

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | B/S体系软件设计 |
| 姓 名： | 周炜 |
| 学 院： | 计算机科学与技术学院 |
| 系： | 计算机科学与技术系 |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3210103790 |
| 指导教师： | 胡晓军 |

2023年 10月 10日

**浙江大学实验报告**

课程名称： B/S体系软件设计 实验类型： 课程大作业

实验项目名称： 物联网设备管理平台

学生姓名： 周炜 专业： 3210103790 学号： 3210103790

同组学生姓名： 指导老师： 胡晓军

实验地点： 实验日期： 年 月 日

目录

[1 项目简介 2](#_Toc153454782)

[2 功能介绍 3](#_Toc153454783)

[2.1 功能模块设置 3](#_Toc153454784)

[2.2 详细功能 3](#_Toc153454785)

[3 环境配置与基本准备 4](#_Toc153454786)

[3.1 环境配置 4](#_Toc153454787)

[3.2 发送消息 4](#_Toc153454788)

[4 操作指南 6](#_Toc153454789)

[5 函数接口 18](#_Toc153454790)

[6 Q&A 常见问题 26](#_Toc153454791)

# 项目简介

+ 物联网（ IoT ，Internet of things ）即“万物相连的互联网”，是互联网基础上的延伸和扩展的网络，将各种信息传感设备与网络结合起来而形成的一个巨大网络，实现在任何时间、任何地点，人、机、物的互联互通。

+ 本网站作为一个物联网应用网站，基于B/S结构设计，通过实现物联网的基本功能，实现指定物联网终端数据的接收与统计分析。

eIoT System是⼀个基于B/S体系结构的物联⽹应⽤管理系统，提供⽤户友好的⽹页界⾯。⽤户可通过注册和登

录进⼊系统，添加、配置、删除物联⽹终端设备，查看统计数据。

eIoT System的前端编写借助Umi框架，数据流⽅案使⽤DvaJS，UI组件库使⽤antd及antd-pro。

后端⽤node.js语⾔编写，数据库使⽤⾮关系型数据库MongoDB。为了简化MQTT数据包的处理，还使⽤了

Mosquito服务器作为消息中间件。

您可以访问eIoT System的GitHub仓库查看源代码。

# 功能介绍

## 功能模块设置

###### 2.1.1 未注册的游客

+ 用户登录模块

+ 首页展示模块

###### 2.1.2 用户

+ 用户管理模块

+ 统计展示模块

+ 设备管理模块

+ 信息展示模块

## 详细功能

+ 能够对mqttServer发布的信息进行监听和保存

+ 提供用户管理功能，能注册登录进入系统，且进行一定的加密与验证处理

+ 用户可以查看并编辑、新建设备信息

+ 用户可以查看某个设备上报的数据信息

+ 用户可以对数据信息进行筛选，并查看设备的状态和历史轨迹

+ 用户可以通过图表形式直观查看设备、信息的统计情况

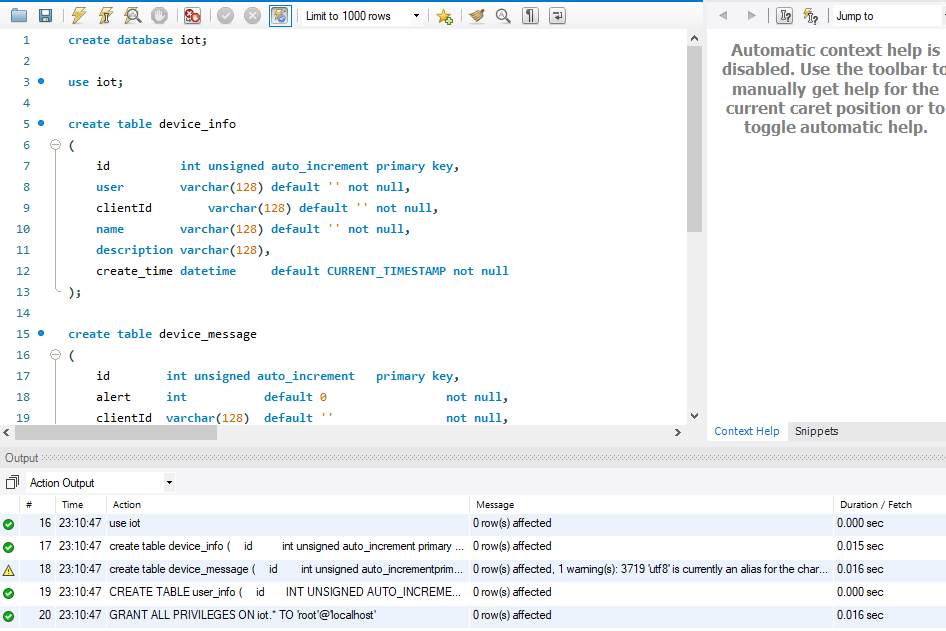
# 环境配置与基本准备

## 环境配置

首先参见**[开发文档]**中有关的环境配置，完成所有必要环境的配置，包括但不限于，前端的React, nvm, nodejs, 后端的python, mqtt, 服务器的java, maven,数据库的mysql等等

## 发送消息

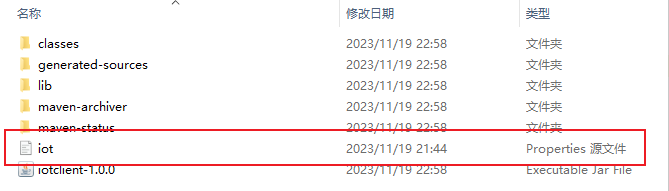
在成功使用MySQL Workbench建表**(注意需要在root下进行建表)**之后



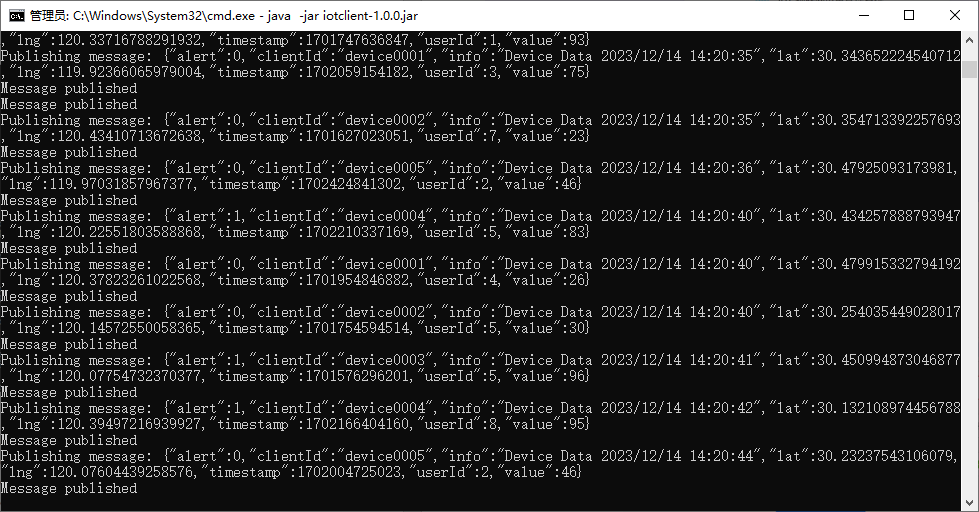
我们可以使用老师提供的java程序来进行物联网基础设备的信息发送， 可以使用cmd和下列命令完成编译：

|  |
| --- |
| # 首先要运行后端程序  mvn clean package  # 同时复制一份iot到target  cd target  java -jar iotclient-1.0.0.jar |

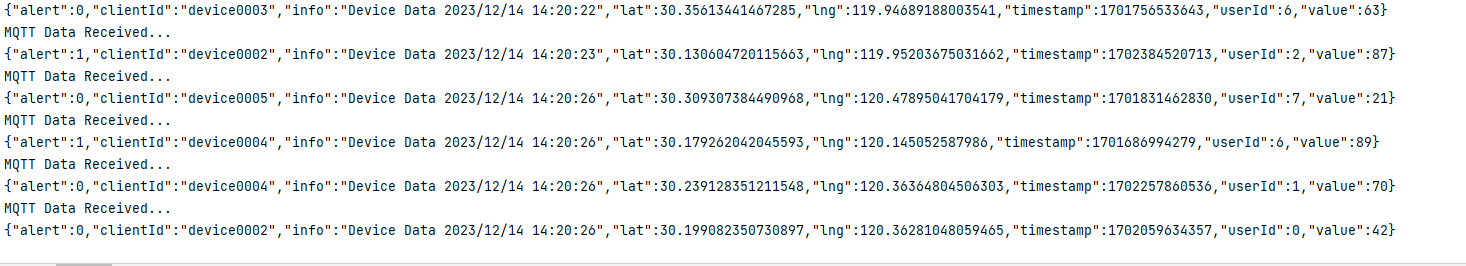
并且注意iot需要拖动一份到target



运行后，cmd的显示如下:



此时后端也相应得做出了反应

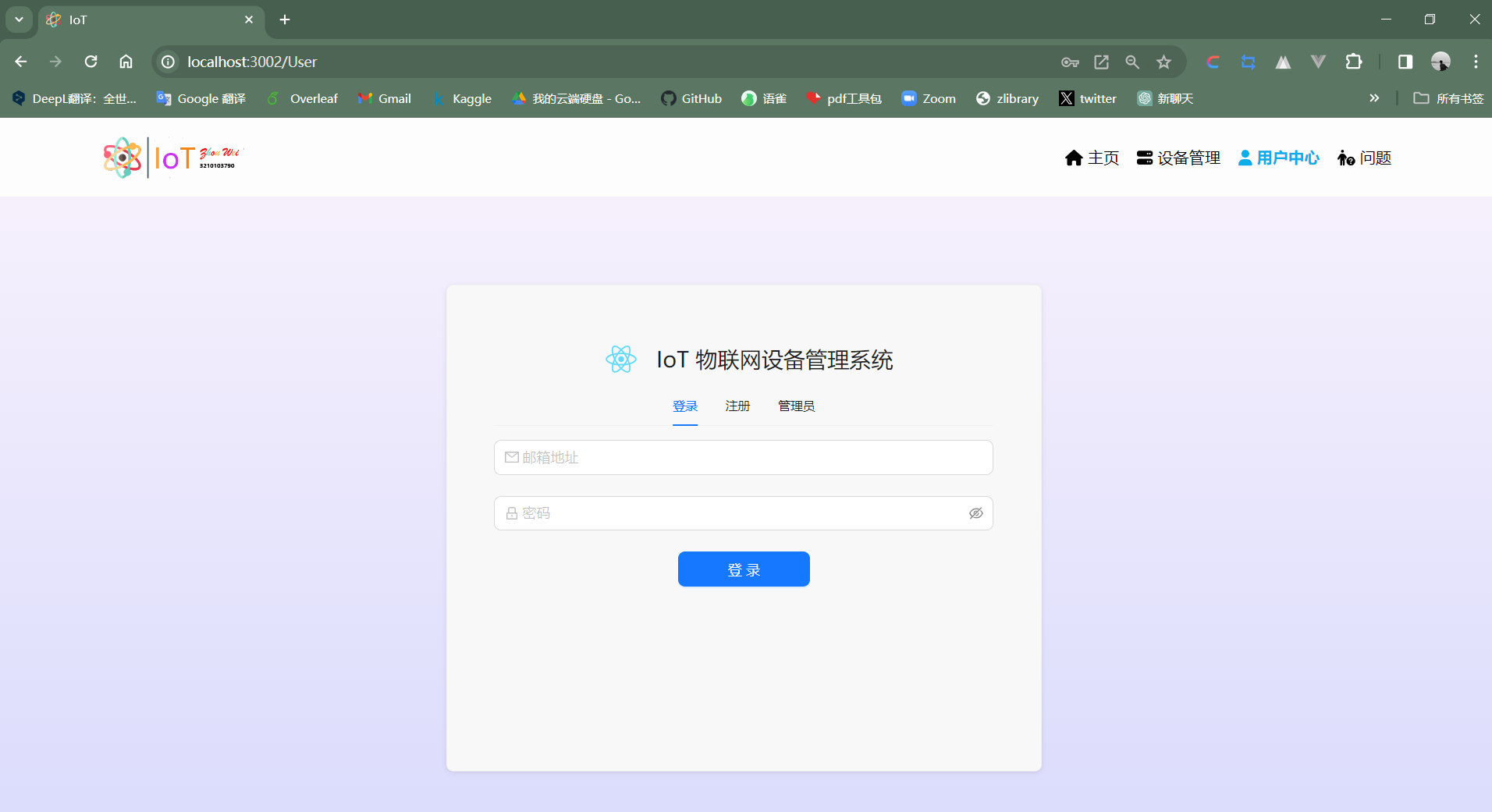


这个时候就已经建立了物联网设备的消息了，之后我们就可以来体验本网站的功能了

# 操作指南

## 用户注册与登录

进入主页后，点击用户中心或者设备管理都会跳出下列界面，即需要先登录才能进行设备管理以及用户信息修改



首先我们可以创建一个账号，首先邮箱格式需要正确



密码需要满足一定格式



两次密码需要相同



注意name和邮箱均为unique，重复会导致注册失败

|  |  |
| --- | --- |
| 邮箱存在 | 用户名存在 |
|  |  |

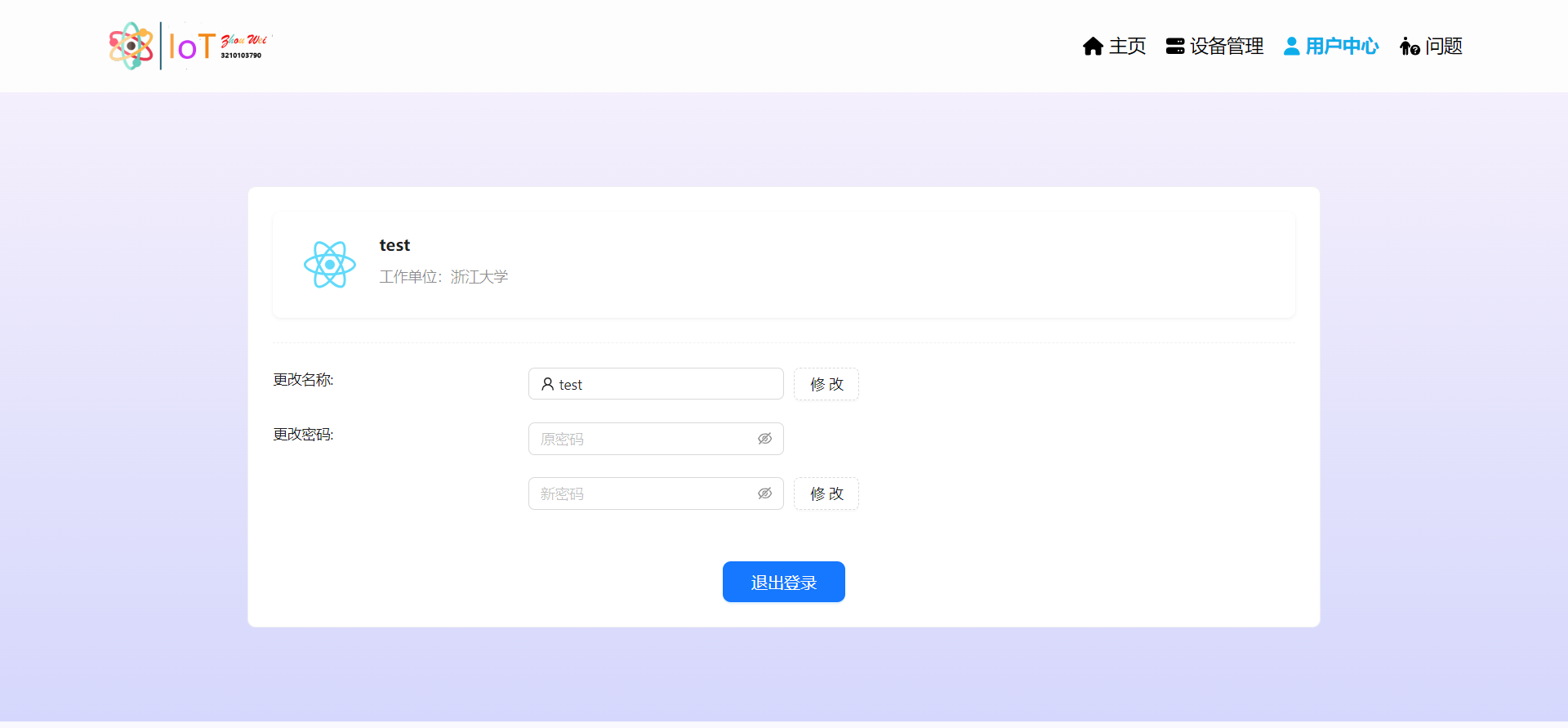
注册成功后，我们用刚刚注册成功的账号进行登录



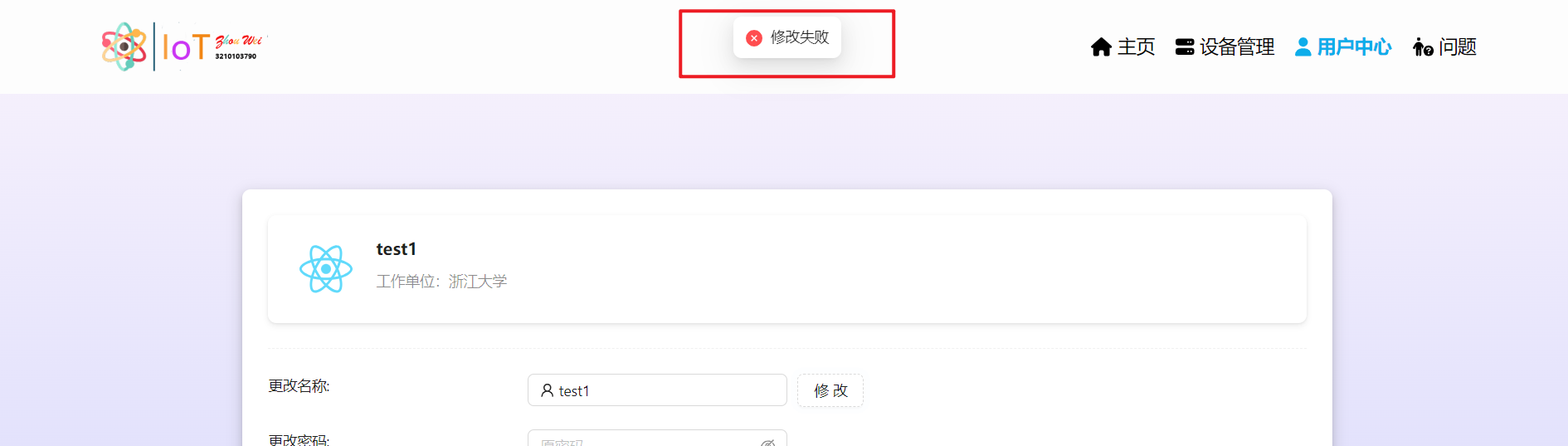
并且注意，登录信息会存在60分钟，所以刷新网页之后，登录的账号不会自动退出。但是时间超过60分钟，则需要重新登录

## 个人信息管理

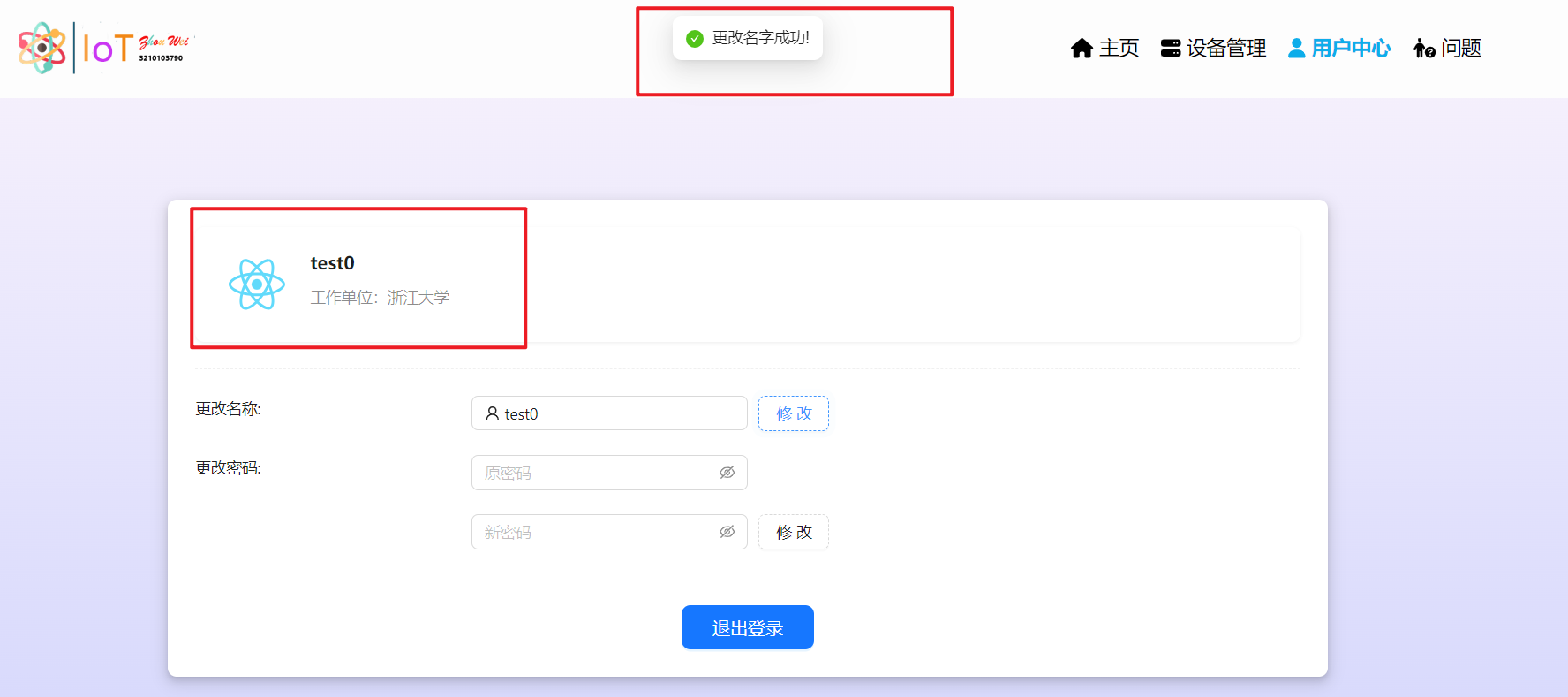
登录成功之后的用户中心界面如下



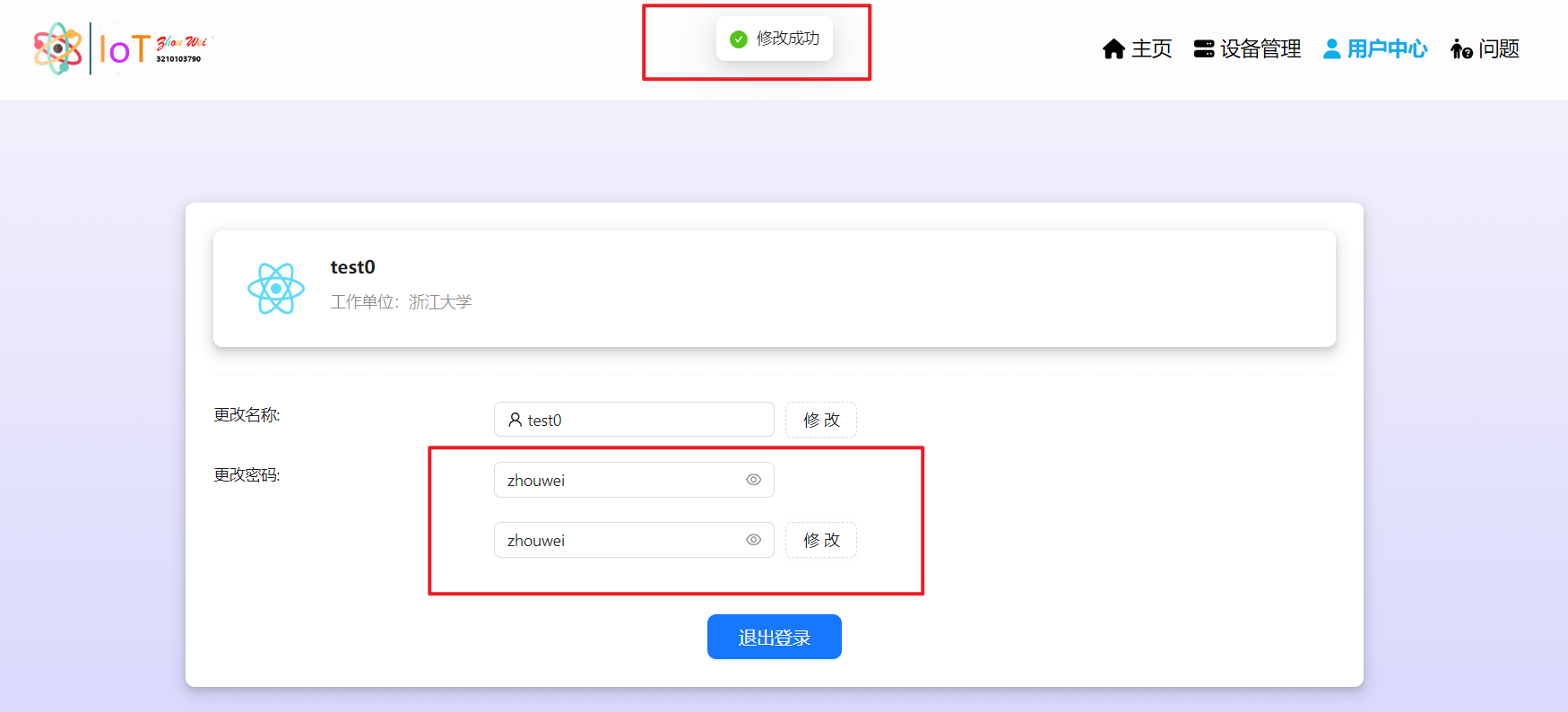
可以通过昵称修改修改姓名，不过姓名具有唯一性，所以如果已经被占用了就无法更改。



更改成功后弹出正确的消息，并且logo旁边显示的名称也相应得改变了

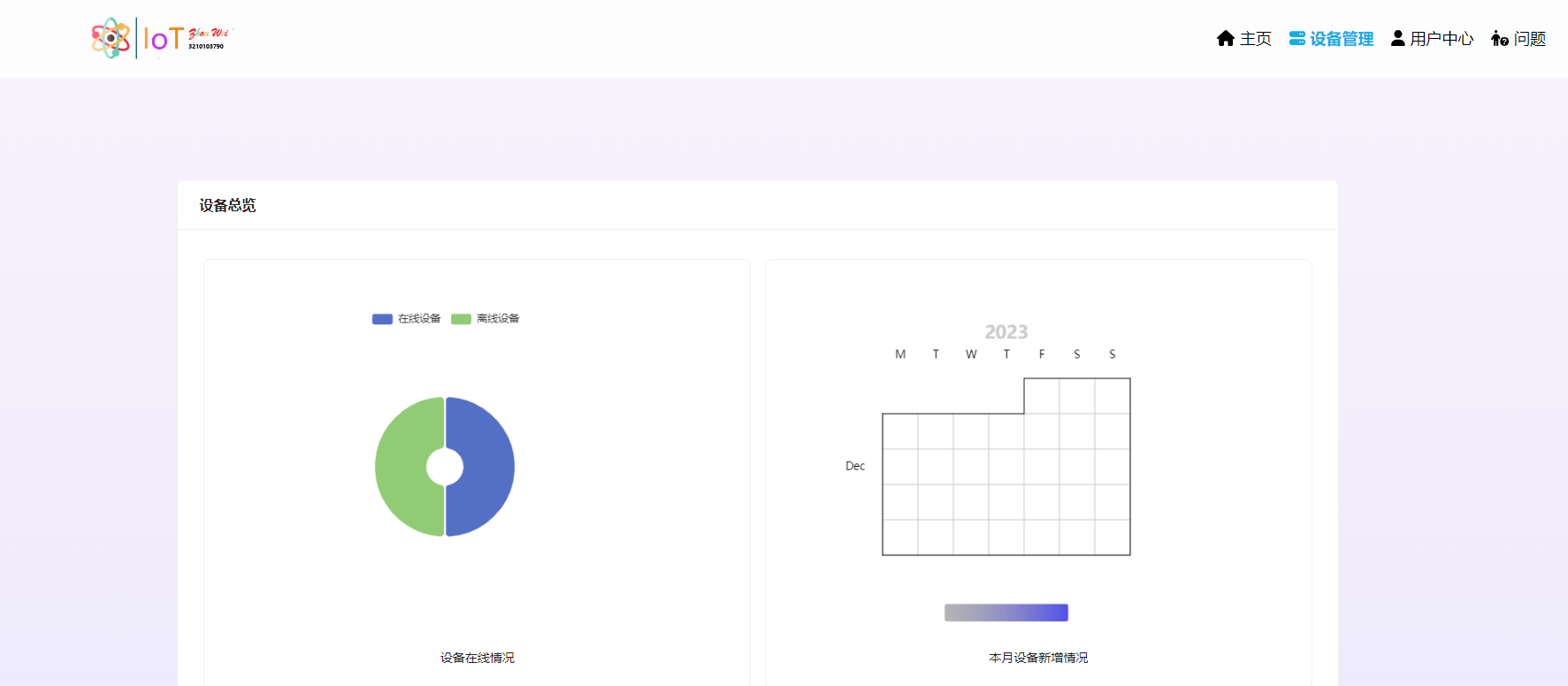


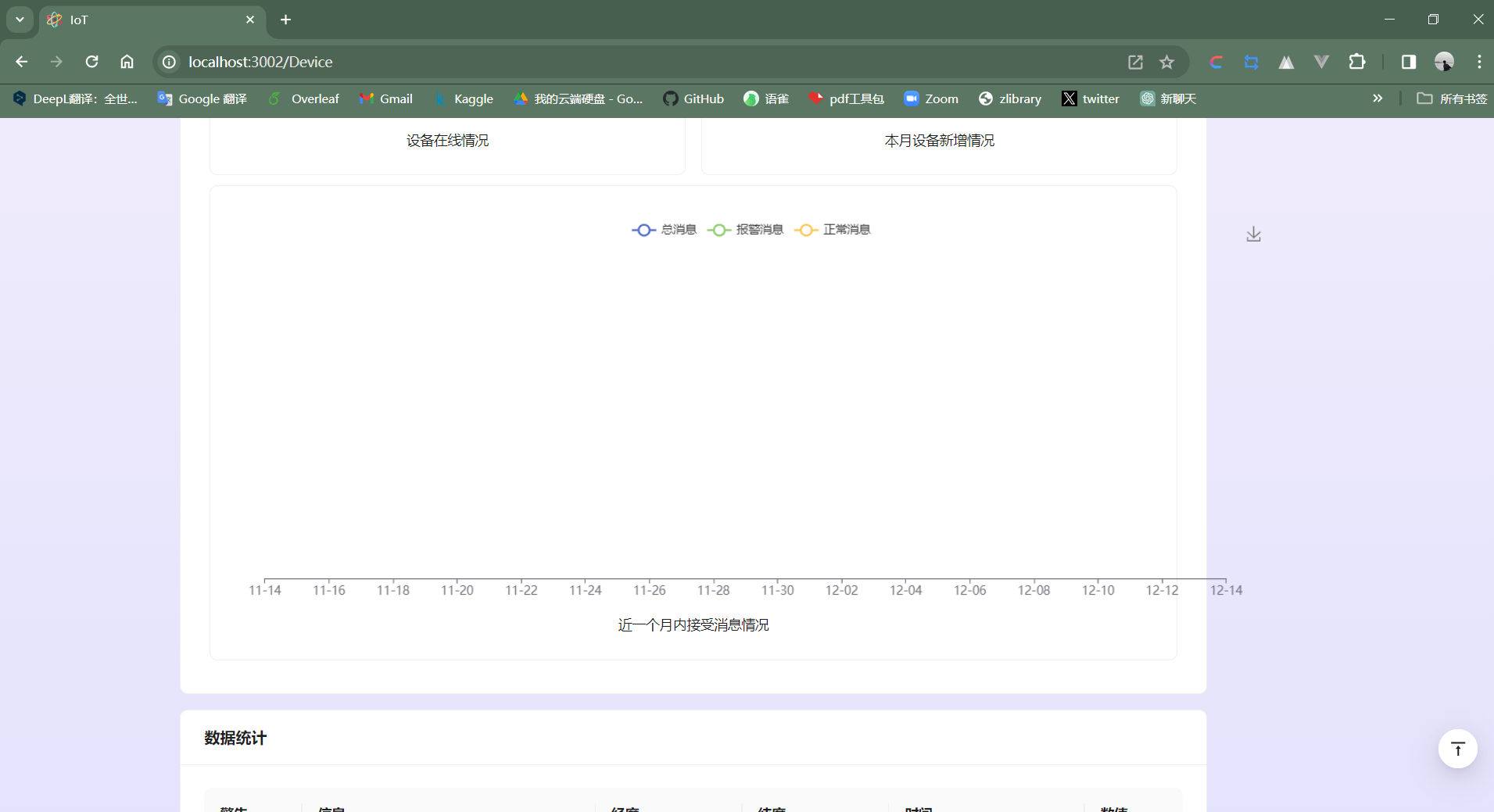
并且用户还能够修改自己的密码:

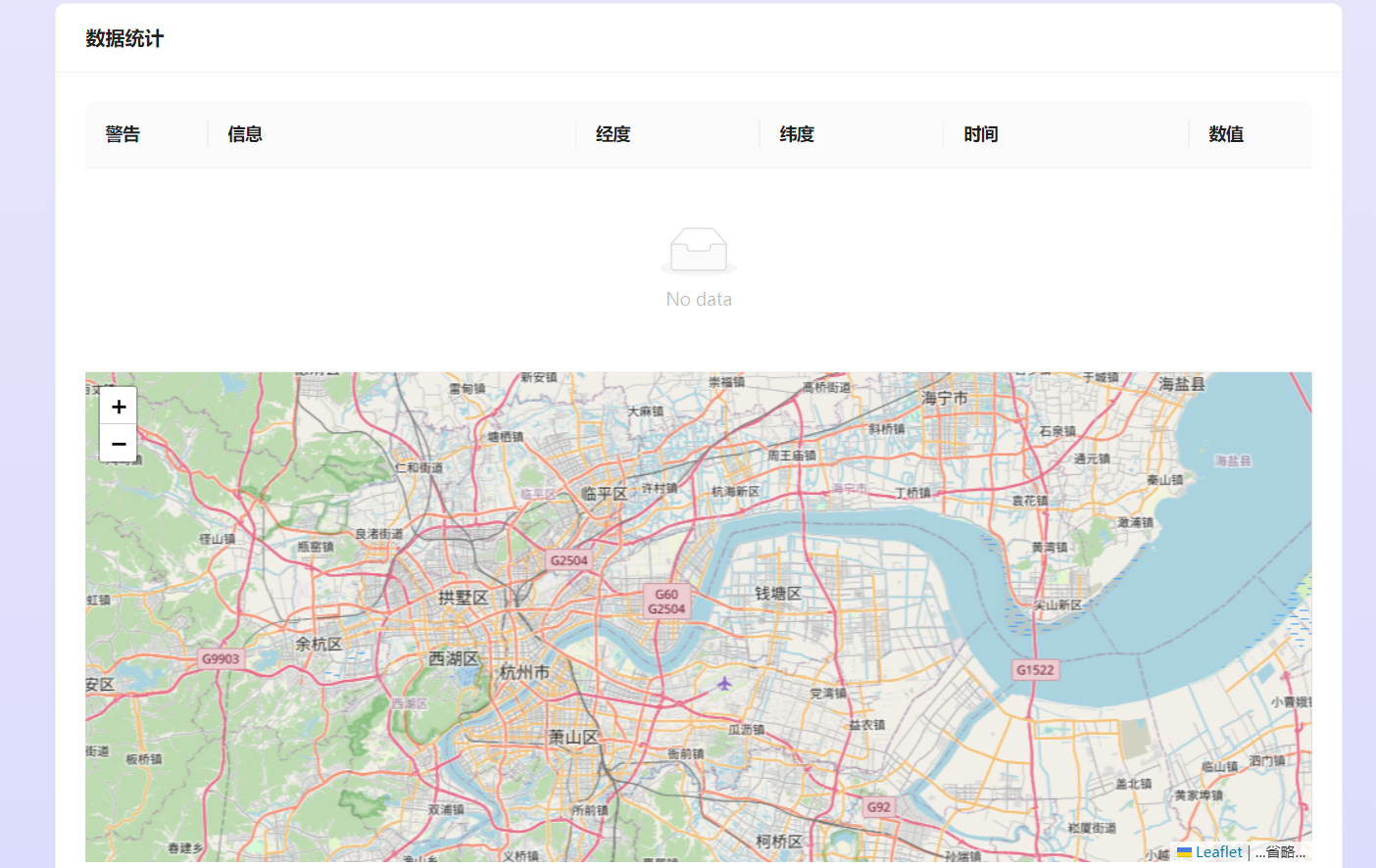


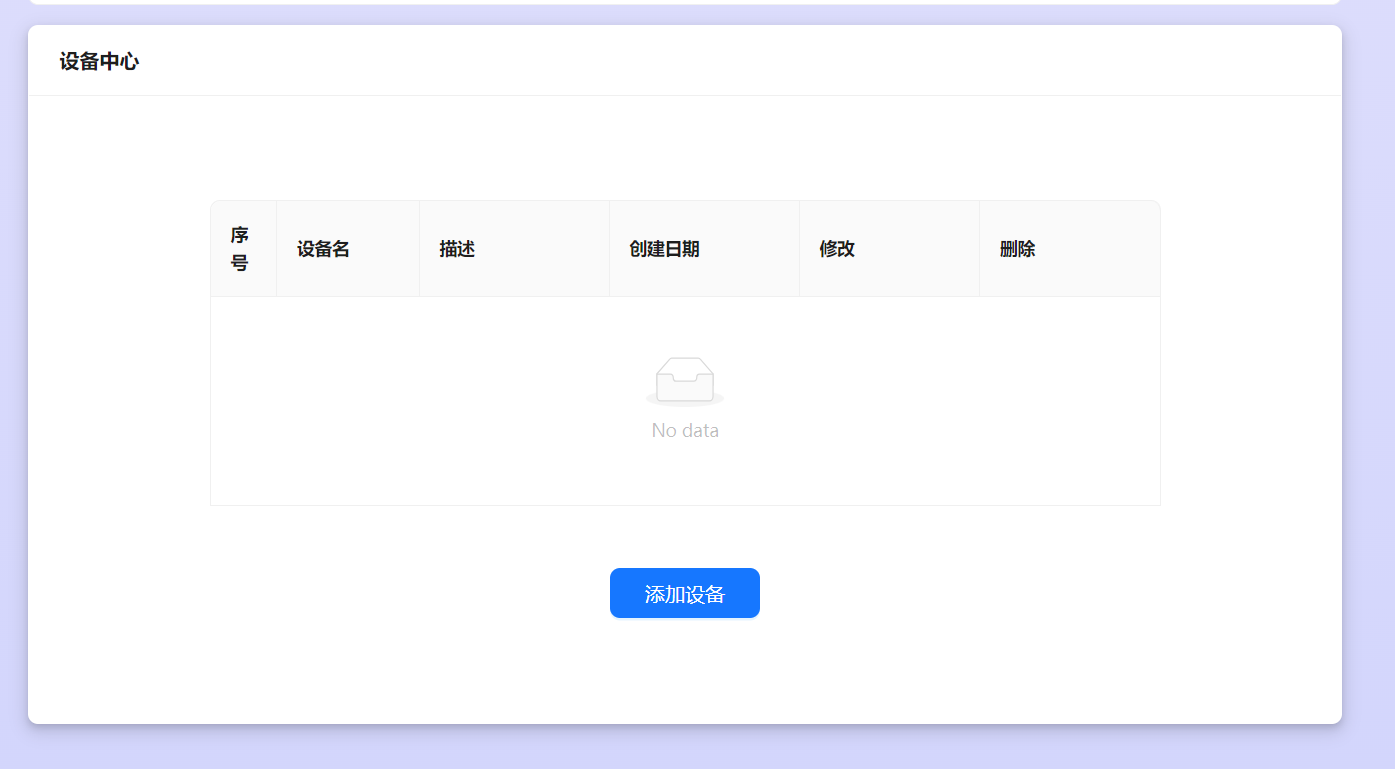
## 设备管理

可以发现刚登录成功后设备数量和消息均为0

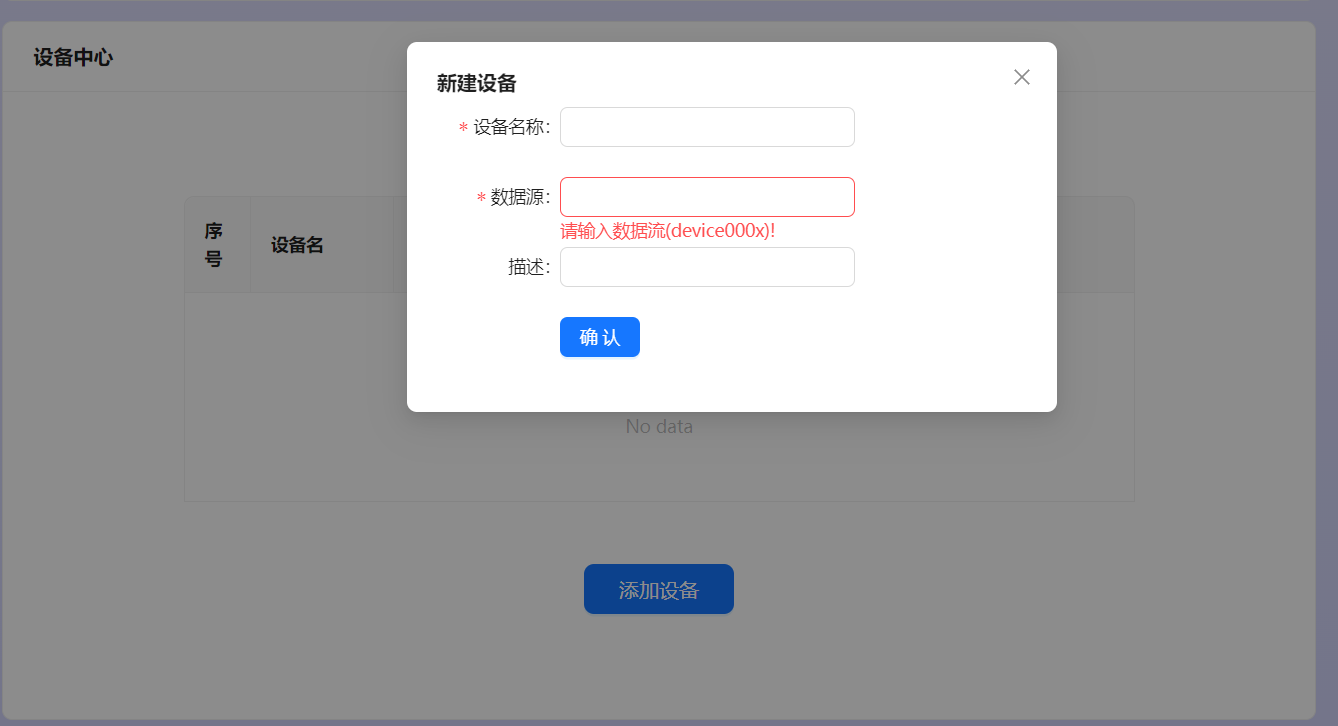




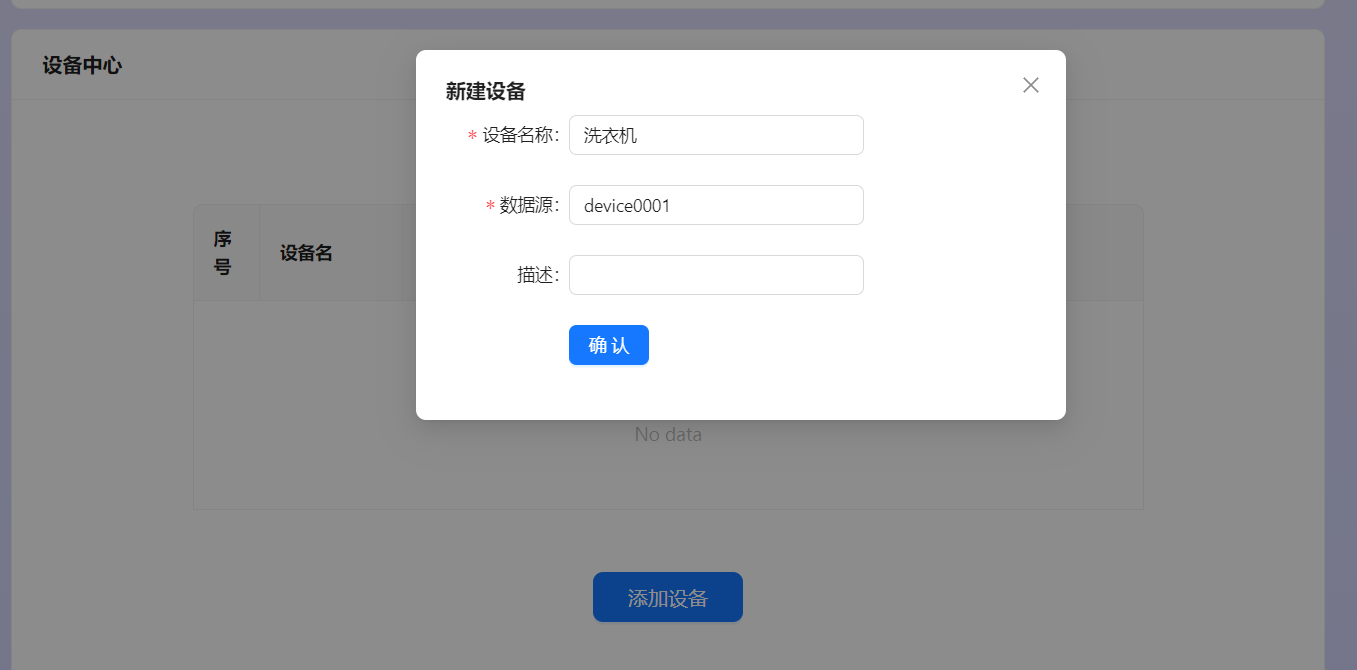




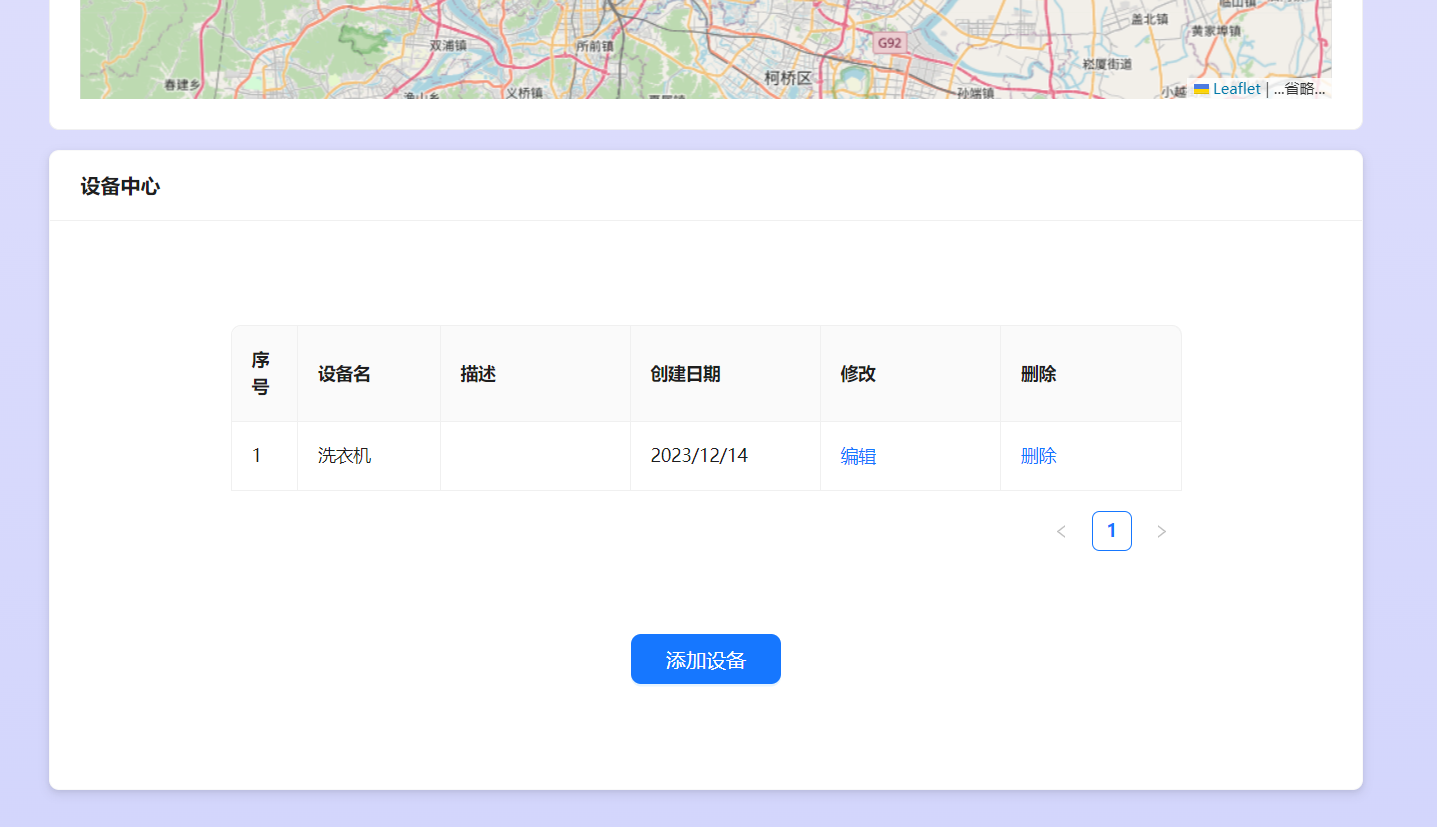
现在我们来新增一个设备,其中数据源一栏是老师提供的java程序中的clientID的值，这里输入的时候我就给出了提示



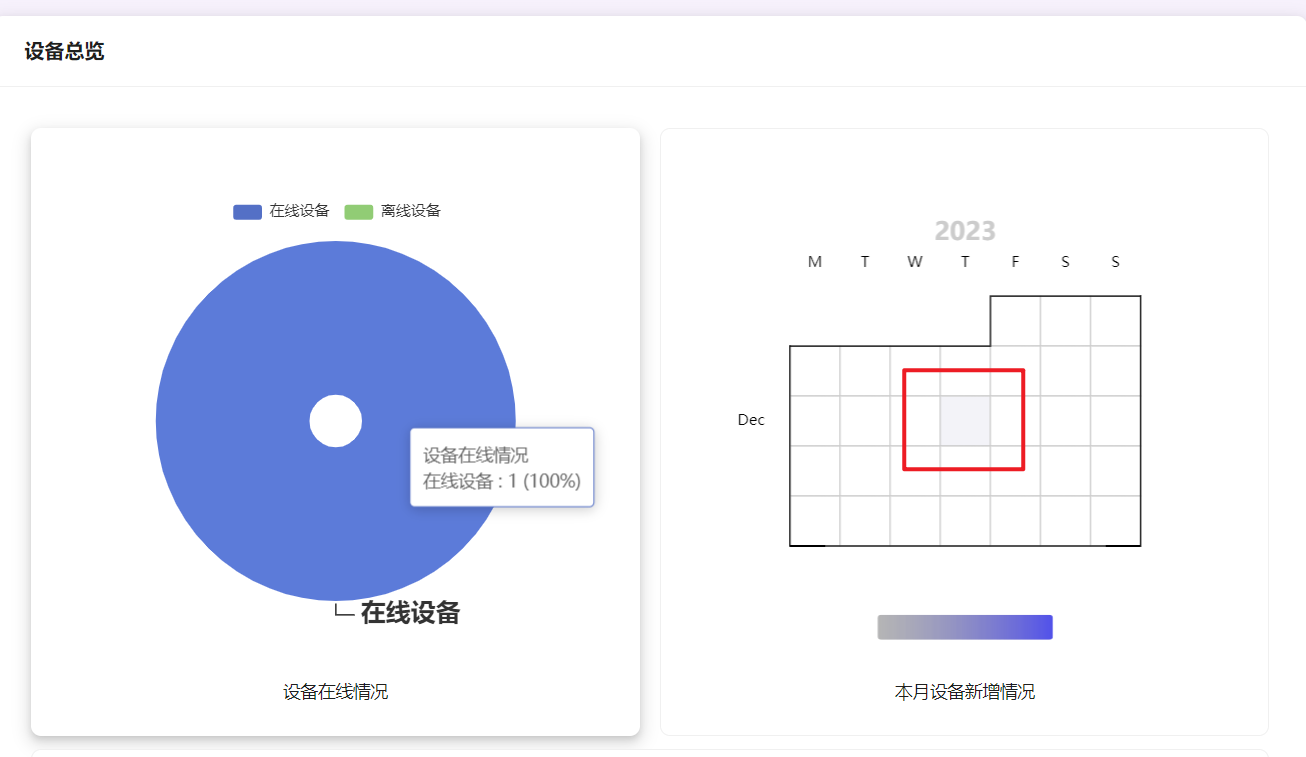
描述可以为空



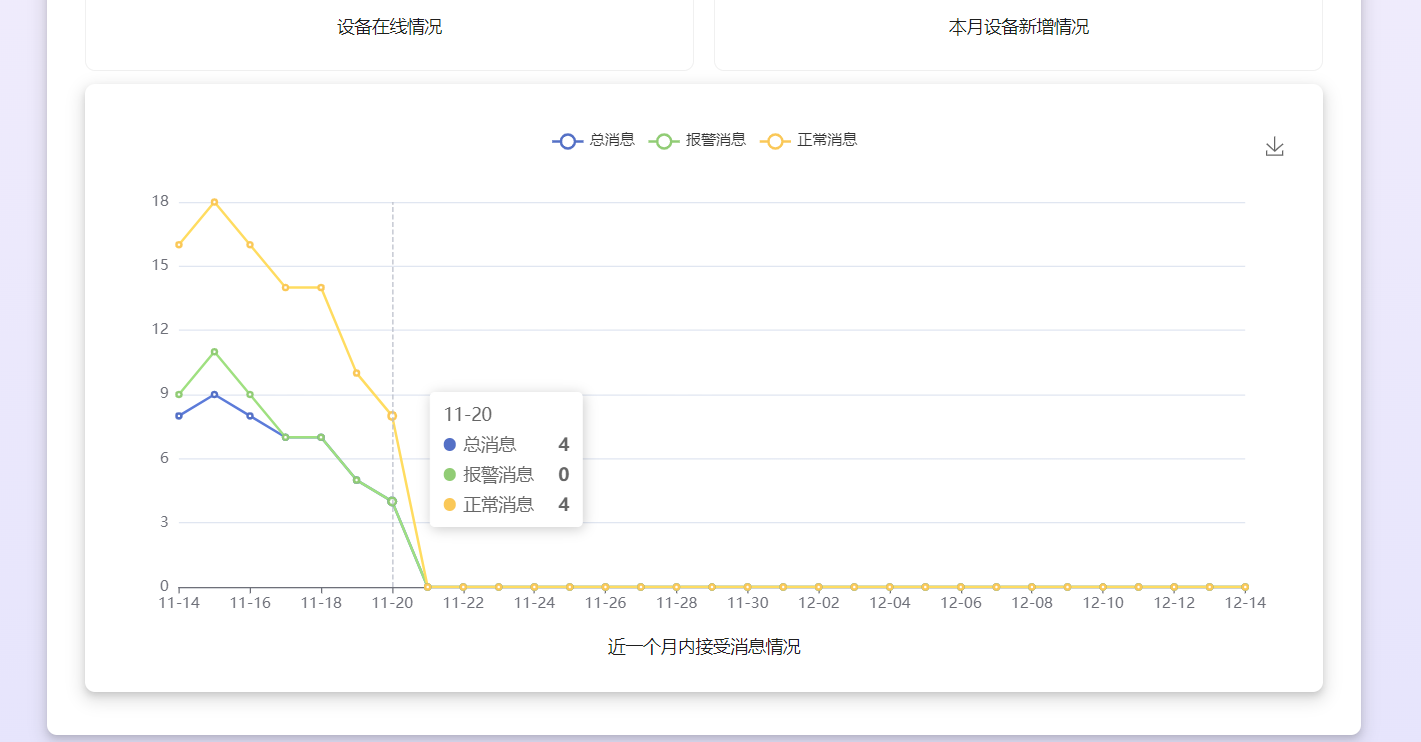
点击确认后，可以发现设备已经显示在列表中了



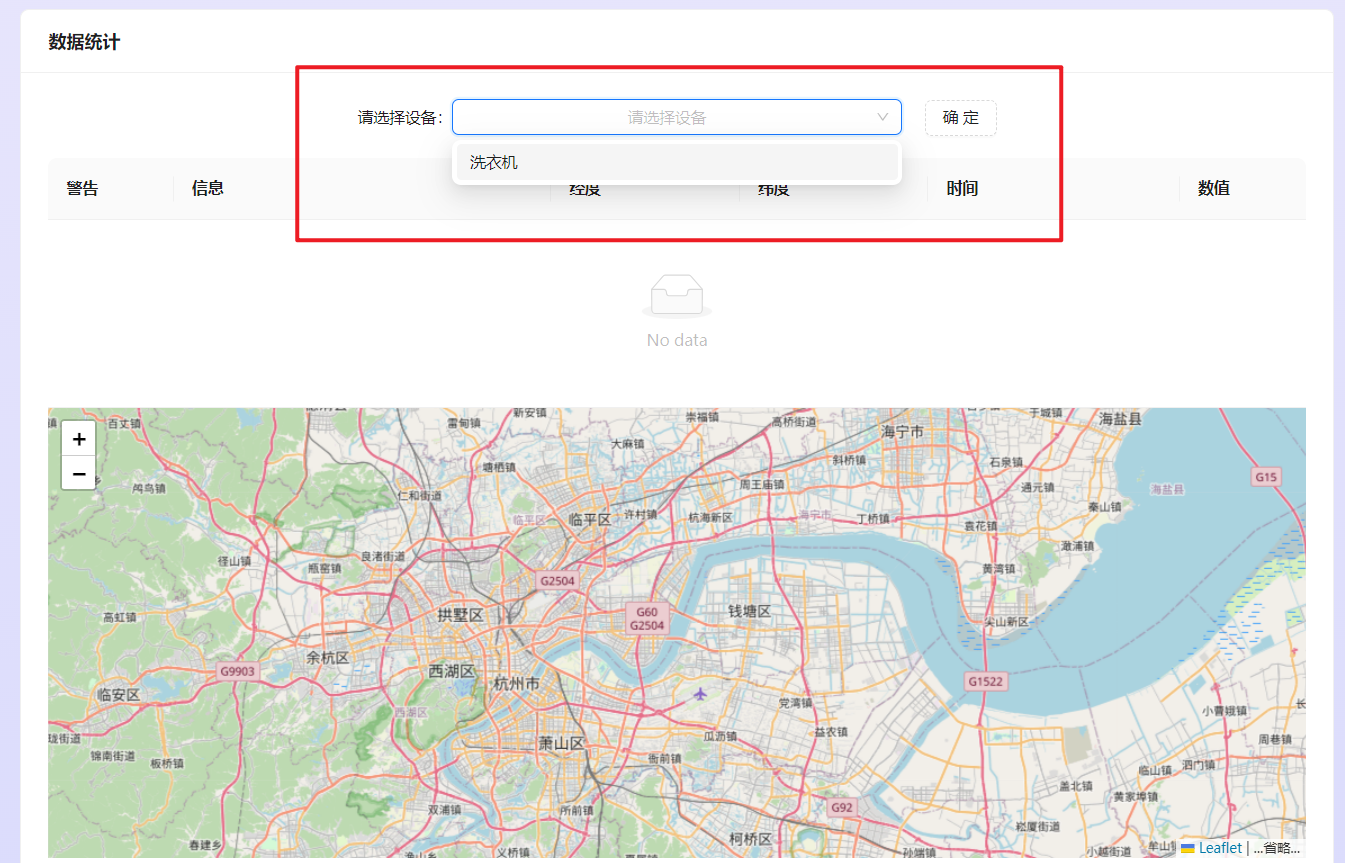
并且可以发现设备总览中的饼图也已经刷新了，日历也刷新了



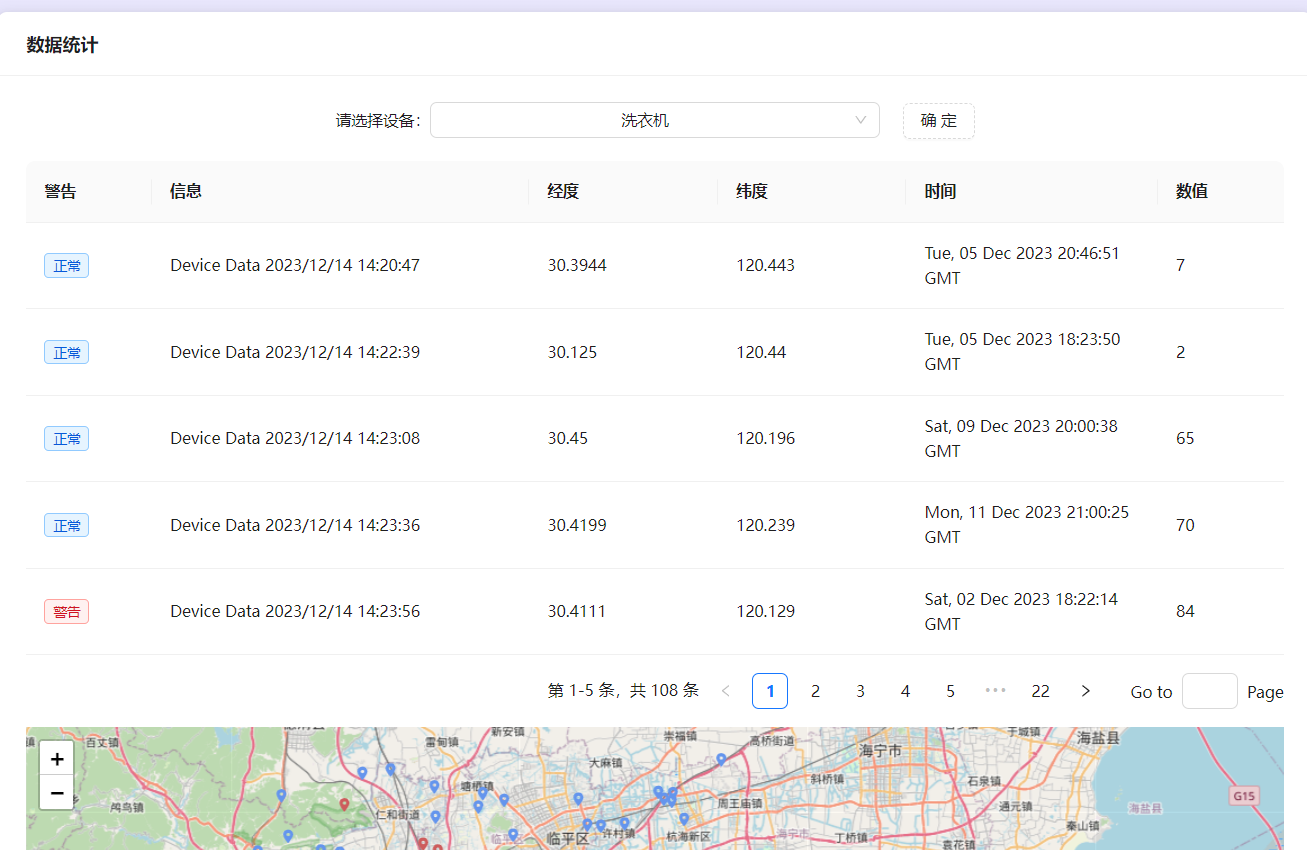
还可以看到该设备的本月消息的数量



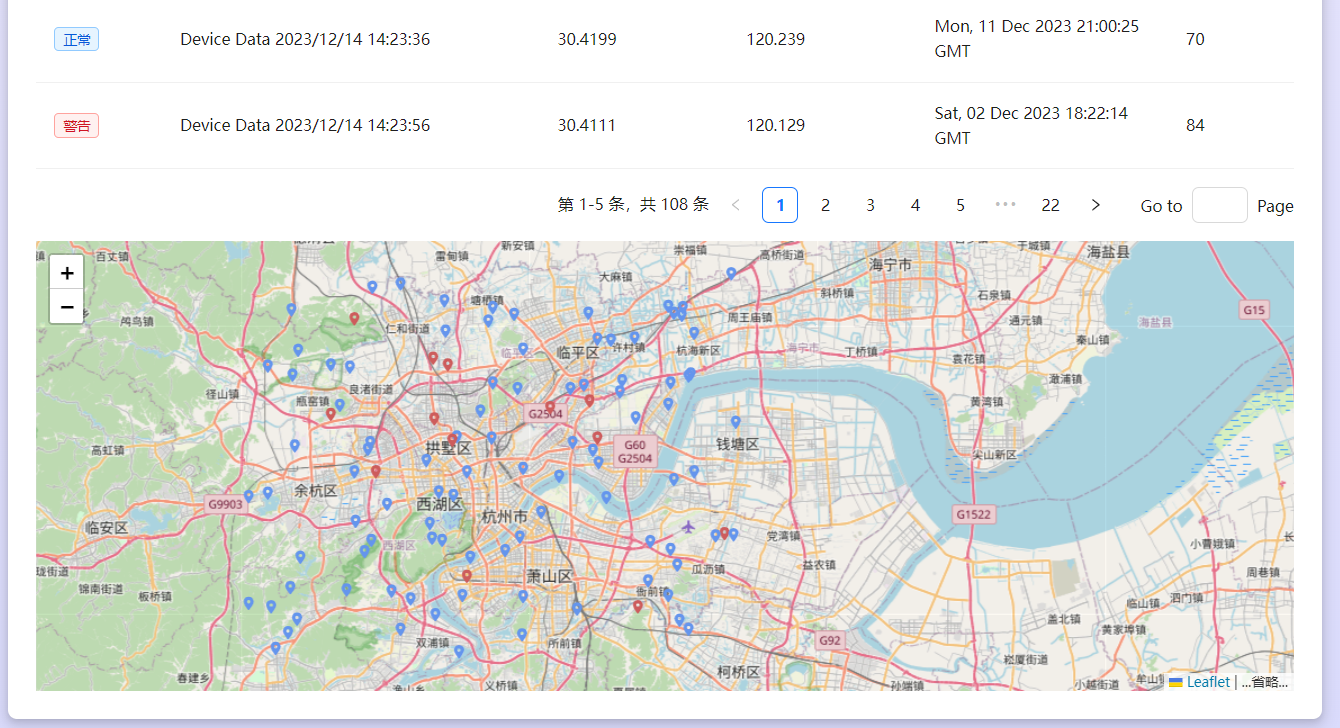
然后在数据统计模块中，可以选中该设备



之后，该设备的消息情况在地图上标记了出来，并且按照不同的类型，颜色也有所不同



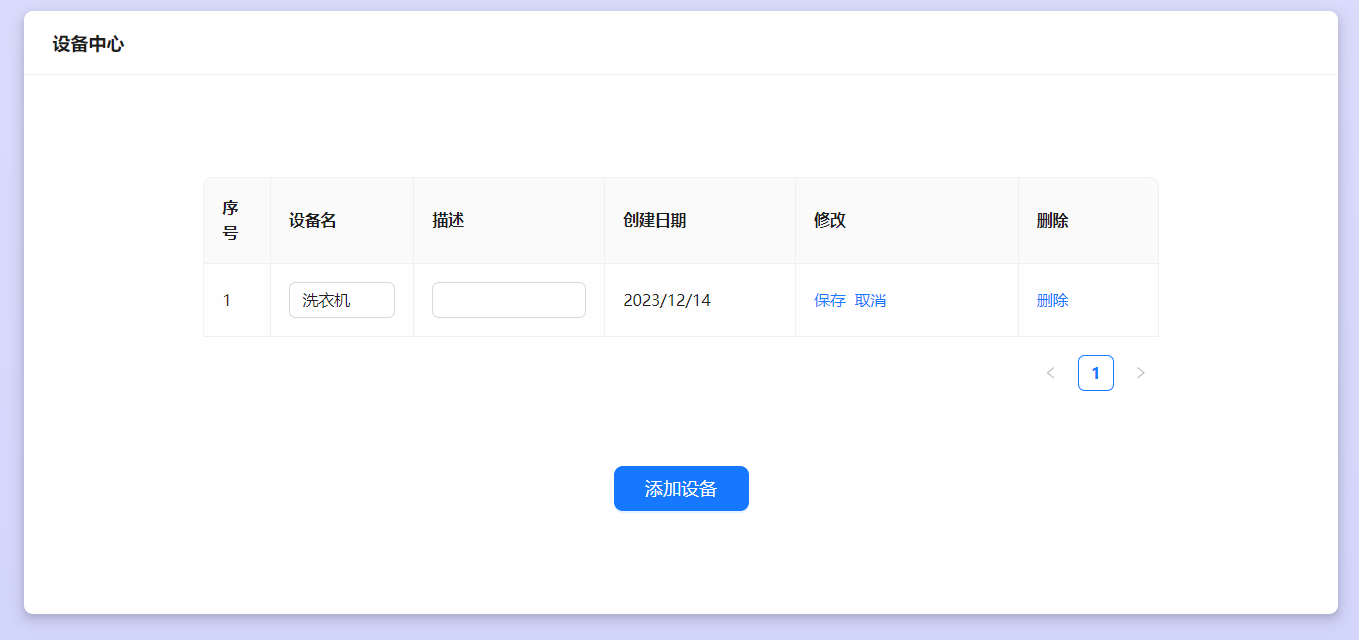
地图效果：



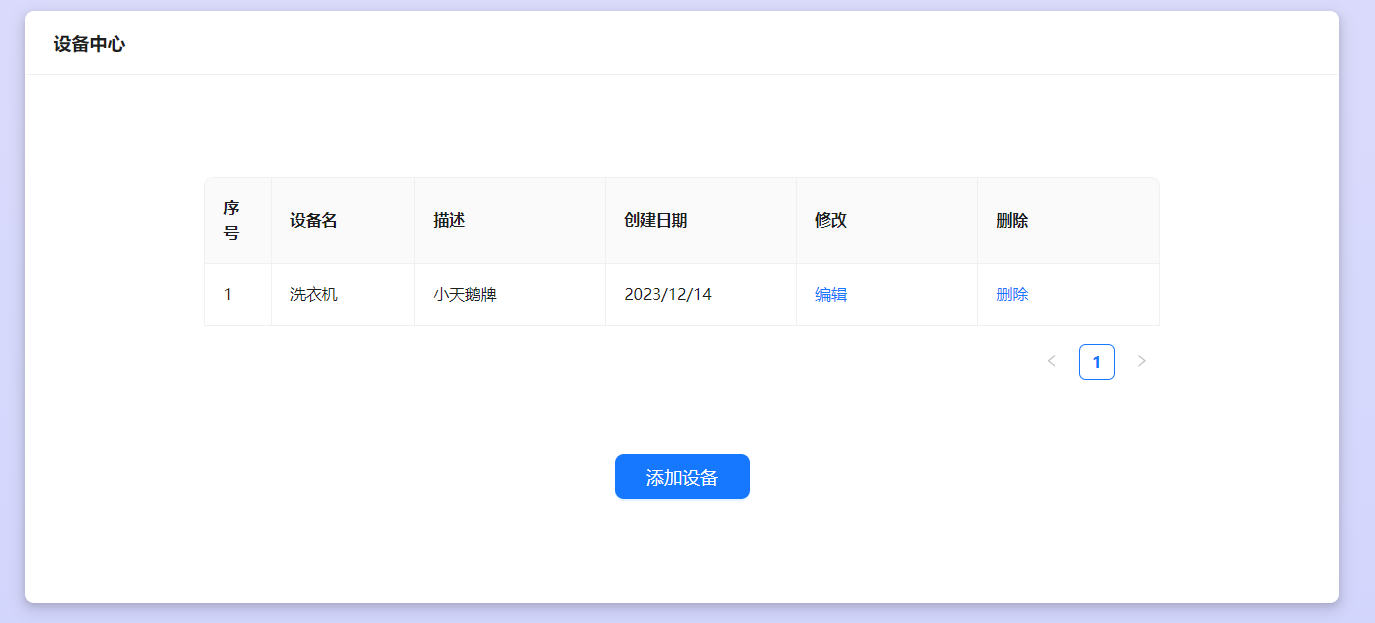
把鼠标悬浮在某个地标上，还能显示该店的一些具体信息



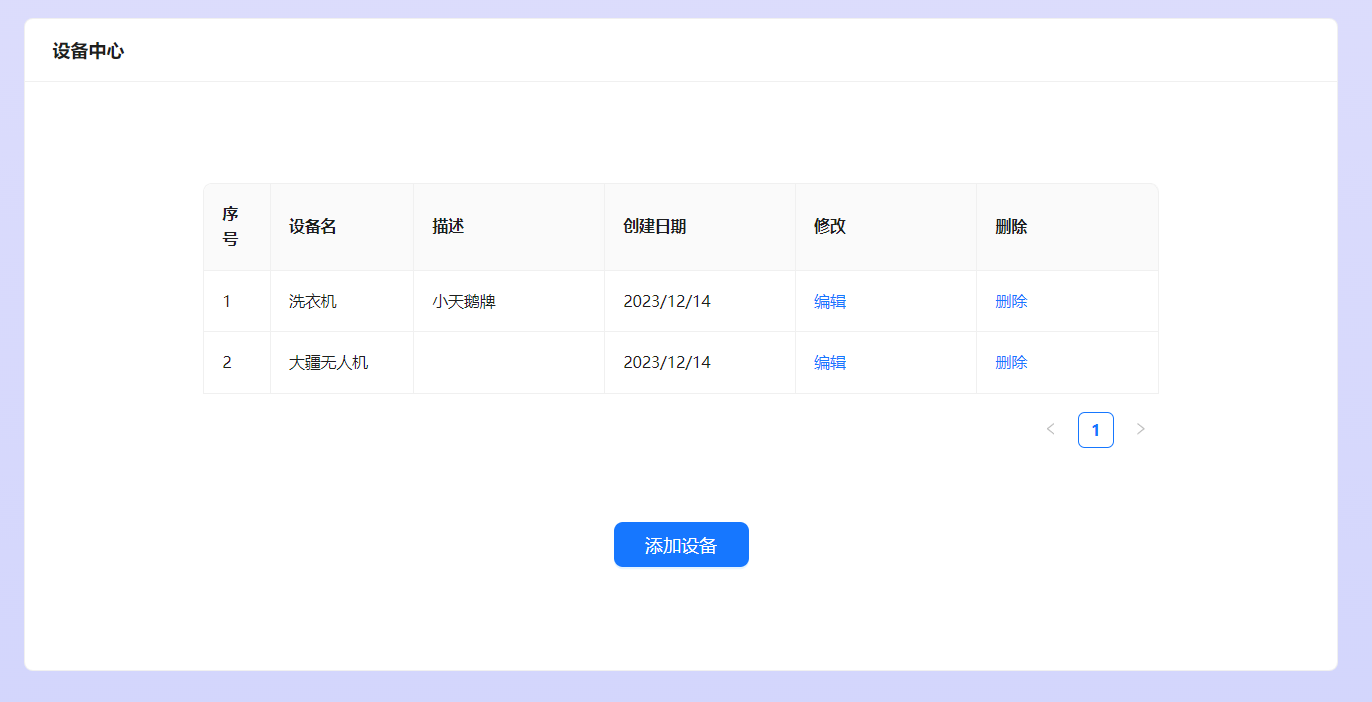
点击设备右侧的更改按钮，进行设备信息的更改



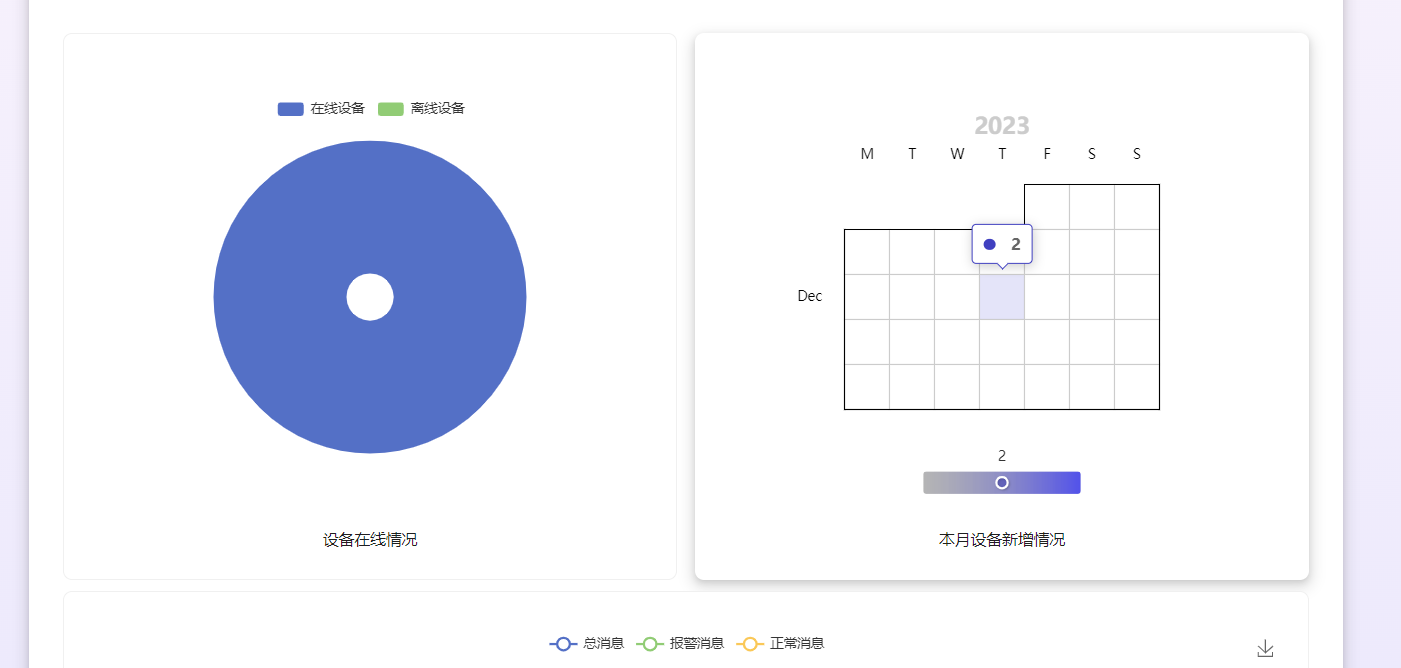
我这里修改了描述，点击保存后可以看到修改的内容被保存



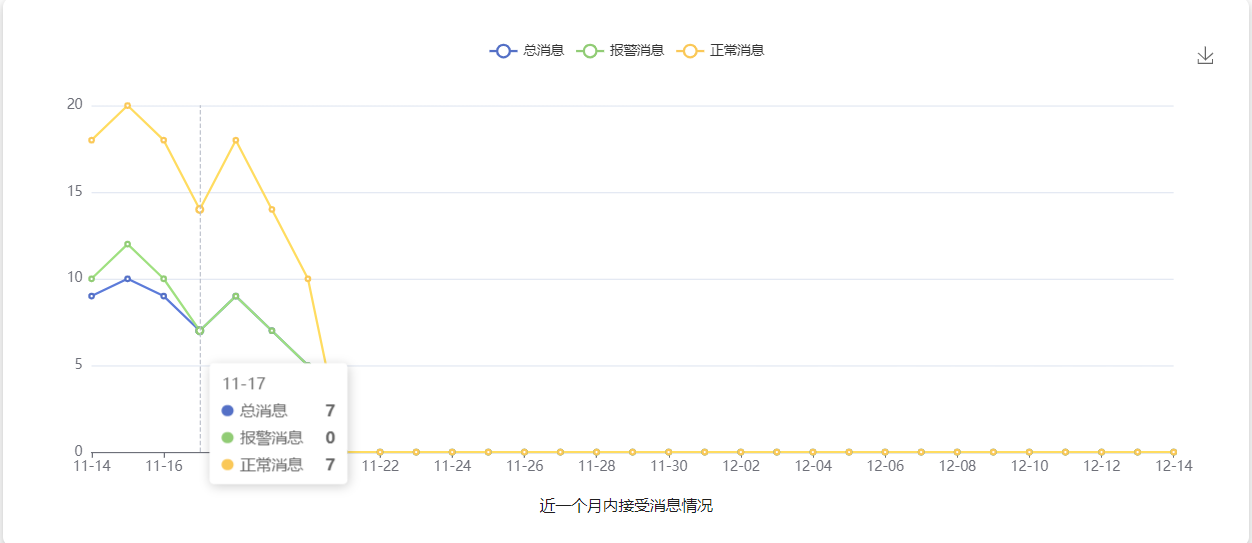
如果我添加第2台设备



可以发现统计图表随之改变

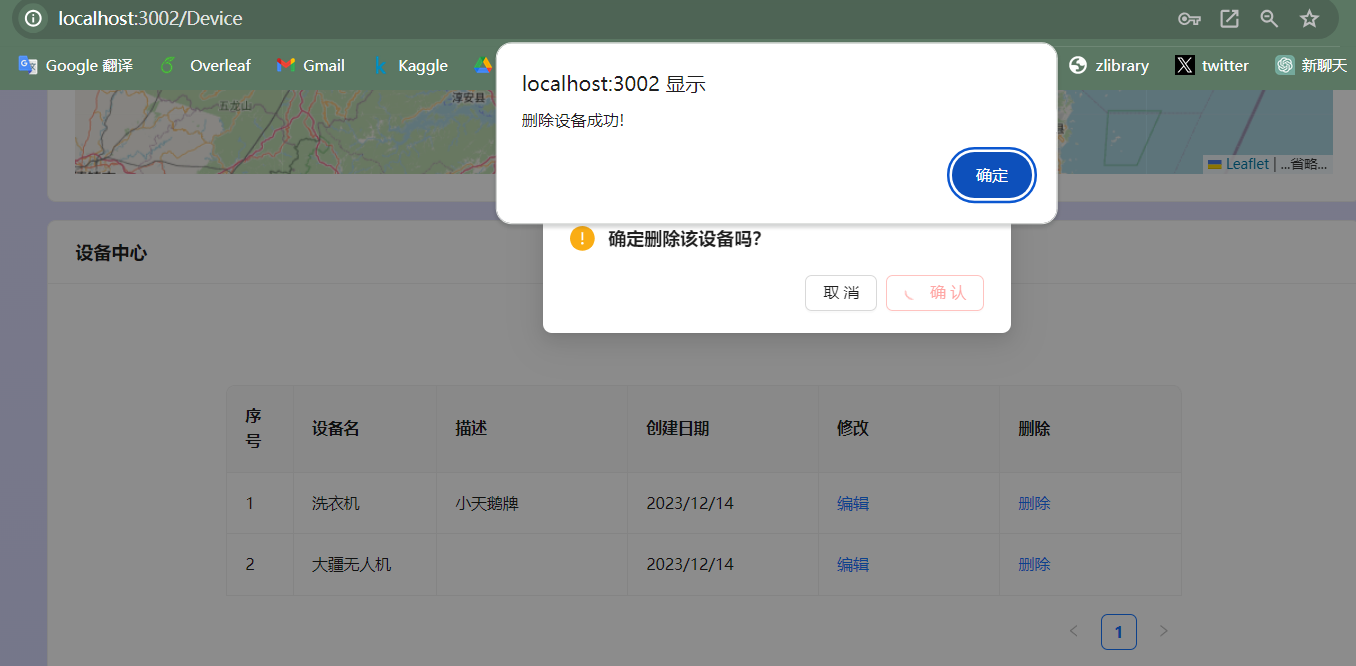


消息数也增加了

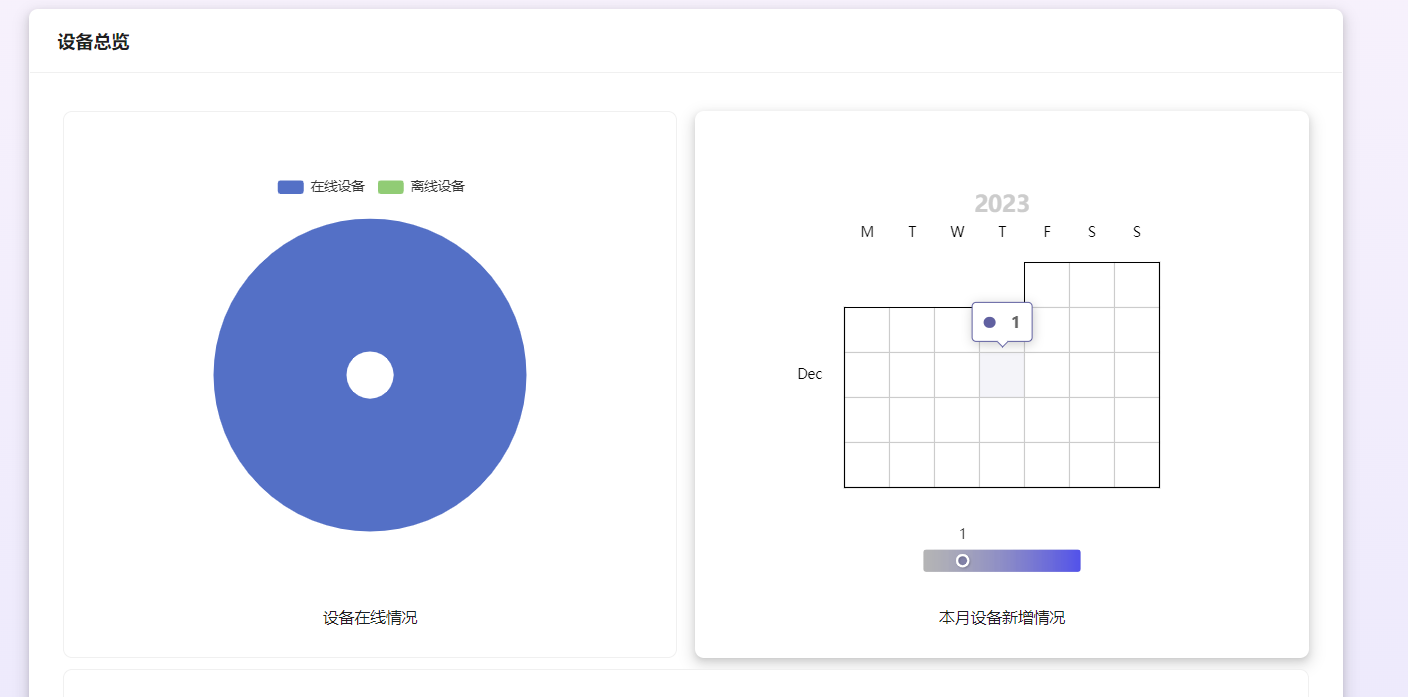


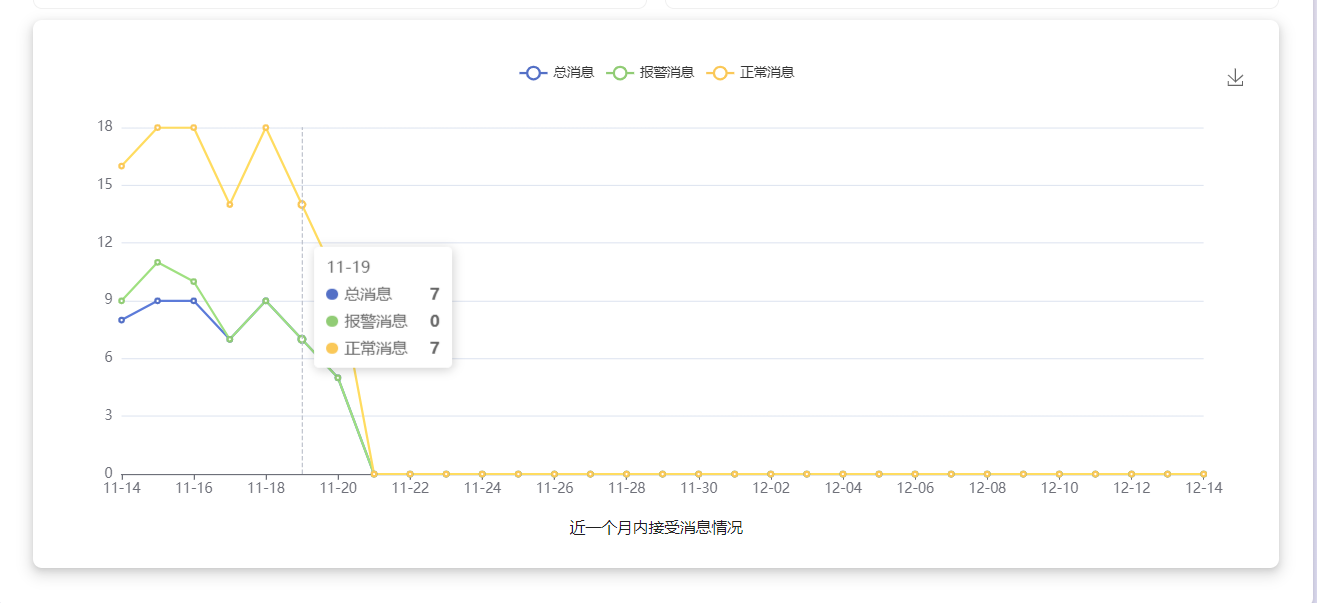
点击右侧的删除按钮并且确定后，设备将会被删除





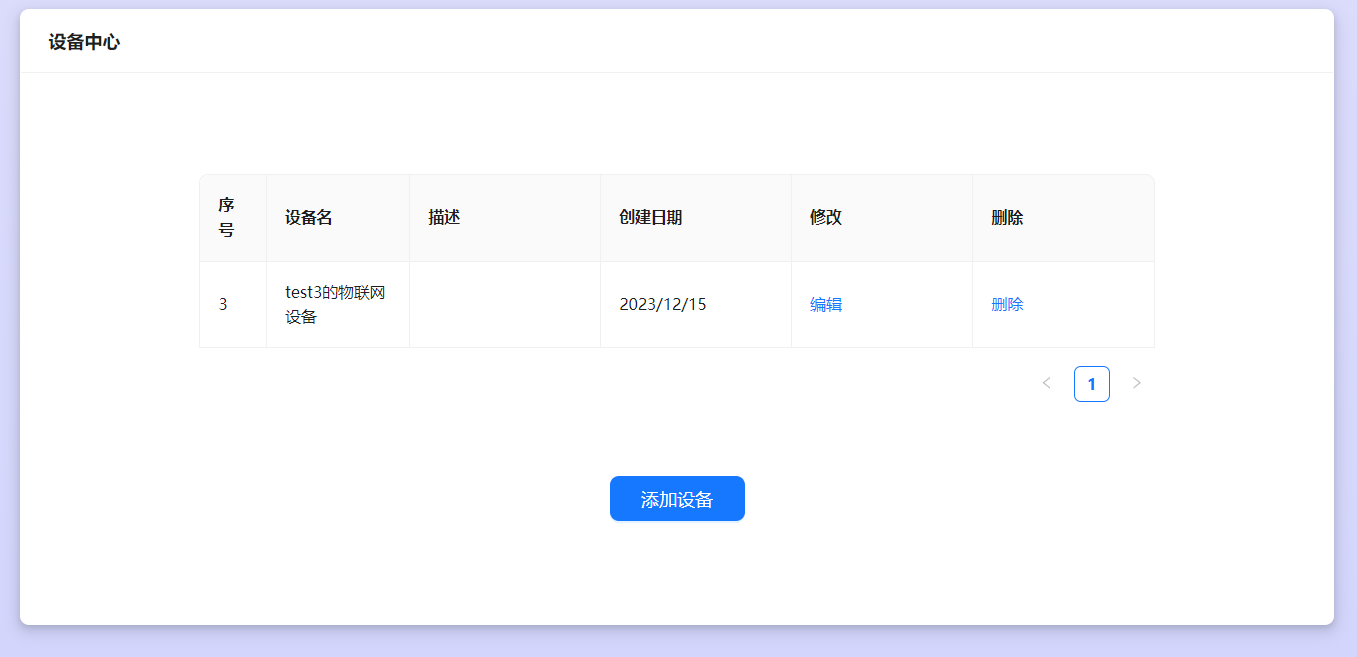
这个时候设备消失了，并且统计图表也进行了刷新





我在另外的两个测试账号分别各有一台物联网设备，可以发现他们的id并不是从1开始的，这是因为id是primary key，并且递增





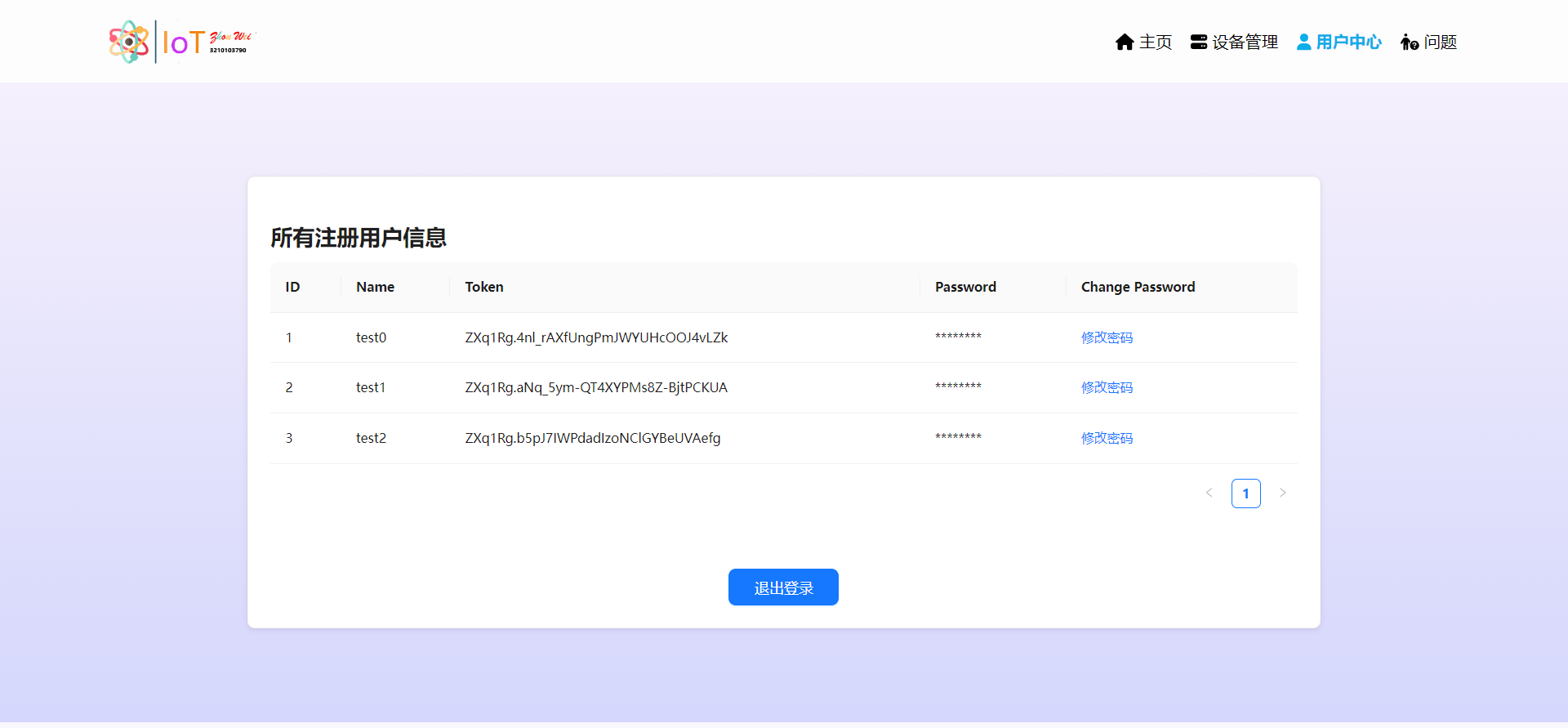
## 管理员模式

在管理员登录窗口输入正确的管理员密钥[[1]](#footnote-1)就可以登录为管理员模式

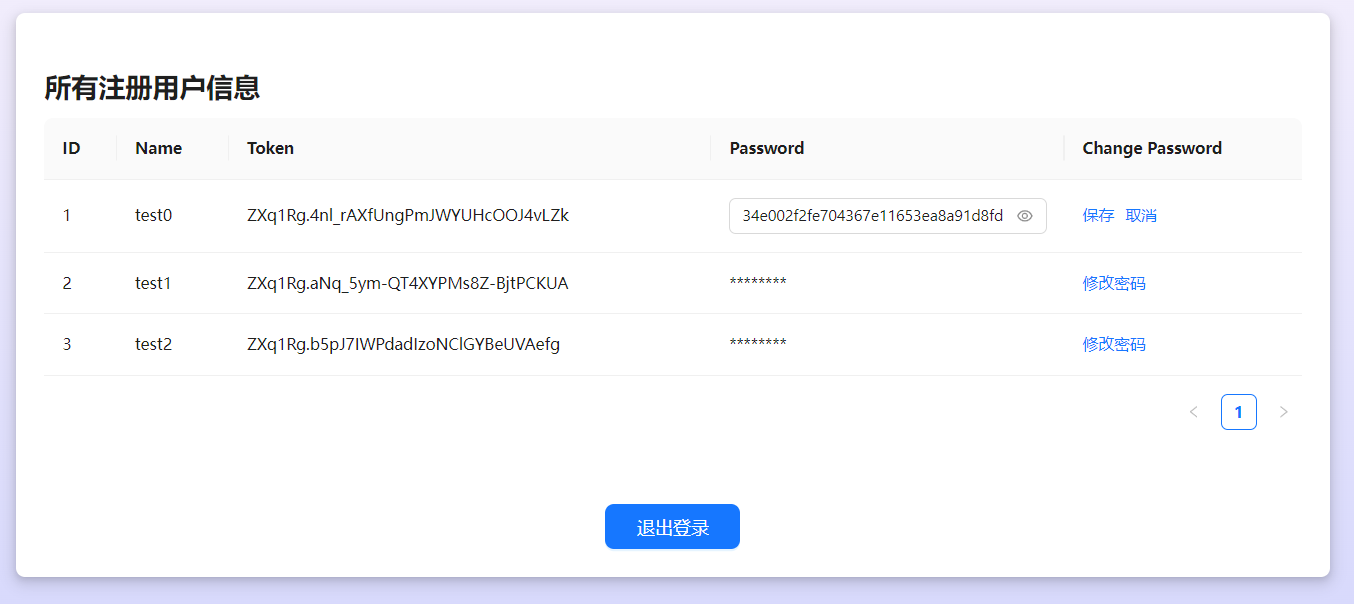


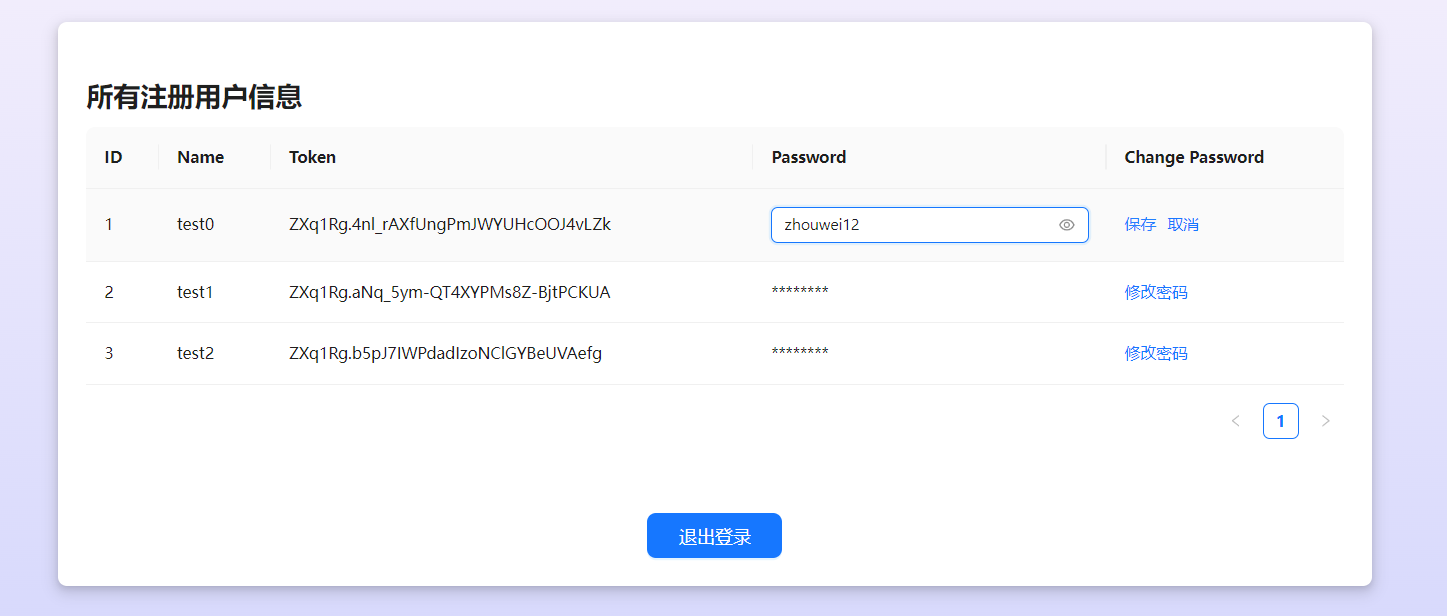
另外需要注意的是，为了保证网站的安全性，管理员模式刷新后即需要重新登录

管理员模式下，用户中心界面显示的是所有用户的信息，包括ID, Name, Token, Password等内容

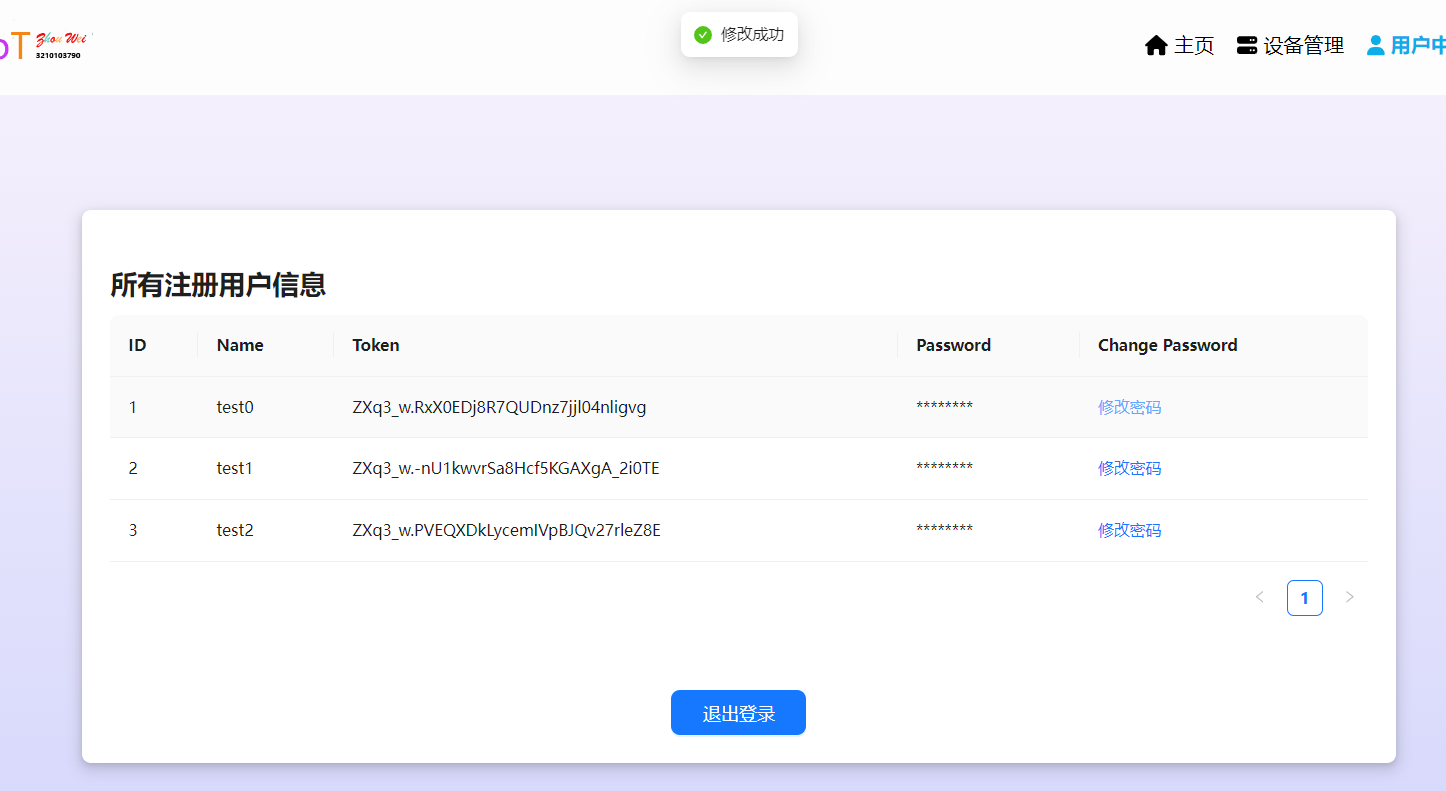


并且，管理员还能帮助用户修改密码(显示的md5加密的结果，由于md5加密不可逆，因此无法显示原密码，显示的是md5码)

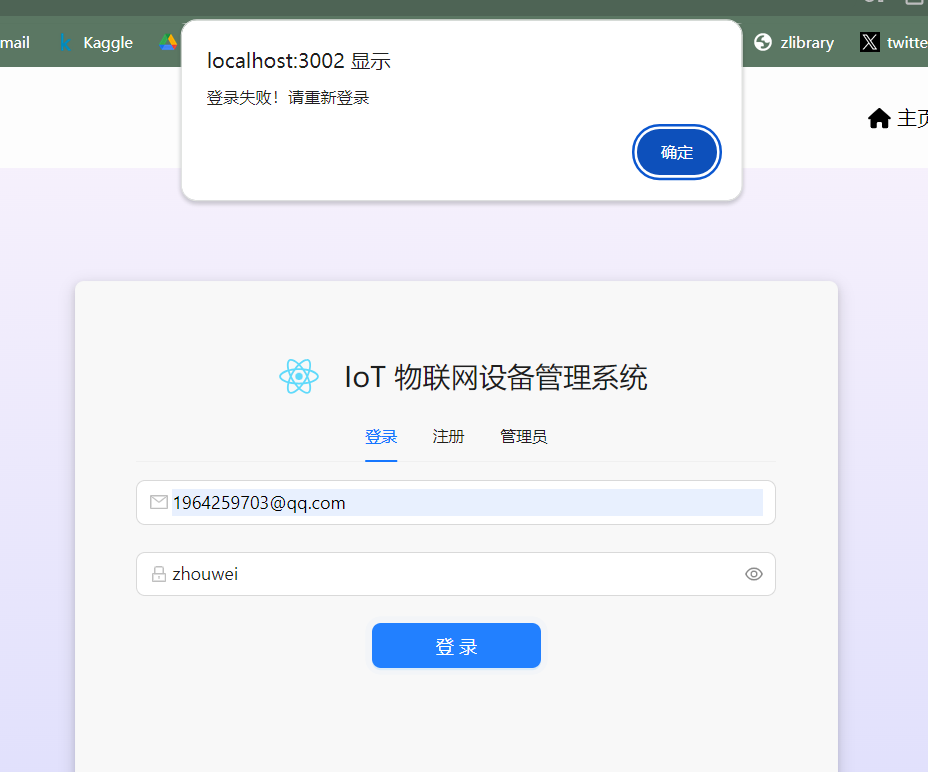




点击保存，即修改成功



这时候，原用户尝试用原密码登录失败



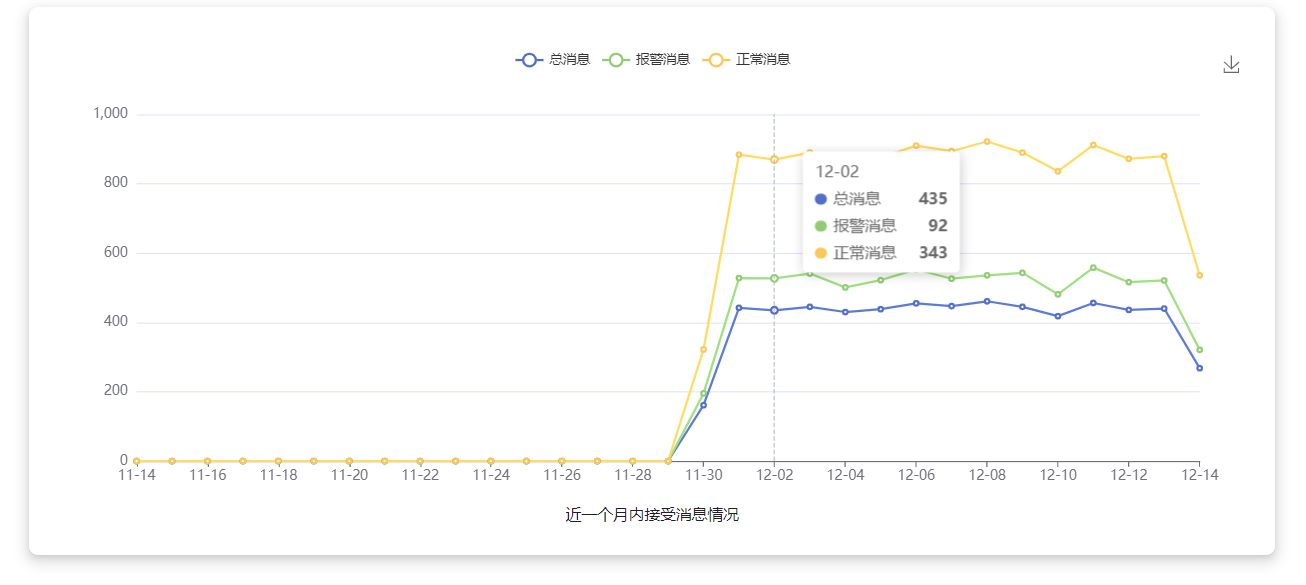
需要改为使用root的密码登录才成功

如果点击设备管理页面，则会立刻跳出一个提示信息，提示用户谨慎操作

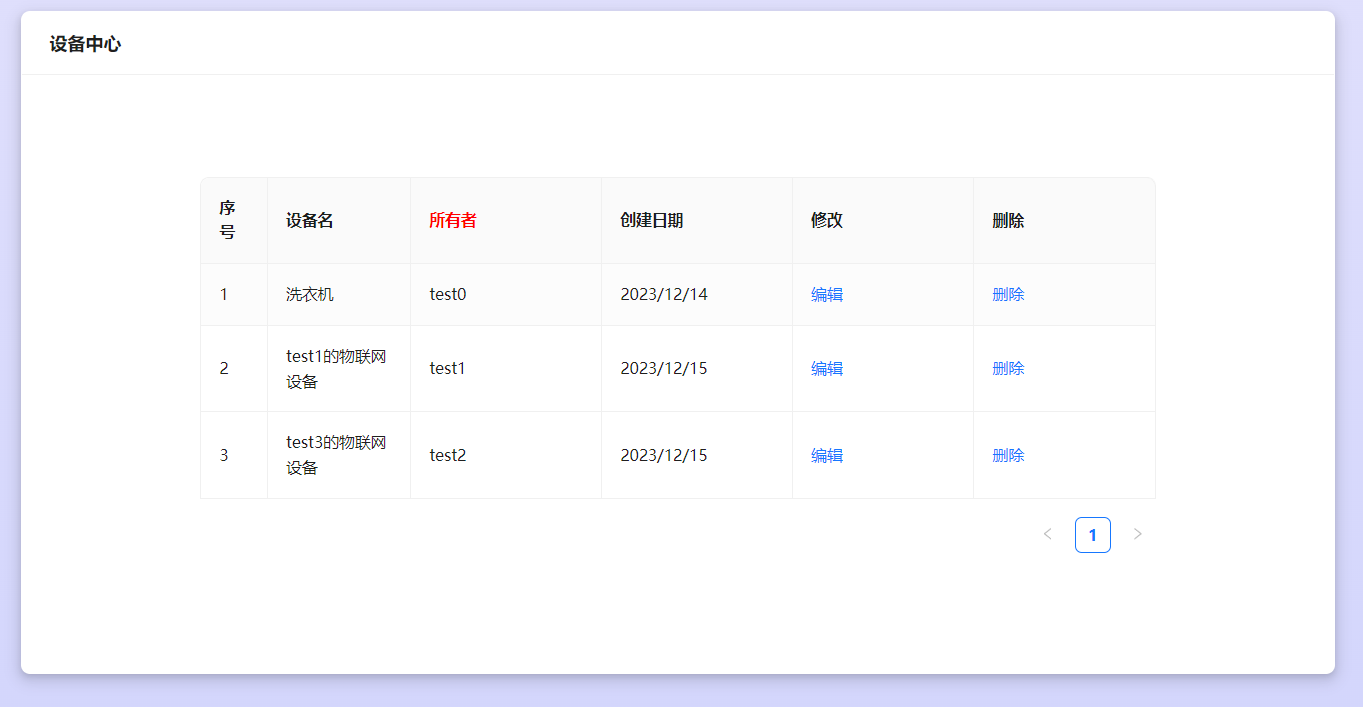


并且此时设备管理页面显示的都是所有用户的总和

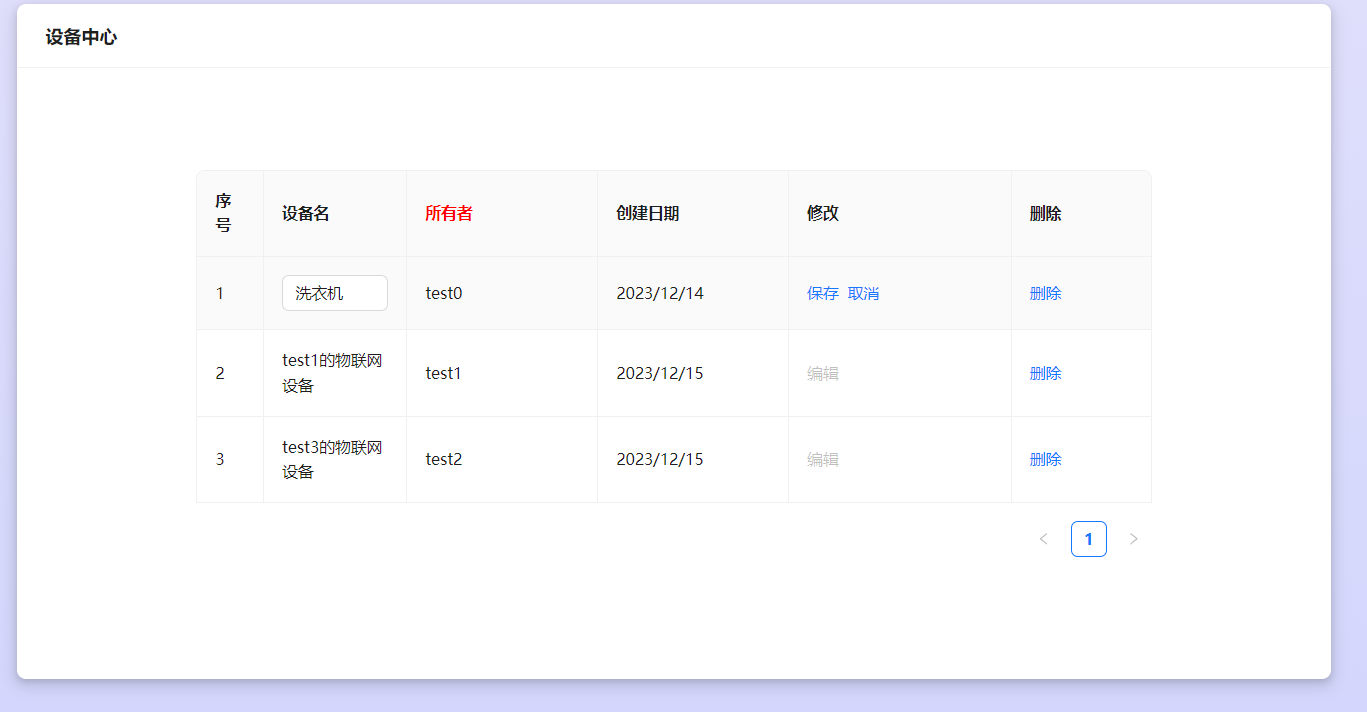




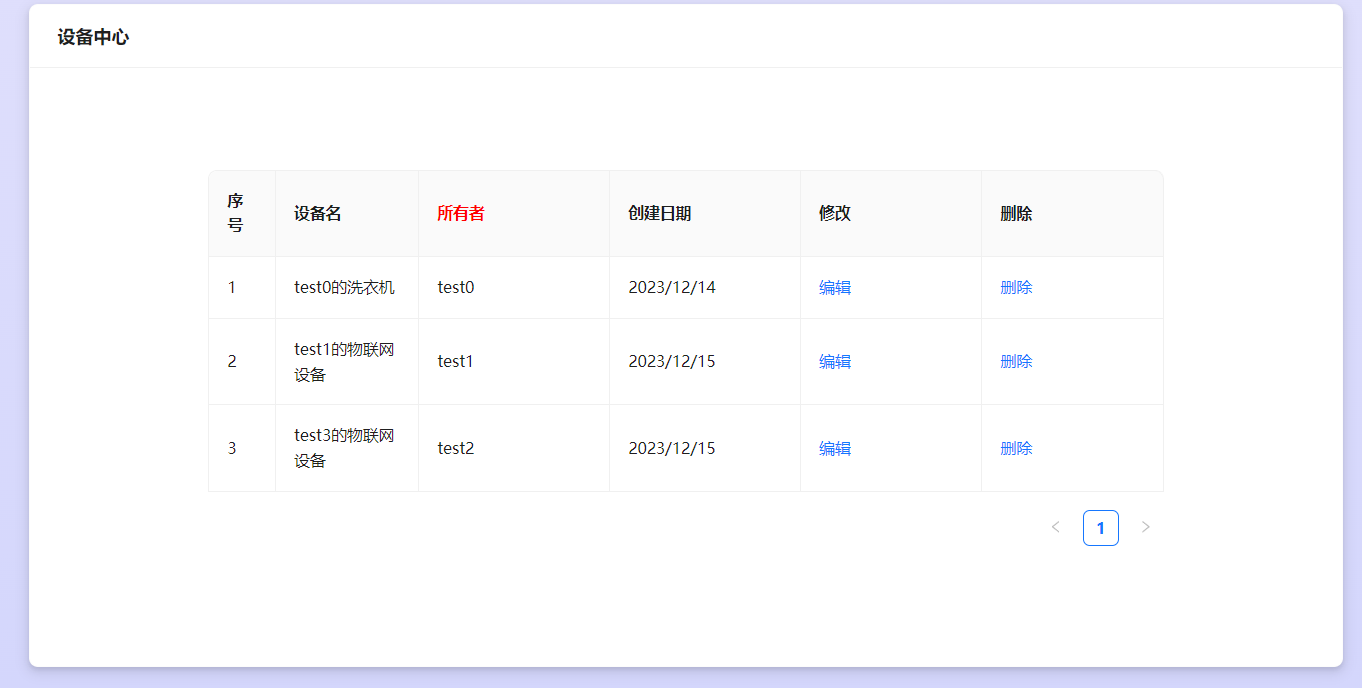
在设备中心一栏中可以发现，test0, test1, test2的设备都显示了出来，与之前演示的建立相符合



点击编辑之后只能编辑设备名了



修改并且保存



这个时候再去test0用户的设备管理界面进行查看，可以发现设备名已经被更改了



# 函数接口

* **getMessage——返回MQTT接收的消息**
* 方法：get
* 接口名：getMessage
* 参数：
  + { 'clientId' : ...}
  + clientId：数据流编号
* 功能：返回这个数据流编号的所有消息
* 返回值
  + json列表（res.data）

{  
code:0,  
data:[  
{  
alert:...,  
info:...,  
lat:...,  
lng:...,  
timestamp:...,  
value:...  
},  
{...}  
]  
}

* **register——新用户注册**
* 方法：post
* 接口名：register
* 参数：
  + { 'name' : ..., 'password':..., 'email':..., }
* 功能：按照当前信息注册新用户
* 返回值
  + json列表（res.data）

{  
code:0,  
   msg:"Register success!"  
}

* **tokenLogin——使用token快速登录**
* 方法：post
* 接口名：tokenLogin
* 参数：
  + { 'token' : ...}
  + token：登录时候传回客户端的身份象征token
* 功能：使用token快速登陆
* 返回值
  + json列表（res.data）

{  
code:0,  
msg:"login success!"  
   data:token  
}

* **login——用户登录**
* 方法：post
* 接口名：login
* 参数：
  + { 'email' : ..., 'password':...}
* 功能：用户正常通过用户密码登录
* 返回值
  + json列表（res.data）

{  
code:0,  
   msg:"Login success!",  
   data:token  
}

* **getUser——获取用户信息**
* 方法：get
* 接口名：getUser
* 参数：
  + { 'token' : ...}
* 功能：通过token解析并返回该用户的名字
* 返回值
  + json列表（res.data）

{  
code:0,  
msg:"getSuccess",  
   data:userName  
}

* **alterPassword——更改密码**
* 方法：post
* 接口名：alterPassword
* 参数：
  + { 'token' : ..., 'oldPsw':..., 'newPsw':...}
* 功能：用户修改密码
* 返回值
  + json列表（res.data）

{  
code:0,  
   msg:"Alter Psw success!"  
}

* **alterName——更改名字**
* 方法：post
* 接口名：alterName
* 参数：
  + { 'token' : ..., 'newName':...}
* 功能：用户更改用户名，顺便把其他数据库的所有有关的都改了
* 返回值
  + json列表（res.data）

{  
code:0,  
   msg:"更改名字成功！"  
}

* **getDevice——获取所有设备**
* 方法：get
* 接口名：getDevice
* 参数：
  + { 'token' : ...}
* 功能：返回这个用户的所有设备
* 返回值
  + json列表（res.data）

{  
code:0,  
data:[  
{  
id:...,  
code:...,  
name:...,  
description:...,  
create\_time:...,  
user:...  
},  
{...}  
]  
}

* **selectDevice——设备搜索**
* 方法：get
* 接口名：selectDevice
* 参数：
  + { 'token' : ..., 'name':...}
  + name：想要搜索的字符串
* 功能：搜索并返回文件命含有相应字符串的设备
* 返回值
  + json列表（res.data）

{  
code:0,  
data:[  
{  
id:...,  
code:...,  
name:...,  
description:...,  
create\_time:...,  
user:...  
},  
{...}  
]  
}

* **alterDevice——更改设备信息**
* 方法：post
* 接口名：selectDevice
* 参数：
  + { 'token' : ..., 'code':.., 'oldName':..., 'newName':..., 'description':...}
* 功能：根据原有的设备信息，更改之后传回来
* 返回值
  + json列表（res.data）

{  
code:0,  
   msg:"修改成功！"  
}

* **createDevice——新建设备**
* 方法：post
* 接口名：selectDevice
* 参数：
  + { 'token' : ..., 'code':.., 'name':..., 'description':..., 'user':..., }
* 功能：根据前端传入的数据新建设备
* 返回值
  + json列表（res.data）

{  
code:0,  
   msg:"create success!"  
}

* **deleteDevice——删除设备**
* 方法：get
* 接口名：selectDevice
* 参数：
  + { 'token' : ..., 'name':... }
* 功能：根据前端传入的设备名删除设备，所以设备名必须唯一
* 返回值
  + json列表（res.data）

{  
code:0,  
   msg:"删除设备成功!"  
}

* **getRecentDevice——获得七天内数据**
* 方法：get
* 接口名：getRecentDevice
* 参数：
  + { 'token' : ...}
* 功能：返回该用户所属的设备过去七天的新建情况
* 返回值
  + json列表（res.data）

{  
code:0,  
   msg:"getRDSuccess!",  
   day:['06-19','06-20','06-21','06-22','06-23','06-24','06-25'],  
count:[0,0,0,0,0,3,1]  
}

* **getRecentMessage——获得七天内消息**
* 方法：get
* 接口名：getRecentMessage
* 参数：
  + { 'token' : ...}
* 功能：token只是用来验证身份，返回七天内收到mqtt消息的情况，分为正常与预警
* 返回值
  + json列表（res.data）

{  
code:0,  
   msg:"getRMSuccess!",  
   day:['06-19','06-20','06-21','06-22','06-23','06-24','06-25'],  
total:[0,0,0,0,0,4,10],  
   normal:[0,0,0,0,0,3,8],  
alert:[0,0,0,0,0,1,2]  
}

# Q&A 常见问题

**为什么目前只有device0001~device0005的数据源**

老师提供的客户端只能发送5个数据流，所以只有这5个数据流，设置成其他数据流的话将会被变成离线状态，没有办法接受到任何消息

1. 我的代码中用**root**作为管理员模式的密码 [↑](#footnote-ref-1)