基于 iMaster NCE-CampusInsight 的大数据疫情监控分析系统设计文档

所在赛道与赛项：（*填写"A/A-ST/B-EP1/B-EP2"其中一项*）。

一、目标问题与意义价值

2020 年年初疫情突发，新型冠状病毒具有极强的人传人能力，为了疫情监控政府推出健康码，行程码等手段，但这些只能大概的定位用户去过哪些地方，得到一个大体的行程路线。当患者进入某些封闭或半封闭的园区（非公共管辖范围）时，则无法获取到更小精度的行动轨迹，导致该园区的所有人员都被标记为高风险人员，为后续的排查工作增加了不少人力物力消耗。

本系统在传统的 SDN 系统基础上，添加了疫情监控、用户行为分析等功能。本系统可以在此基础上弥补"最后一公里"的不足，通过输入确诊者的手机唯一识别号（mac）准确的定位出患者在该系统覆盖范围内具体去过哪些地方并且生成出一条可视化的轨迹路线，对于防疫工作来说具有更细的精度和更加准确的摸排。除了可视化路线外，该系统还会返回患者轨迹附近有可能与患者接触的人员名单。该名单可以帮助相关部门更快的找到疫情传播轨迹、源头等信息。并且通过接触人员名单，通过热力图等形式直观的显示出疫情的扩张趋势、有可能被污染的区域。

其次该系统还可以接入政府相关平台获取现存确诊名单，通过比对数据库接入数据可以实现当有患者曾来过或正滞留在园区时快速响应告警。如相关部门需求，该系统还可以提供相应的 api 接口给有关部门调用，有助于疫情的防控。

此外，本系统还有人群热力图功能，通过用户接入信息、用户行为数据可以帮助管理者分析出系统范围内用户的画像，帮助管理者更好的改善和管理园区。比如改善园区人流的导流、改善交通规划等等

二、设计思路与方案

技术选型：

该系统是基于华为提供的 iMaster NCE-CampusInsight 接口（下称华为接口）实现的，因此我们需要先从华为接口获取原始数据，经过系统的数据处理模块处理数据后返回给用户。此外还计划实现多端设备查看园区网络数据和将园区数据推送或提供接口给相关部门调度。故这里选择前后端分离的架构。

框架与工具选择：

前端框架方面，JavaScript 框架选择 Vue.js、网络请求库使用 Axios、UI 框架使用 element UI。并使用 Vue-cli、Vue-router、Vuex 等插件进行项目、路由、网页存储的管理。对于数据的可视化展示采用了当前流行的 Echarts 框架绘制为图表，使数据信息更加简易理解。

后端框架选择的是 APIFlask 框架，该框架是一个基于 Flask 和 marshmallow-code 的轻量级 web 框架。ORM 框架方面选择的是 SQLAlchemy，鉴权方面采用 APIFlask 集成的 Flask-HTTPAuth 扩展。网络请求方面采用 aiohttp 和 requests 库。

数据库方面采用 MySQL 作为持久化存储，Redis 作为缓存数据库

服务器采用 Nginx+Gunicorn 作为网关。Nginx 是一个高性能服务器，支持负载均衡、正反向代理，此次部署的是网络管理系统采用前后端分离结构，因此前后端的交互会非常的频繁，这之中还存在跨域等一系列问题，故采用 Nginx 的反向代理、负载均衡等功能可以有

效的解决这些问题。此外，Gunicorn 是一个 UNIX 下的 WSGI HTTP 服务器，它既支持 eventlet ， 也支持 greenlet。在管理 worker 上，使用率 pre-fork 模型（一个 master 管理多个 worker)。是一个高性能的异步网关，上和 nginx 下和项目的 aiohttp 结合起来使用效果显著优于其他组合。

服务端的通信流程如下:

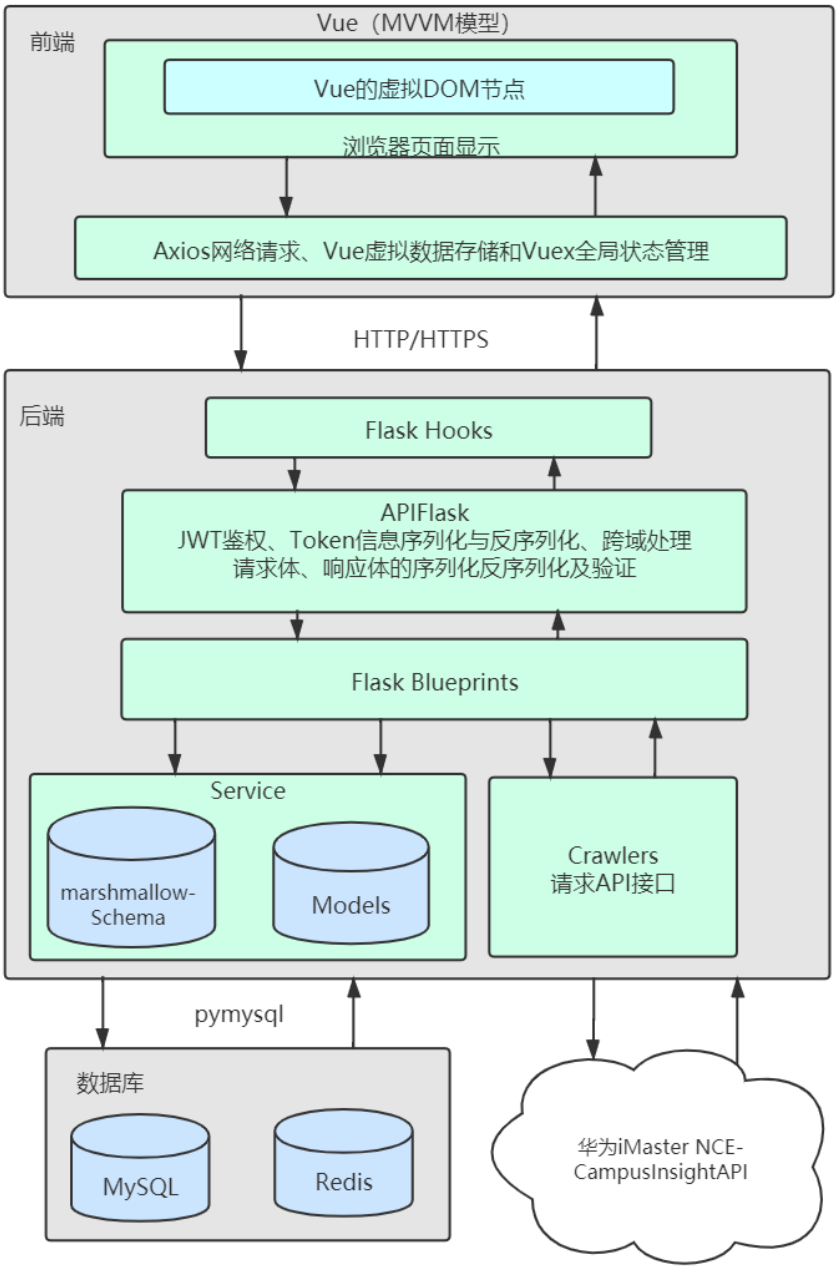
nginx<->gunicorn<->flask

详细设计：

基础端口（不需要计算加工）后端根据前端参数进行请求

加工端口: 后端根据前端请求，将任务添加到任务队列里面，定时任务定时从任务队列里面获取任务进行数据处理，前端一段时间后再请求相应接口获取加工好的数据。

整体设计流程如图所示：



三、方案实现

1. 前端设计

1.1 页面解构设计

总体结构按照以下结构进行设计

1）系统状态（系统总览）

1. 站点选择
2. 时间选择
3. 系统信息概览

2）用户列表

1. 用户列表
2. 用户数据分析

3）健康度分析

1. 接入成功率
2. 接入耗时
3. 信号干扰
4. 漫游达标率

4) 园区热力地图

5）用户轨迹图

1.2 组件设计

1.2.1 站点数设计

功能：能展示站点的树形结构，站点数据的获取与处理逻辑在组件内部进行

用途：在需要使用的页面，提供给用户选择站点功能。

1.2.2 时间选择器

功能：能让用户方便的选择日期，组件默认选择近一周。

用途：各个需要依赖 Unix 时间戳参数的功能

参数：start->起始时间；end->结束时间

组件事件:

i. changes：事件被更改后触发，参数一作为数据对象

ii. 参数一:

type：Array

describe：以数组形式返回开始时间和截止时间

1.3 页面功能设计

1.3.1 系统总览（或者叫系统状态）

功能：浏览系统各个维度的信息

布局：页面以弹性布局为主，每个维度的状态单独封装成一个盒子组件（卡片组件）自动根据页面宽高进行自适应。

展示内容:

1、接入设备数量、AP 数量、历史接入设备数量（状态概览卡片）

2、登录用户信息；（账号信息卡片）

3、接入失败信息；（echarts 饼图）

4、站点总速率；（echarts 仪表盘）

5、站点状态概览；（echarts 雷达图）

6、服务器 cpu 使用情况；（echarts 仪表盘）

7、服务器内存使用情况。（echarts 饼图）

1.3.2 用户列表

相关 API：获取用户数据分析信息

功能：接入园区网络所有用户（设备）的信息 （对站点下的用户数据进行统计和分析）.

展示内容：历史接入用户的概览以及详细信息、可以指定页码跳转并且能查看选中用户的轨迹路线。

布局：表格展示概况、详细卡片展示单独用户的详细情况、分页控制器（每页可选 5,10,20,.40,100,200）

1.3.3 健康度分析

功能：展示园区站点接入信息以及各种健康度指标。

展示内容：漫游达标率、信号干扰、接入耗时、接入成功率等。

1.3.4 位置地图（热力图）

相关 API：取热力图

数据处理方法（二选一）：

1. 直接计算映射权重：(count – 最小 count) / (最大 count – 最小 count)
2. 选择性精准扶贫法：
   i. 定义：
      1. 太小：值小于平均值的 5%
      2. 一般小：值大于平均值 5%但小于 15%
   ii. 处理：
      1. 若数据太小，则丢弃
      2. 若数据一般小，映射权重设为平均值的 50% + 原值
      3. 其他情况映射权重与原值相同

1.3.5 用户轨迹图

功能：查看用户的园区内的移动轨迹。

展示内容：

1. 显示 ap 节点信息
2. 显示 ap 节点健康信息（绿色、黄色、红色表示）
3. 显示用户的移动轨迹

1.3.6 日志分析

功能：查看选择站点的详细日志信息

内容：各个时刻的关联总数、关联成功数、关联失败数、认证通过数、认证失败数、DHCP 成功数、DHCP 失败数等信息

布局：表格展示，内容作为表格的字段

1.3.7 账号管理

功能：查看 SDN 系统平台的账号以及管理相关账号的权限，登录状态和新增、删除账号等功能。

布局： 表格布局，账号状态用 check button 控制开关（开表示可登录，关表示不可登录）。

2. 后端设计

2.1 模块设计

后端系统采用工厂模式，各模块如下设计：

1. 蓝图模块（Blueprints）
2. 模型模块（Models）
3. 验证模块（Schemas）
4. 爬虫模块（Crawlers）
5. 错误处理模块（Exceptions）
6. 日志模块（Logger）
7. 配置模块（Settings）
8. 辅助函数模块（Utils）

2.2 爬虫系统设计：

1. 基础 crawler

功能：请求华为 API，获取原始数据（raw data）并返回

私有字段：token（表示华为 API 的 token）

方法：

1) 生成 http 请求头，方法名：generate_header
2) 处理 GET 请求的 API，方法名: fetch
3) 处理 POST 请求的 API，方法名: post
4) 处理 PUT 请求的 API，方法名: put
5) 获取华为 API 的 token，方法名: get_token
6) 上锁获取华为 API 的 token，方法名：mtx_token (loop_token)

错误处理

1) 鉴权错误（没有 token 或者 token 过期）
处理方法：抛出 token 鉴权失败错误（错误类型从错误处理模块中导入）
2) 连接错误
修改原始错误信息，抛到最外层处理 handler 处理。

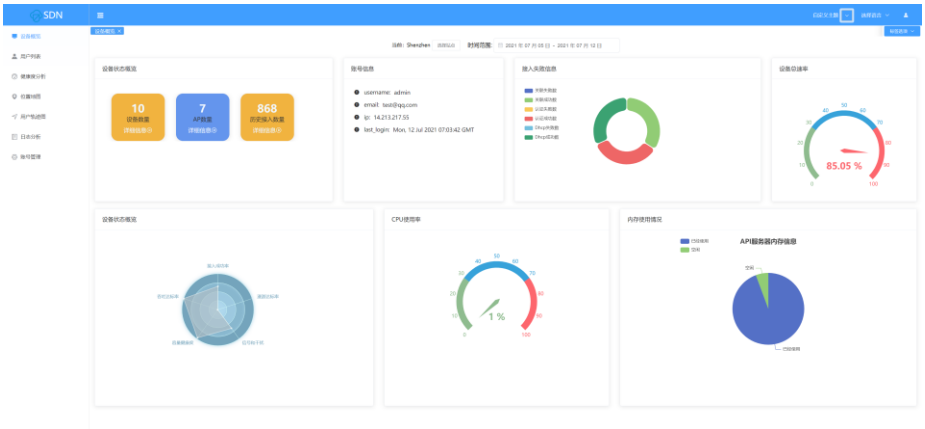
2. 实现模块

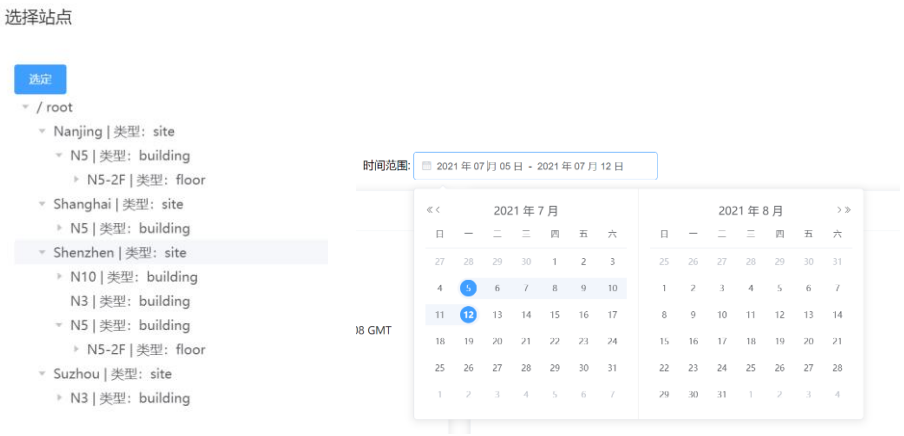
功能：根据华为 API 文档，请求具体数据。

处理 token 错误: 利用基础 crawler 的 mtx_token (上锁获取 token) 方法获取 token 后再次请求接口数据。

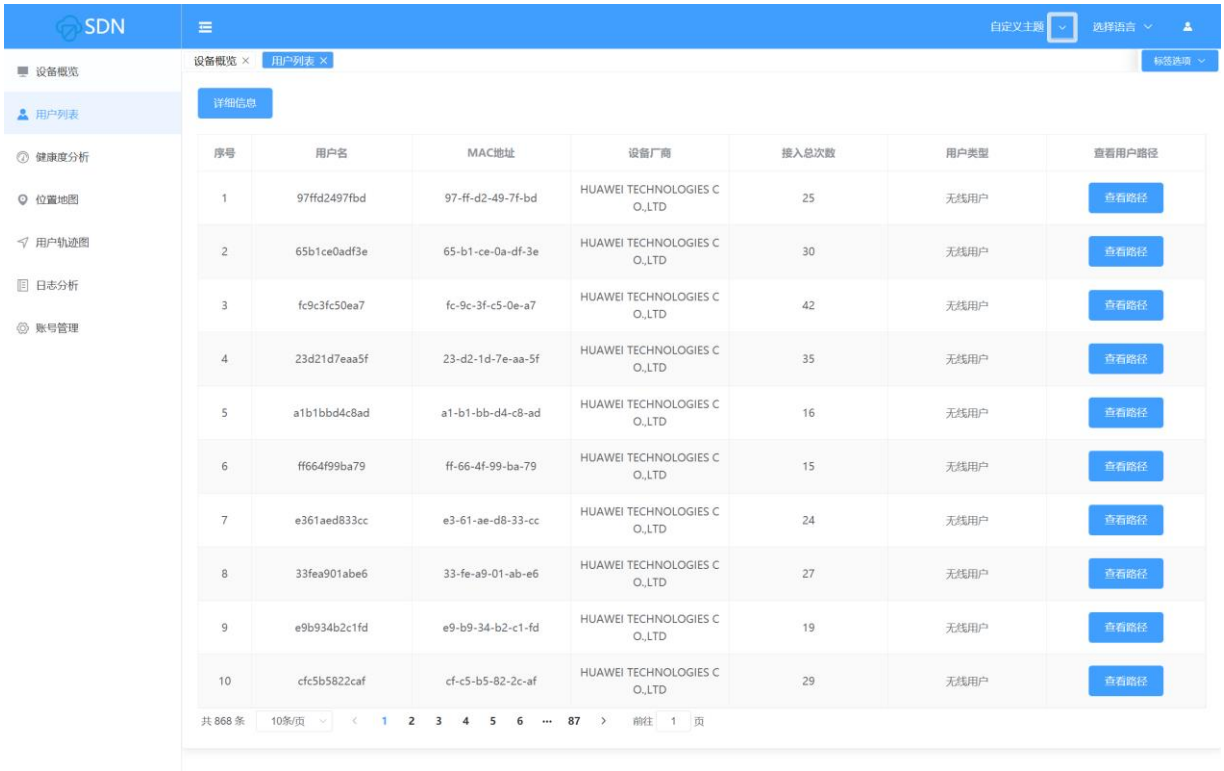
四、运行结果/应用效果

项目在线运行地址：http://c4.farmer233.top/ 用户名: admin 密码: admin





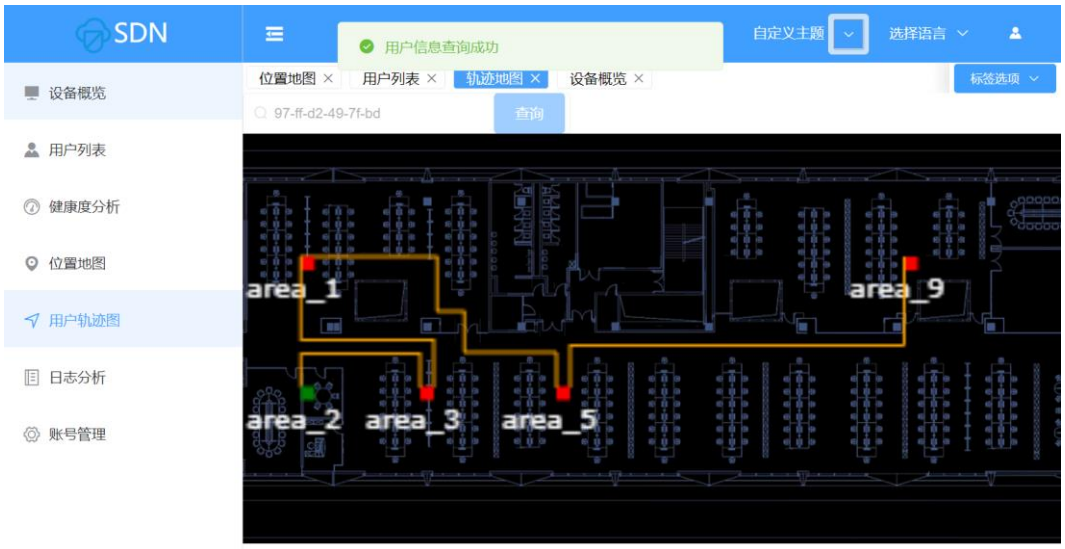
首页主要展示了系统及设备的状态，首页的站点选择器、时间选择器可控制系统全局展示某一站点某一时间段的数据信息；设备状态概览模块提供设备数量、AP 数量、历史接入数量等信息，历史接入数量可跳转查看历史接入用户的详细信息；账号信息模块主要展示了本次登入此系统的账号信息；接入失败信息，移动到饼图区域中，可查看系统当前关联成功数、关联失败数、认证成功数、认证失败数、Dhcp 成功数、Dhcp 失败数等基本信息；设备速率显示的是当前站点设备的速率信息；设备状态概览模块展示了当前站点设备的接入成功率、漫游达标率、信号与干扰、容量健康度、吞吐达标率等基本信息；CPU 使用率模块、内层使用情况模块主要展示了服务器的运行状态；顶部的标签栏可实现页面切换，记录浏览过的页面，也可以通过点击标签尾部的 x 图标关闭当前页面，标签选项拥有关闭其他标签、关闭所有标签功能。



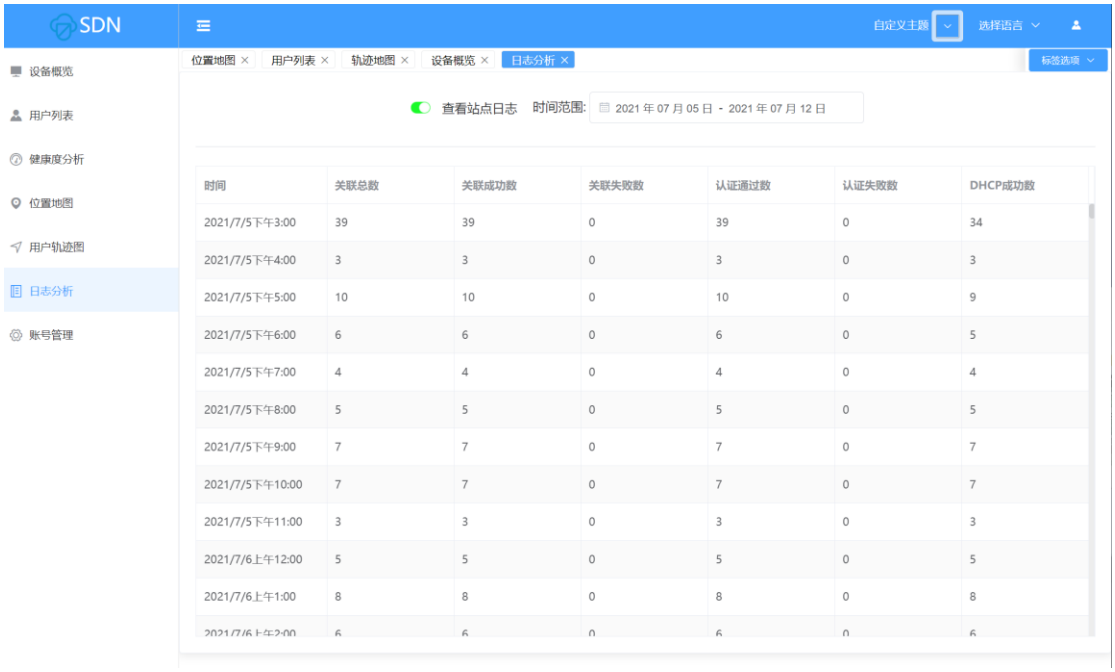
通过用户列表页面可以查看到用户的基本信息，例如用户名、MAC 地址、设备厂商、接入总次数、用户类型信息，点击左上角的“详细信息按钮”可查看当前页面展示用户的详细信息，主要包括了用户名、用户 MAC、是否为 VIP 用户、设备厂商、接入时间、总接入时间、接入失败次数、第一次接入时间、总体验时长、最近接入、平均接入耗时、平均 RSSI、平均速率、平均信噪比、总流量、时延、丢包率等信息，通过点击查看路径按钮可查看当前用户在当前区域行走的轨迹图。

健康度页面主要展示了当前站点、当前时间段的站点设备数据，通过本页面的时间选择器可指定展示某一时间段的数据信息，本页面图表拥有左右滑动、伸缩、保存为图片、折线图与柱状图来回切换功能。

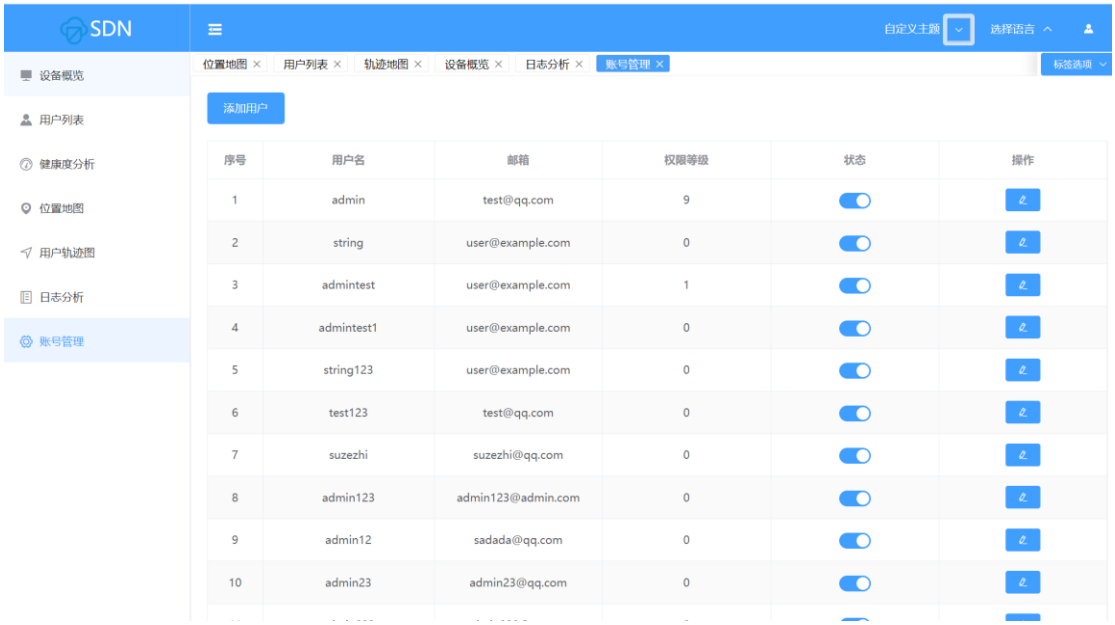
热力图主要展示了当前时间段用户在本区域的位置信息、人群分散、聚集情况。通过结合用户轨迹，查看实现感染用户所经过区域的人群聚集情况，从而推测出密切接入人员。



用户轨迹图描绘了某一用户在当前区域的行走路径。可通过顶部搜索框输入用户 MAC 地址查看指定用户的行走路径图。



通过日志分析页面、可查看当前站点设备接入的日志情况，默认显示为七天，可以通过时间选择器来修改指定时间范围，查看指定时间的设备日志信息。



用户管理页面主要包括了登入此系统的账号信息，展示当前系统所有用户的用户名、邮箱、权限等级、状态等信息、通过修改用户状态，可实现该用户是否能正常登入系统，通过点击操作中的修改按钮，可以修改用户的邮箱、密码信息，通过点击添加用户按钮，可以实现添加系统账号。

五、创新与特色

1. 与传统 SDN 相比，本系统加入了用户轨迹图功能，热力图功能，能更直观的看到园区人流信息、人流轨迹，方便管理园区。
2. 本系统能接入政府相关部门的平台，能及时的响应、辅助上报疫情详情。
3. 后端使用了 APIFlask 框架和 aiohttp，利用协程来处理华为 API 的请求。

六、项目结构树

前端结构树:

```
.
├── README.md
├── babel.config.js
├── images
│   └── bg.jpg
├── package-lock.json
├── package.json
├── postcss.config.js
├── public
│   ├── favicon.ico
│   └── index.html
├── src
│   ├── App.vue
│   ├── api
│   │   ├── auth.js
│   │   ├── index.js
│   │   ├── sdn.js
│   │   ├── server.js
│   │   └── user.js
│   ├── assets
│   │   ├── css
│   │   │   └── common.scss
│   │   ├── img
│   │   │   ├── about
│   │   │   │   ├── bg.gif
│   │   │   │   ├── bisai1.jpg
│   │   │   │   ├── close.png
│   │   │   │   ├── my-logo.png
│   │   │   │   ├── open.png
│   │   │   │   ├── 拓扑.png
│   │   │   │   └── 网络安全.png
│   │   │   └── equipment
│   │   │       ├── AP.png
│   │   │       ├── Controller.png
│   │   │       ├── Firewall.png
│   │   │       ├── Server.png
│   │   │       ├── Switch.png
│   │   │       ├── Switch2.png
│   │   │       └── Yun.png
│   │   ├── lang
│   │   │   ├── en.js
│   │   │   └── zh-cn.js
│   │   └── logo.png
│   ├── axios.config.js
│   ├── components
│   │   ├── ThemePicker.vue
│   │   ├── fAnnularCard.vue
│   │   ├── fCpuRate.vue
│   │   ├── fDashboard.vue
│   │   ├── fDeviceBlock.vue
│   │   ├── fGeneralStateView.vue
│   │   ├── fMemory.vue
│   │   ├── fMessageCard.vue
│   │   ├── fRadarmap.vue
│   │   ├── fSiteSelector.vue
│   │   ├── others
│   │   │   ├── Annular.vue
│   │   │   ├── CpuRate.vue
│   │   │   ├── Dashboard.vue
│   │   │   ├── MemoryInfo.vue
│   │   │   └── RadarMap.vue
│   │   ├── timeSelector.vue
│   │   └── xiaotao
│   │       └── SiteTree.vue
│   ├── main.js
│   ├── router
│   │   ├── index.js
│   │   ├── staticRouter.js
│   │   └── whiteList.js
│   ├── store
│   │   └── index.js
│   ├── utils
│   │   ├── converter.js
│   │   └── request.js
│   └── views
│       ├── ESMap
│       │   └── esmap1.vue
│       ├── Map
│       │   ├── index.vue
│       │   └── main.vue
│       ├── account
│       │   └── AccountManagement.vue
│       ├── device
│       │   └── Devicedetails.vue
│       ├── error
│       │   ├── AppError403.vue
│       │   ├── AppError404.vue
│       │   └── AppError500.vue
│       ├── flow
│       │   └── FlowMonitoring.vue
│       ├── heatmap
│       │   ├── canvas.vue
│       │   └── index.vue
│       ├── index
│       │   └── index.vue
│       ├── layout
│       │   ├── TheLayout.vue
│       │   ├── TheLayoutEmpty.vue
│       │   ├── TheLayoutHeader.vue
│       │   ├── TheLayoutMain.vue
│       │   ├── TheLayoutSidebar.vue
│       │   ├── TheLayoutSubSidebar.vue
│       │   └── TheLayoutTags.vue
│       ├── log
│       │   ├── EquLog.vue
│       │   └── UserLog.vue
│       ├── login
│       │   ├── AppLogin.vue
│       │   └── AppRegister.vue
│       ├── pages
│       │   └── PageHome.vue
│       ├── topology
│       │   ├── Topo.vue
│       │   └── Topology.vue
│       └── user
│           ├── UserList.vue
│           └── Users.vue
├── vue.config.js
└── yarn.lock
```

后端结构树:

```
├── README.md
├── build
│   ├── Dockerfile
│   └── README.md
├── demo
│   ├── ClassStartMaginc.py
│   └── sysMessage.py
├── pkg
│   ├── __init__.py
│   ├── blueprints
│   │   ├── auth.py
│   │   ├── sdn.py
│   │   ├── server.py
│   │   ├── test
│   │   │   └── __init__.py
│   │   └── users.py
│   ├── crawlers
│   │   ├── base.py
│   │   ├── setting.py
│   │   ├── system
│   │   │   ├── get_all.py
│   │   │   ├── get_single.py
│   │   │   └── speeds.py
│   │   └── users
│   │       ├── get_err.py
│   │       ├── get_floor_device.py
│   │       ├── get_sites.py
│   │       ├── get_token.py
│   │       ├── get_user_location.py
│   │       ├── get_users.py
│   │       └── heatmap.py
│   ├── exceptions
│   │   ├── reqerror.py
│   │   └── token.py
│   ├── extensions.py
│   ├── models
│   │   └── __init__.py
│   ├── schemas
│   │   ├── __init__.py
│   │   ├── auth.py
│   │   ├── auth_test.py
│   │   └── sdn.py
│   ├── setting
│   │   ├── __init__.py
│   │   └── setting.py
│   ├── settings.py
│   ├── storages
│   │   └── redis.py
│   ├── util
│   │   ├── __init__.py
│   │   ├── stamp.py
│   │   └── time_parse.py
│   └── utils.py
├── request_demo
│   └── Python_demo.py
├── requirement.txt
└── wsgi.py
```