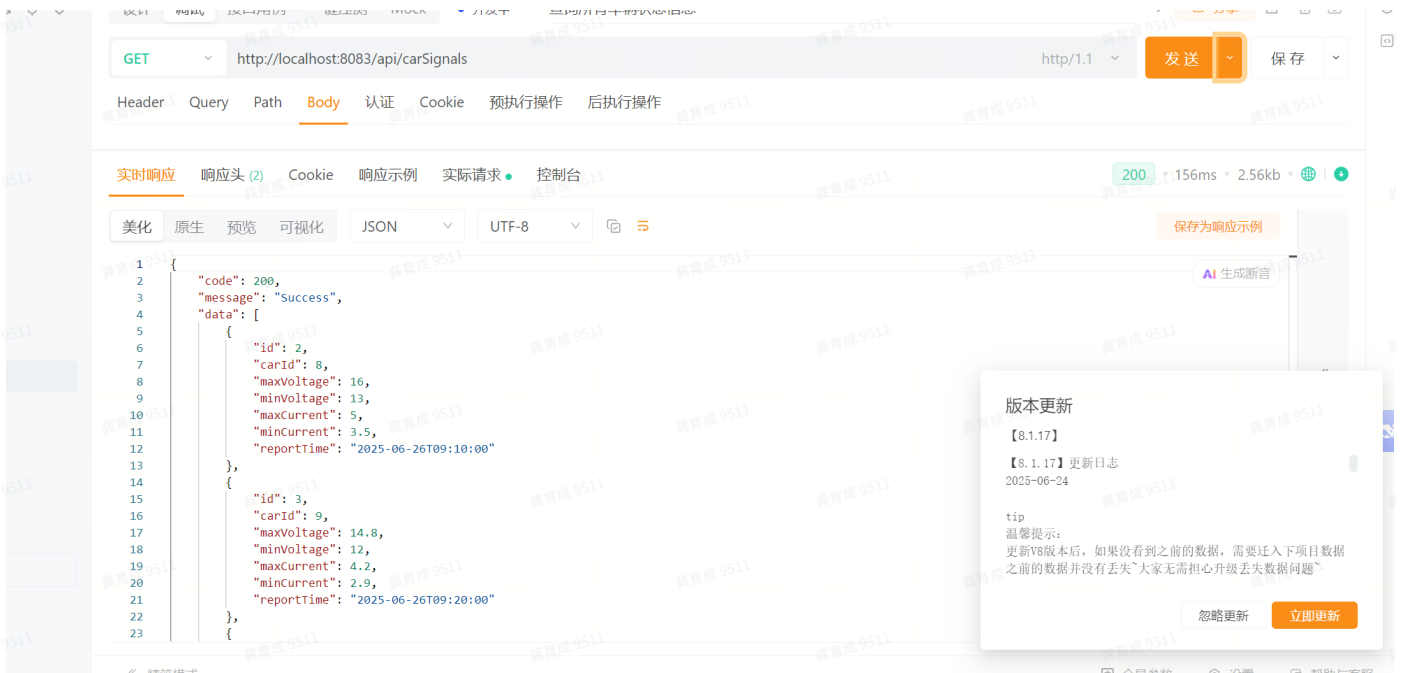
```

```

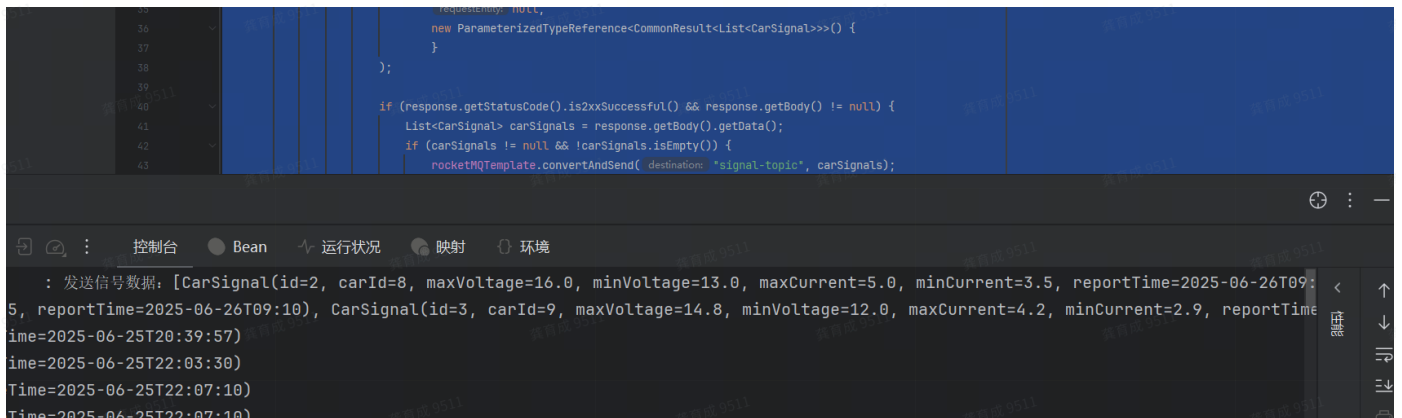
4 public class SignalDataScanTask {
5
6     @Autowired
7     private RocketMQTemplate rocketMQTemplate;
8     @Autowired
9     private RestTemplate restTemplate;
10    @Autowired
11    private CarSignalMapper carSignalMapper;
12
13    @Scheduled(fixedRate = 60000)
14    public void scanAndSendSignalData() {
15        try {
16            ResponseEntity<CommonResult<List<CarSignal>>> response =
17                restTemplate.exchange(
18                    "http://localhost:8083/api/carSignals",
19                    HttpMethod.GET,
20                    null,
21                    new
22                    ParameterizedTypeReference<CommonResult<List<CarSignal>>>() {
23
24                    };
25
26            if (response.getStatusCode().is2xxSuccessful() &&
27                response.getBody() != null) {
28                List<CarSignal> carSignals =
29                    response.getBody().getData();
30
31                if (carSignals != null && !carSignals.isEmpty()) {
32                    rocketMQTemplate.convertAndSend("signal-topic",
33                        carSignals);
34
35                    log.info("发送信号数据: {}", carSignals);
36                } else {
37                    log.warn("未获取到任何车辆信号数据");
38                }
39            } else {
40                log.error("请求 carSignals 接口失败: {}",
41                    response.getStatusCode());
42            }
43        } catch (Exception e) {
44            log.error("扫描并发送信号数据时出错", e);
45        }
46    }
47 }

```

所有车辆信号信息



发送成功



消费者消费信息，生成预警信息，同步到warn\_info表

```

1 package com.xml.bms.warn.MQ;
2
3 import com.fasterxml.jackson.databind.JsonNode;
4 import com.fasterxml.jackson.databind.ObjectMapper;
5 import com.xml.bms.carinfo.domain.CarInfo;
6 import com.xml.bms.carinfo.domain.CarSignal;
7 import com.xml.bms.carinfo.service.CarInfoService;
8 import com.xml.bms.common.web.CommonResult;
9 import com.xml.bms.warn.domain.CarSignalWarning;
10 import com.xml.bms.warn.domain.WarnRequest;
11 import com.xml.bms.warn.domain.WarnResponse;
12 import com.xml.bms.warn.mapper.CarSignalWarningMapper;
13 import com.xml.bms.warn.service.CarSignalWarningReluService;
14 import com.xml.bms.warn.service.CarSignalWarningService;
15 import com.xml.bms.warnrelu.domain.SignalData;

```

```

16 import com.xml.bms.warnrelu.domain.WarningResult;
17 import lombok.extern.slf4j.Slf4j;
18 import org.apache.rocketmq.spring.annotation.RocketMQMessageListener;
19 import org.apache.rocketmq.spring.core.RocketMQListener;
20 import org.checkerframework.checker.units.qual.C;
21 import org.springframework.beans.factory.annotation.Autowired;
22 import org.springframework.http.HttpMethod;
23 import org.springframework.http.ResponseEntity;
24 import org.springframework.stereotype.Component;
25 import org.springframework.web.client.RestTemplate;
26
27 import java.util.ArrayList;
28 import java.util.List;
29 import java.util.Objects;
30 import java.util.stream.Collectors;
31
32 @Slf4j
33 @Component
34 @RocketMQMessageListener(
35     topic = "signal-topic",
36     consumerGroup = "signal-consumer-group",
37     selectorExpression = "*"
38 )
39 public class SignalDataConsumer implements RocketMQListener<List<CarSignal>> {
40     @Autowired
41     private RestTemplate restTemplate;
42
43     @Autowired
44     private CarSignalWarningReluService carSignalWarningReluService;
45     @Autowired
46     private ObjectMapper objectMapper;
47     @Autowired
48     private CarSignalWarningMapper carSignalWarningMapper;
49     @Override
50     public void onMessage(List<CarSignal> carSignals) {
51         System.out.println("接收数据"+carSignals);
52         List<CarSignalWarning> carSignalWarningList = new ArrayList<>();
53
54         for (CarSignal carSignal : carSignals) {
55             Long carId = carSignal.getCarId();
56             CarSignalWarning carSignalWarning = new CarSignalWarning();
57             SignalData signalData = new SignalData();
58             signalData.setMx(carSignal.getMaxVoltage());
59             signalData.setMi(carSignal.getMinVoltage());
60             signalData.setIx(carSignal.getMaxCurrent());
61             signalData.setIi(carSignal.getMinCurrent());
62             String url = "http://localhost:8083/api/carInfos/" + carId;

```

```

63         ResponseEntity<CommonResult> response = restTemplate.exchange(
64             url,
65             HttpMethod.GET,
66             null,
67             CommonResult.class
68         );
69         Object data = Objects.requireNonNull(response.getBody()).getData();
70         CarInfo carInfo = objectMapper.convertValue(data, CarInfo.class);
71         System.out.println(carInfo);
72         if (carInfo == null) {
73             // 处理无对应 carId 的情况 (例如返回一个错误信息)
74             carSignalWarningList.add(null);
75         } else {
76             String batteryType = carInfo.getBatteryType();
77             for (long ruleId = 1; ruleId <= 2; ruleId++) {
78                 if (ruleId == 1) {
79                     WarningResult warningResult =
carSignalWarningReluService.getWarningResult(ruleId, batteryType, signalData);
80                     carSignalWarning.setCarId(carId);
81
carSignalWarning.setWarnLevel(Math.toIntExact(warningResult.getWarnLevel()));
82
carSignalWarning.setWarnName(warningResult.getWarnName());
83                     carSignalWarning.setRuleId(1L);
84                     carSignalWarningList.add(carSignalWarning);
85                 }
86
87                 if (ruleId == 2) {
88                     WarningResult warningResult =
carSignalWarningReluService.getWarningResult(ruleId, batteryType, signalData);
89                     carSignalWarning.setCarId(carId);
90
carSignalWarning.setWarnLevel(Math.toIntExact(warningResult.getWarnLevel()));
91
carSignalWarning.setWarnName(warningResult.getWarnName());
92                     carSignalWarning.setRuleId(2L);
93                     carSignalWarningList.add(carSignalWarning);
94                 }
95             }
96         }
97     }
98     // 1. 过滤 null 值
99     List<CarSignalWarning> validWarnings = carSignalWarningList
100         .stream()
101         .filter(Objects::nonNull)
102         .collect(Collectors.toList());
103     System.out.println("接收到的信号数据: " + carSignalWarningList);

```



```
104         try {
105             int result =
carSignalWarningMapper.batchInsertOrUpdate(validWarnings);
106             log.info("执行成功，影响记录数: {}", result);
107         } catch (Exception e) {
108             log.error("批量插入或更新失败", e);
109         }
110
111     }
112 }
```

2.消费数据成功

```
接收数据[CarSignal(id=2, carId=8, maxVoltage=16.0, minVoltage=13.0, maxCurrent=5.0, minCurrent=3.5, reportTime=2025-06-26T09:10), CarSignal(id=3, carId=9, maxVolt
CarInfo(id=8, vid=XM20252279364902, batteryType=铁锂电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T20:39:57)
CarInfo(id=9, vid=XM20253348763326, batteryType=三元电池, totalKm=25700, healthRate=99.9, createTime=2025-06-25T22:03:30)
CarInfo(id=10, vid=XM20252884666205, batteryType=三元电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T22:07:10)
CarInfo(id=11, vid=XM20252411750017, batteryType=铁锂电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T22:07:10)
CarInfo(id=8, vid=XM20252279364902, batteryType=铁锂电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T20:39:57)
CarInfo(id=8, vid=XM20252279364902, batteryType=铁锂电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T20:39:57)
CarInfo(id=9, vid=XM20253348763326, batteryType=三元电池, totalKm=25700, healthRate=99.9, createTime=2025-06-25T22:03:30)
CarInfo(id=9, vid=XM20253348763326, batteryType=三元电池, totalKm=25700, healthRate=99.9, createTime=2025-06-25T22:03:30)
CarInfo(id=10, vid=XM20252884666205, batteryType=三元电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T22:07:10)
CarInfo(id=10, vid=XM20252884666205, batteryType=三元电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T22:07:10)
CarInfo(id=11, vid=XM20252411750017, batteryType=铁锂电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T22:07:10)
CarInfo(id=11, vid=XM20252411750017, batteryType=铁锂电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T22:07:10)
CarInfo(id=8, vid=XM20252279364902, batteryType=铁锂电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T20:39:57)
CarInfo(id=9, vid=XM20253348763326, batteryType=三元电池, totalKm=25700, healthRate=99.9, createTime=2025-06-25T22:03:30)
CarInfo(id=10, vid=XM20252884666205, batteryType=三元电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T22:07:10)
CarInfo(id=11, vid=XM20252411750017, batteryType=铁锂电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T22:07:10)
CarInfo(id=8, vid=XM20252279364902, batteryType=铁锂电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T20:39:57)
CarInfo(id=9, vid=XM20253348763326, batteryType=三元电池, totalKm=25700, healthRate=99.9, createTime=2025-06-25T22:03:30)
CarInfo(id=10, vid=XM20252884666205, batteryType=三元电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T22:07:10)
CarInfo(id=11, vid=XM20252411750017, batteryType=铁锂电池, totalKm=1000, healthRate=100.0, createTime=2025-06-25T22:07:10)
接收到的信号数据: [CarSignalWarning(id=null, carId=8, ruleId=2, warnLevel=0, warnName=电流差报警 - 铁锂电池, warnTime=null), CarSignalWarning(id=null, carId=8, ruleId=
```

同步到数据库表warn\_info

```
接收到的信号数据: [CarSignalWarning(id=null, carId=8, ruleId=2, warnLevel=0, warnName=电流差报警 - 铁锂电池, warnTime=null), CarSignalWarning(id=null, carId=8, ruleId=
2025-06-26 20:14:22.045 INFO 24044 --- [onsumer-group_2] com.xm.bms.warn.MQ.SignalDataConsumer : 执行成功，影响记录数: 40
```

同步成功

id	car_id	rule_id	warn_level	warn_name	warn_time
54	9	2	1	电流差报警 - 三元	2025-06-26 12:
55	9	2	1	电流差报警 - 三元	2025-06-26 12:
56	9	2	1	电流差报警 - 三元	2025-06-26 12:
57	10	2	1	电流差报警 - 三元	2025-06-26 12:
58	10	2	1	电流差报警 - 三元	2025-06-26 12:
59	10	2	1	电流差报警 - 三元	2025-06-26 12:
60	10	2	1	电流差报警 - 三元	2025-06-26 12:
61	11	2	0	电流差报警 - 铁链	2025-06-26 12:
62	11	2	0	电流差报警 - 铁链	2025-06-26 12:
63	11	2	0	电流差报警 - 铁链	2025-06-26 12:
64	11	2	0	电流差报警 - 铁链	2025-06-26 12:
65	8	2	0	电流差报警 - 铁链	2025-06-26 12:
66	8	2	0	电流差报警 - 铁链	2025-06-26 12:
67	9	2	1	电流差报警 - 三元	2025-06-26 12:
68	9	2	1	电流差报警 - 三元	2025-06-26 12:
69	10	2	1	电流差报警 - 三元	2025-06-26 12:
70	10	2	1	电流差报警 - 三元	2025-06-26 12:
71	11	2	0	电流差报警 - 铁链	2025-06-26 12:
72	11	2	0	电流差报警 - 铁链	2025-06-26 12:
73	8	2	0	电流差报警 - 铁链	2025-06-26 12:
74	8	2	0	电流差报警 - 铁链	2025-06-26 12:
75	9	2	1	电流差报警 - 三元	2025-06-26 12:
76	9	2	1	电流差报警 - 三元	2025-06-26 12:
77	10	2	1	电流差报警 - 三元	2025-06-26 12:
78	10	2	1	电流差报警 - 三元	2025-06-26 12:
79	11	2	0	电流差报警 - 铁链	2025-06-26 12:
80	11	2	0	电流差报警 - 铁链	2025-06-26 12:

3.先远程调用得到所有车辆状态信息。根据规则查询，先在carInfo模块查找是否有车辆信息（RPC），再根据有无规则id生成预警信息，并同步到数据库表warn\_info

无预警id

设计 调试 接口用例 一键压测 Mock 开发中 新增一条车辆状态信息

POST http://localhost:8084/api/carSignalWarnings/warn http/1.1 发送 保存

Header Query Path Body (1) 认证 Cookie 预执行操作 后执行操作

none form-data urlencoded binary msgpack raw JSON

基于数据结构生成

```
1 [
2   {
3     "carId": 8,
4     "signal": "{\\\"Mx\\\":11.0,\\\"Mi\\\":9.6,\\\"Ix\\\":12.0,\\\"Ii\\\":11.7}"
5   }
6 ]
```

字典库 动态值

Q 参数/备注/描述

字典库 如何使用?

实时响应 响应头 (2) Cookie 响应示例 实际请求 控制台 200 • 217ms • 0.16kb

美化 原生 预览 可视化 JSON UTF-8 保存为响应示例

```
1  "warnLevel": 2,
2  "warnName": "电流差报警 - 铁锂电池"
3 },
4 {
5   "carId": 8,
6   "ruleId": 2,
7   "warnLevel": 2,
8   "warnName": "电流差报警 - 铁锂电池"
9 }
10 ]
```

生成断言 断言与校验

## 有预警id

设计 调试 接口用例 一键压测 Mock 开发中 新增一条车辆状态信息

POST http://localhost:8084/api/carSignalWarnings/warn http/1.1 发送 保存

Header Query Path Body (1) 认证 Cookie 预执行操作 后执行操作

none form-data urlencoded binary msgpack raw JSON

基于数据结构生成

```
1 [
2   {
3     "carId": 8,
4     "warnId": 1,
5     "signal": "{\\\"Mx\\\":11.0,\\\"Mi\\\":9.6,\\\"Ix\\\":12.0,\\\"Ii\\\":11.7}"
6   }
7 ]
```

字典库 动态值

Q 参数/备注/描述

字典库 如何使用?

实时响应 响应头 (2) Cookie 响应示例 实际请求 控制台 200 • 19ms • 0.08kb

美化 原生 预览 可视化 JSON UTF-8 保存为响应示例

```
1 [
2   {
3     "carId": 8,
4     "ruleId": 1,
5     "warnLevel": 1,
6     "warnName": "电压差报警 - 铁锂电池"
7   }
8 ]
```

生成断言

```
1 public List<WarnResponse> processWarning(List<WarnRequest> warnData) {
2     List<WarnResponse> carSignalWarningList = new ArrayList<>();
3
4     for (WarnRequest warnRequest : warnData) {
5         Long carId = warnRequest.getCarId();
6         Long warnId = warnRequest.getWarnId();
7         String signal = warnRequest.getSignal();
8         SignalData signalData = new SignalData();
9         WarnResponse carSignalWarning = new WarnResponse();
10        try {
```

```

11         JsonNode rootNode = objectMapper.readTree(signal);
12         // 手动将 JsonNode 中的字段值提取出来并设置到 SignalData 对象
13         signalData.setMx(rootNode.get("Mx").asDouble());
14         signalData.setMi(rootNode.get("Mi").asDouble());
15         signalData.setIx(rootNode.get("Ix").asDouble());
16         signalData.setIi(rootNode.get("Ii").asDouble());
17         System.out.println(signalData);
18     } catch (Exception e) {
19         e.printStackTrace();
20         // 错误处理: 无法解析信号数据
21     }
22
23     String url = "http://localhost:8083/api/carInfos/" + carId;
24     ResponseEntity<CarInfoResponse> response = restTemplate.exchange(
25         url,
26         HttpMethod.GET,
27         null,
28         CarInfoResponse.class // We expect the response to be wrapped
in CarInfoResponse
29     );
30
31     // Get the data (CarInfo object) from the response
32     CarInfo carInfo = response.getBody().getData();
33     System.out.println(carInfo);
34     if (carInfo == null) {
35         // 处理无对应 carId 的情况 (例如返回一个错误信息)
36         carSignalWarningList.add(null);
37     } else {
38         String batteryType = carInfo.getBatteryType();
39         if (warnId != null) {
40             // 根据指定的 warnId 计算报警等级
41             WarningResult warningResult =
carSignalWarningReluService.getWarningResult(warnId, batteryType, signalData);
42             carSignalWarning.setCarId(carId);
43             carSignalWarning.setWarnLevel(warningResult.getWarnLevel());
44             carSignalWarning.setWarnName(warningResult.getWarnName());
45             carSignalWarning.setRuleId(warnId);
46             carSignalWarningList.add(carSignalWarning);
47         } else {
48             for (long ruleId = 1; ruleId <= 2; ruleId++) {
49                 if (ruleId == 1) {
50                     WarningResult warningResult =
carSignalWarningReluService.getWarningResult(ruleId, batteryType, signalData);
51                     carSignalWarning.setCarId(carId);
52                     carSignalWarning.setWarnLevel(warningResult.getWarnLevel());

```

```

53 carSignalWarning.setWarnName(warningResult.getWarnName());
54         carSignalWarning.setRuleId(1L);
55         carSignalWarningList.add(carSignalWarning);
56     }
57
58     if (ruleId == 2) {
59         WarningResult warningResult =
60 carSignalWarningReluService.getWarningResult(ruleId, batteryType, signalData);
61         carSignalWarning.setCarId(carId);
62
63 carSignalWarning.setWarnLevel(warningResult.getWarnLevel());
64 carSignalWarning.setWarnName(warningResult.getWarnName());
65         carSignalWarning.setRuleId(2L);
66         carSignalWarningList.add(carSignalWarning);
67     }
68 }
69
70 }
71 }
72 }
73 return carSignalWarningList;
74 }

```

4.规则逻辑可先根据电池类型和reluNo去数据查找对应描述，返回一个list用 case进行判断。

```

1  @Override
2  public WarningResult getWarningResult(Long reluNo, String batteryType,
3      SignalData signalData) {
4      List<BatteryWarningRule> rules =
5      judgeByRulesMapper.selectByRuleNoAndBatteryType(reluNo, batteryType);
6      if (rules == null || rules.isEmpty()) {
7          return new WarningResult(-1L, "无规则");
8      } else {
9          case1:
10         case2:
11         case3:
12         .....
13     }
14 }

```

## 6. 维护

## 7. 思考

- 系统每天处理信号量为百万甚至千万数据级别：考虑数据量对系统性能的影响，给出合理设计数据存储和查询方案。
- 定时扫描 上传车辆状态信息，同步到数据库，根据车id和ruleId唯一标识车辆预警信息，设为key存入Redis。用户主动去查询返回缓存，缓存失效，数据库再去兜底。（车辆预警信息应该是更新的很快的）