

本科实验报告

课程名称： B/S 体系软件设计

姓 名： 黄文杰

学 院： 计算机科学与技术学院

系： 计算机系

专 业： 软件工程

学 号： 3210103379

指导教师： 胡晓军

2023 年 10 月 21 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： B/S体系设计

实验项目名称： 物联网设备管理平台

学生姓名： 黄文杰 专业： 软件工程 学号： 3210103379

同组学生姓名： 无 指导老师： 胡晓军

实验地点： 曹光彪西-503 / 线上 实验日期： 2023 年 10 月 21 日

【物联网设备管理平台】

**设计文档**

负责人：黄文杰

日期：2023 年 10 月 21 日

1. **项目概述**
   1. **项目目的**

本项目的主要目的是利用Web开发技术实现一个物联网设备管理平台。该平台将允许用户注册、登录，并提供物联网设备的管理和数据统计功能。具体要求包括搭建MQTT服务器、用户认证、设备管理、数据查询和统计等功能。

* 1. **项目背景**

随着物联网（IoT）技术的快速发展，越来越多的物理设备和传感器连接到互联网，产生大量的数据。这些数据对于监测、控制和优化各种领域的设备和系统至关重要，包括智能城市、农业、医疗保健、工业制造和能源管理等。为了有效地管理和利用这些物联网设备和数据，物联网设备管理平台应运而生。

这样一个背景下，我们需要开发一个物联网设备管理平台，该平台将允许用户轻松地注册、连接、配置和监视物联网设备。它还将提供数据收集、存储和可视化工具，以便用户可以实时跟踪设备状态、查询历史数据、分析趋势和制定决策。

物联网设备管理平台的实施不仅可以提高设备的效率和性能，还可以帮助用户实现以下目标：

1. **实时监控和控制**：用户可以随时随地监测物联网设备的状态，并根据需要采取控制措施。
2. **数据分析和决策支持**：平台提供了数据分析工具，帮助用户分析历史数据、检测异常情况，从而支持决策制定。
3. **资源优化**：通过数据的实时监控和分析，用户可以更好地管理资源，提高效率，减少能源浪费和成本。
4. **安全性**：平台应具备安全性功能，以确保设备和数据的保护，防止未经授权的访问和攻击。

总之，物联网设备管理平台的开发将有助于实现更智能、高效和安全的设备管理，促进各种领域的创新和进步。该平台将成为物联网应用生态系统中的关键组成部分，为用户提供更好的管理和控制物联网设备的能力。

* 1. **设计要求**
     1. **基本功能**

1. **MQTT服务器搭建：**搭建一个MQTT服务器，能够接收指定的物联网终端模拟器发送的数据。
2. **用户注册和登录：**实现用户注册和登录功能。用户注册时需要填写必要的信息并进行验证，包括用户名、密码（要求至少6字符）、邮箱格式验证，并确保用户名和邮箱在系统中唯一。注册后的用户可以进行以下操作。
3. **用户信息修改：**已登录的用户可以在个人信息界面查看和修改自己的个人信息，如重置密码等。
4. **设备配置界面：**提供一个设备配置界面，用户可以创建或修改设备信息，包括设备ID、设备名称、设备类型等。
5. **设备上报数据查询统计界面：**提供设备上报数据的查询和统计界面。
6. **地图界面展示设备信息：**展示设备信息，并区分正常和告警信息。对于某些设备类型，历史数据可以展示成历史轨迹。
7. **首页统计信息：**在首页提供统计信息，包括设备总数、在线设备总数、接收的数据量等。使用图表方式展示，如柱状图、折线图等。
   * 1. **增强功能**

**手机端适配：**确保系统在手机浏览器和微信等内置浏览器中能够友好显示，适配不同屏幕尺寸。

* + 1. **安全需求**

1. **保密性**

**· 用于身份验证的用户名和密码应防止未经授权的用户访问系统。**

**· 应构建访问控制以防止合法用户非法使用系统资源。**

**· 某些敏感数据（如用户名，密码）在交换时应加密。密码在存储之前应加密。**

**· 在用户登录期间，应该防止 SQL 注入，密码强制破解和伪造会话入侵。**

1. **完整性**

**·防止非法用户对数据进行无意或恶意的修改、插入、删除，防止数据丢失。**

1. **约束性**

**·为数据库加上一定的约束，对关键性操作如删除、修改进行限制，并对用户进行警示。**

* + 1. **性能需求**

**·** 在高访问量时仍能保证并发运行（至少50人次的并发访问不会致使崩溃）

· 系统应保证运行稳定，避免出现崩溃

· 能保证在主流的浏览器能够正常运行

· 保证网页的加载时间不会过长

1. **技术选型**
   1. **技术栈**

**·前端开发**

**· 前端基础技术**：HTML/CSS/JavaScript

· **框架：**总体上采用Vue前端框架来构建前端项目

· **辅助技术：**Element UI（CSS样式库） Axios（HTTP 库） Echarts（数据可视化图表库）

· **移动端适配：**使用响应式设计或CSS框架实现手机端适配

·**后端开发**

**· 编程语言：**Java

· **Web框架：**Spring

· **数据库：**Mysql

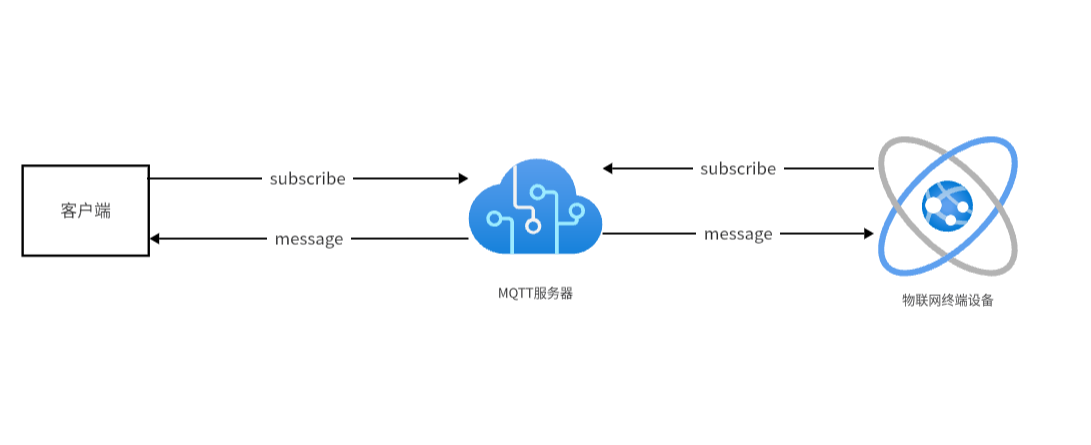
**· ORM框架：**Mybatis

· **MQTT服务器：**Mosquitto

· **用户认证：**使用JWT（JSON Web Tokens）进行用户认证

· **开发框架：**Spring Boot

1. **架构设计**
   1. **总体结构图**



* 1. **前端架构**
     1. **前端技术总览**

**·Vue：**作为前端主要框架，用于构建用户界面。

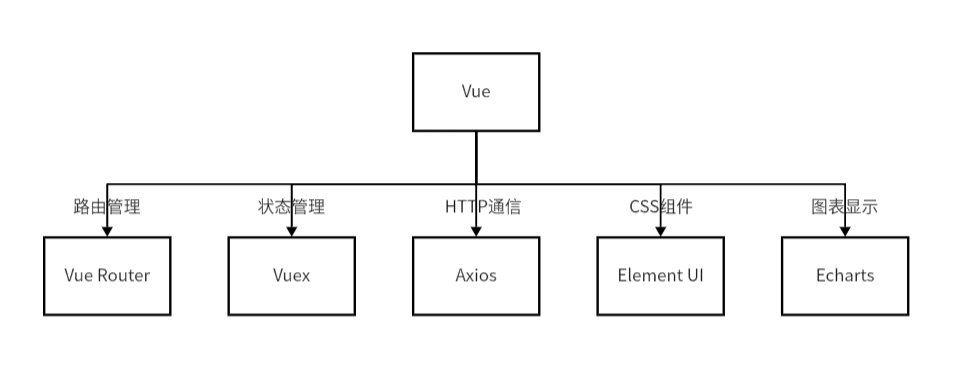
·**Vue Router：**用于处理前端路由。

·**Vuex：**用于状态管理（存储全局信息）。

·**Axios：**用于与后端API进行通信。

·**Element UI：**用于UI组件和响应式设计（CSS样式）。

·**Echarts：**用于绘制统计图表。



* + 1. **前端模块**

1. **用户认证模块**

**· 用户注册：**提供注册界面，要求用户输入用户名、密码、邮箱等信息。前端通过API发送注册请求到后端。

**· 用户登录：**提供登录界面，用户输入用户名和密码进行登录。前端通过API发送登录请求到后端，并在成功登录后接收和存储JWT令牌。

**· JWT 管理：**前端需要存储和管理JWT令牌，以便在每个后续请求中进行用户身份验证。

1. **设备管理模块**

**· 设备列表：**展示用户拥有的设备列表，包括设备ID、设备名称、设备类型等信息。用户可以搜寻设备进行编辑。

**· 创建设备：**提供界面以创建新设备，包括设备ID、设备名称、设备类型等。前端通过API发送创建请求到后端。

**· 设备编辑：**用户可以编辑设备信息，如名称和类型。前端通过API发送编辑请求到后端。

**· 删除设备：**用户可以删除设备，前端通过API发送删除请求到后端。

1. **数据查询和可视化模块**

**· 数据查询：**用户可以选择设备并查询设备上报的数据。前端发送查询请求到后端，后端返回相应的数据。

**· 数据可视化：**使用数据可视化库（Echart库），将查询到的数据绘制成图表，以帮助用户分析和理解数据。

1. **统计信息模块**

**· 首页统计信息：**在首页上展示统计信息，包括设备总数、在线设备总数、接收的数据量等。使用数据可视化库来绘制图表。

1. **地图展示模块**

**· 地图界面：**展示设备信息，包括设备位置。使用地图API来标记设备的位置，区分正常和告警设备。

* 1. **后端架构**
     1. **后端技术总览**

**· Spring Boot：**作为后端的主要框架，用于响应和处理服务请求。

**·** **Spring Security：**用于用户认证和授权。

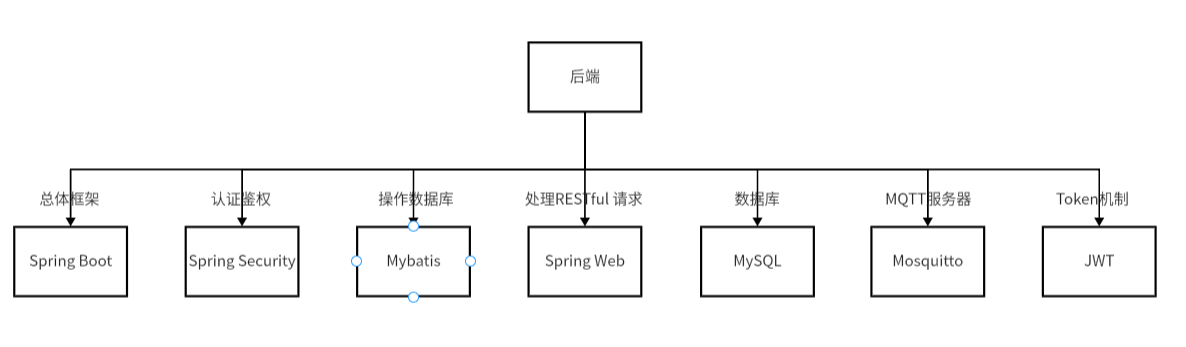
**·** **Mybatis：**用于与数据库交互。

**·** **Spring Web：**用于构建RESTful API。

**· MySQL数据库：**用于存储用户信息、设备信息和数据记录。

**· Mosquitto 或其他MQTT服务器：**用于接收设备数据。

**· JWT（JSON Web Tokens）：**用于用户身份验证。



* + 1. **后端模块**

1. **用户认证和授权模块**

**· 用户认证：**使用Spring Security进行用户认证和授权。处理用户注册、登录和颁发JWT令牌。

**· JWT管理：**生成和验证JWT令牌，确保用户身份的安全性和可信度。

1. **设备管理模块**

**· 设备管理：**使用Spring Data JPA管理设备信息的创建、编辑和删除。

· **设备信息API：**提供RESTful API以支持前端设备管理操作。

1. **数据查询和统计模块**

**· 数据查询API：**提供RESTful API以支持前端数据查询操作。

**· 统计信息API：**提供RESTful API以获取设备统计数据。

**（4）MQTT模块**

**· MQTT服务器：**使用Mosquitto，接收设备数据并将其存储到数据库中。

1. **数据库设计**

基于项目要求，需要设计数据库表来存储用户信息、设备配置信息和消息记录。

* 1. **数据表设计**
     1. **User表**

**·**该表用来保存用户的相关信息

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **描述** | **备注** |
| id | INT | 用户的id | 主键、非空、自增 |
| username | VARCHAR(255) | 用户名 | 非空，且唯一 |
| password | VARCHAR(255) | 用户的密码 | 非空，加密存储 |
| email | VARCHAR(255) | 用户的邮箱 | 非空，且唯一 |
| phone | VARCHAR(15) | 用户的手机号 | 可以为空 |

* + 1. **Device表**

**·**该表用来保存设备的相关信息

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **描述** | **备注** |
| device\_id | VARCHAR(15) | 设备的id | 主键、非空、自增 |
| user\_id | INT | 关联的用户ID | 非空，外键（参考用户数据表） |
| device\_name | VARCHAR(255) | 设备名称 | 非空 |
| device\_type | INT | 设备类型 | 非空，按照整数来映射相应的设备类型（1:智能家居 2:生命安全设备 3:车载设备 4:可穿戴设备 5:工业智能网关 6：其他） |
| device\_description | VARCHAR(255) | 设备描述 | 非空 |
| registration\_time | DATETIME | 设备注册日期 | 非空 |
| last\_update | DATETIME | 设备信息上次更新日期 | 非空 |
| is\_active | BOOLEAN | 设备是否处于活动状态 | 非空（0:离线；1:活跃） |

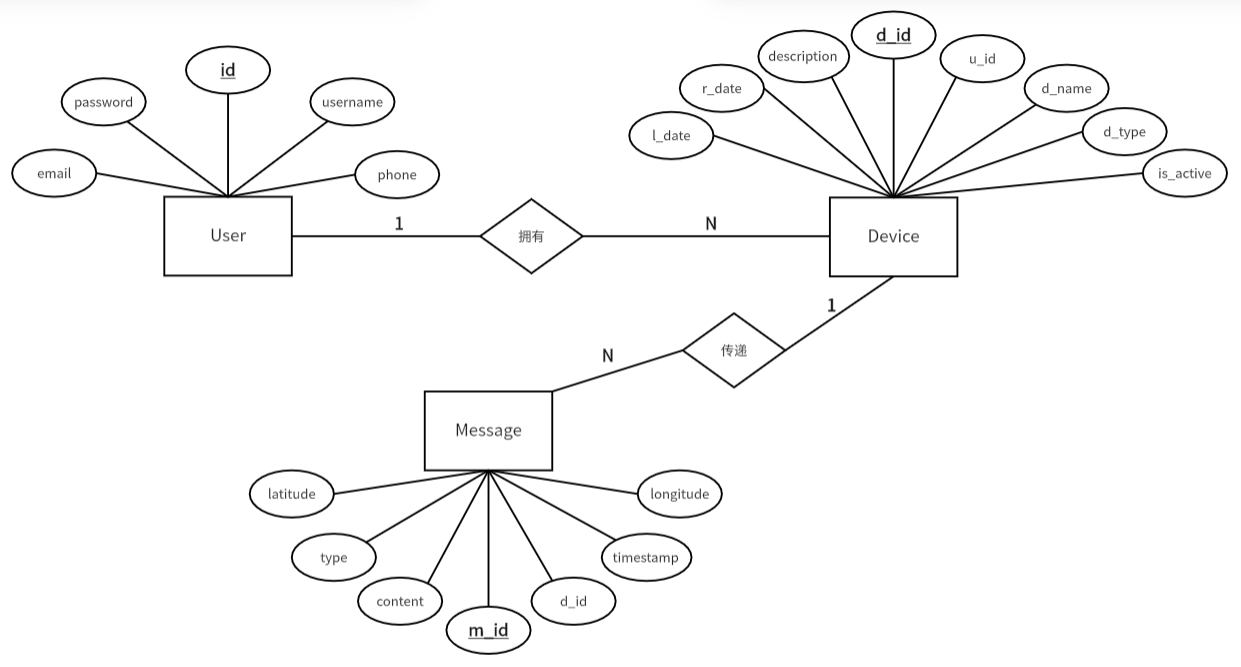
* + 1. **Message表**

**·**该表用来保存和设备相关的信息（与mqtt服务器交互，此表后期可能会有所改动，仅供参考）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **描述** | **备注** |
| message\_id | INT | 信息记录的唯一标识符 | 非空，主键，自增 |
| device\_id | VARCHAR(15) | 关联的设备ID | 非空，外键（参考设备数据表） |
| timestamp | DATETIME | 信息记录的时间戳 | 非空 |
| message\_type | INT | 区分正常信息和异常信息 | 0: 正常  1: 异常 |
| message\_content | VARCHAR(255) | 信息内容 | 非空 |
| latitude | DECIMAL(10, 6) | 设备的纬度信息 | 非空 |
| longitude | DECIMAL(10, 6) | 设备的经度信息 | 非空 |
| value | INT | 设备数据 | 非空 |

* 1. **ER 图**

（在该ER图中，为了方便起见，将某些属性值进行了略写，例如用d\_id来代替了device\_id）



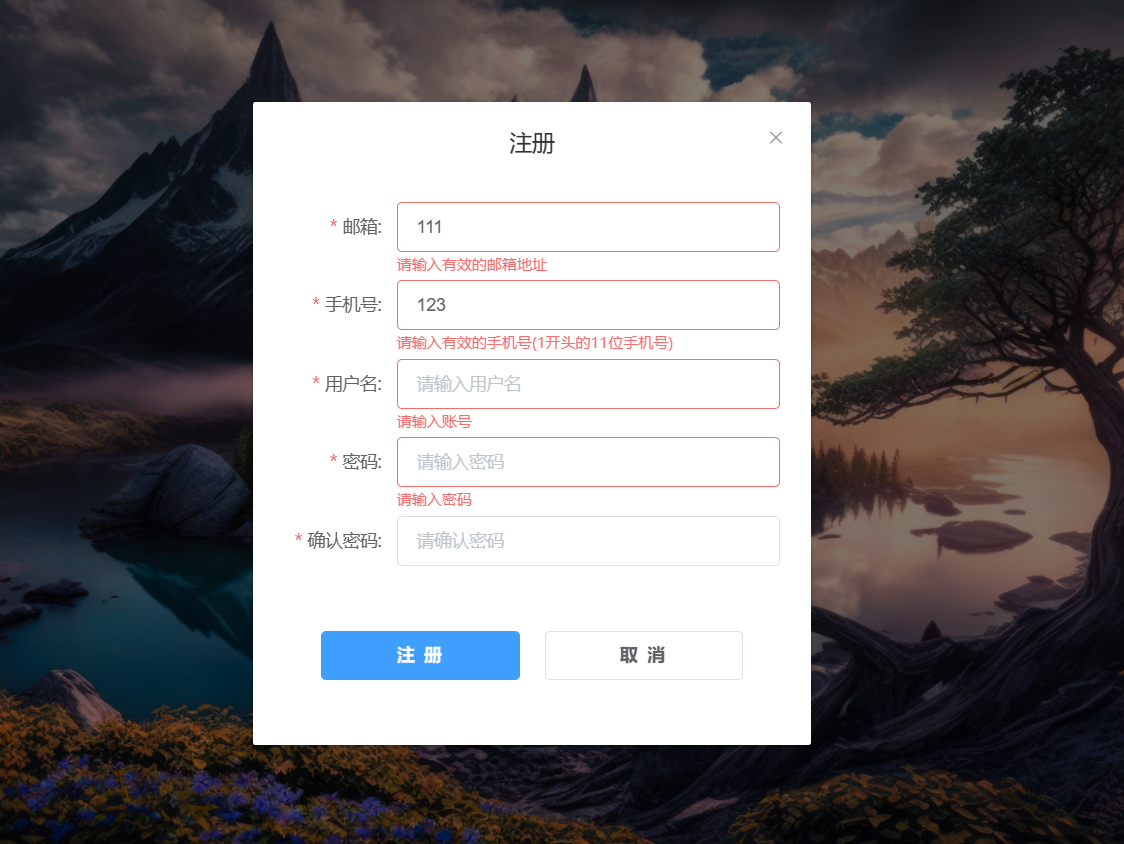
1. **用户界面设计**

此处提供的界面为项目完成之前设计的界面（部分界面可能会与实际成果出入较大，所以这里仅供参考）。

* 1. **登录界面**



* 1. **注册界面**

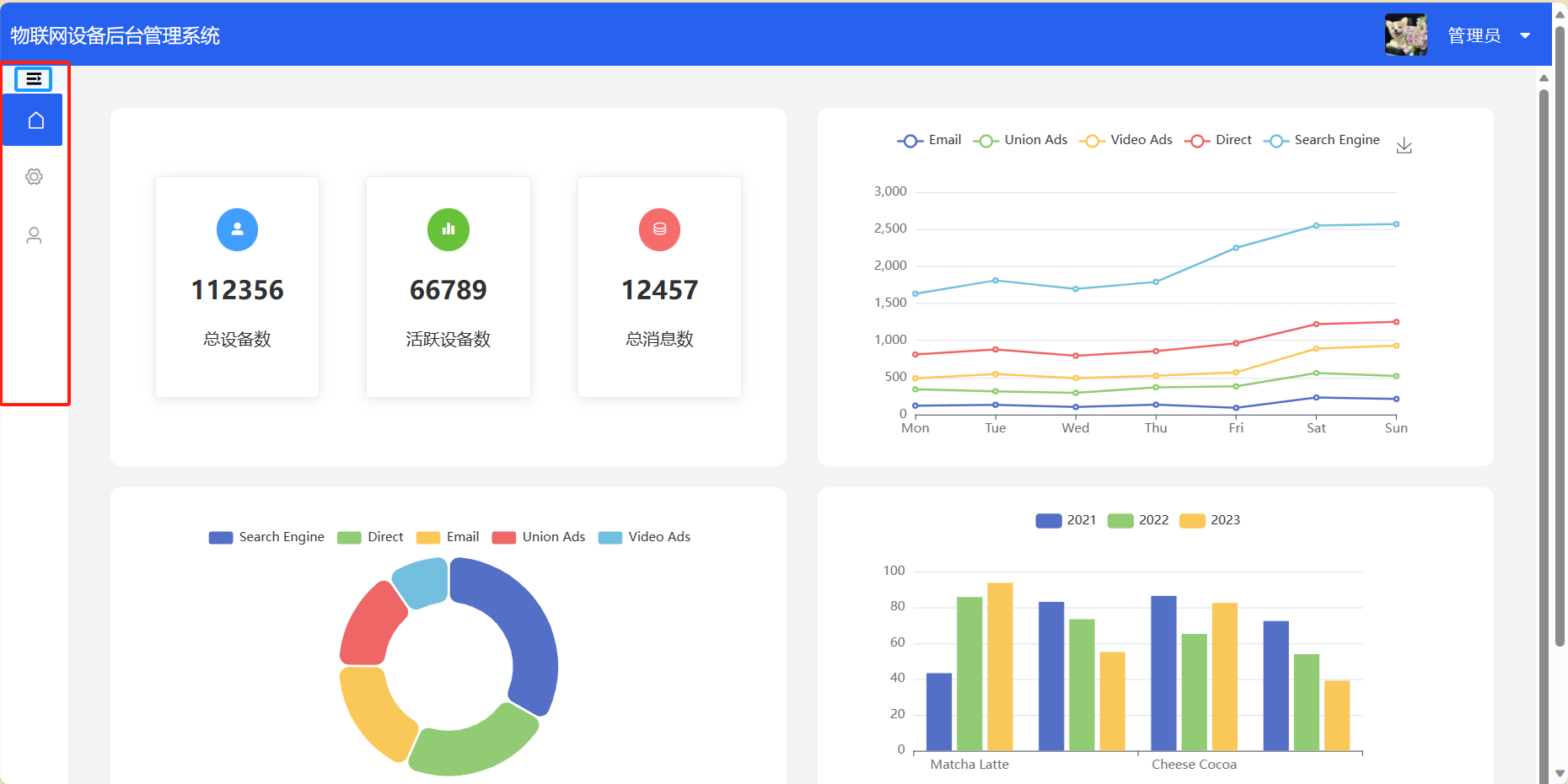


* 1. **主页**

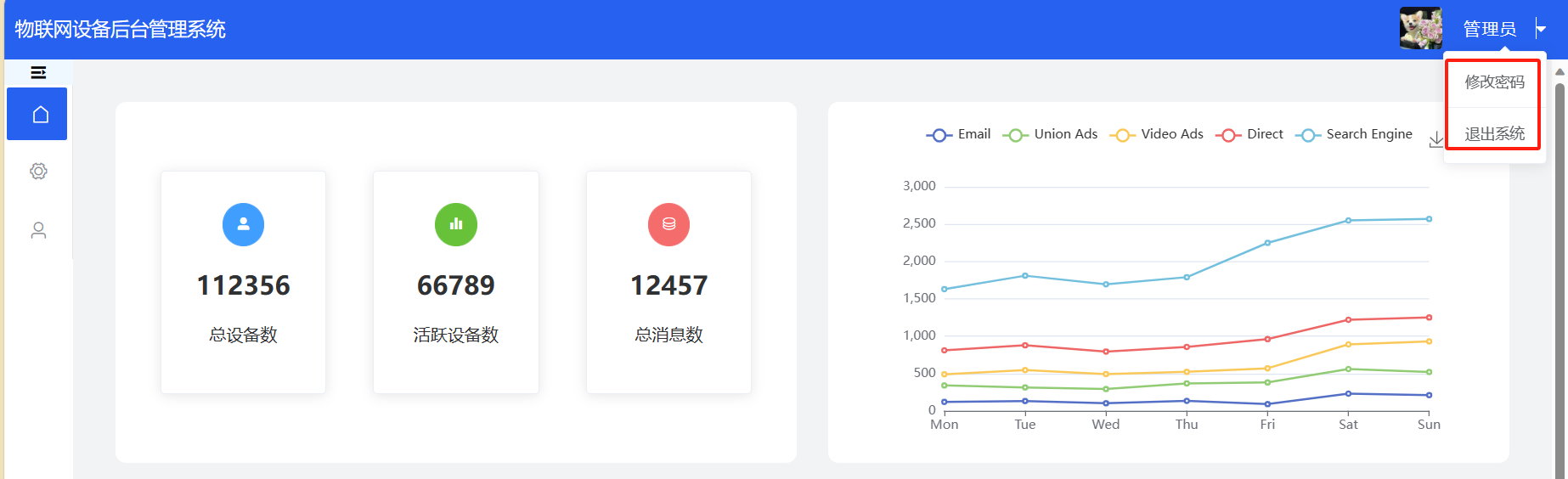
主页主要用来展示一些信息的总览情况（其中具体细节将在项目后期进行确认，这里只提供一个大致的界面示意图）。



主页还提供了侧边栏收缩的功能（点击蓝色框里的按钮即可实现）



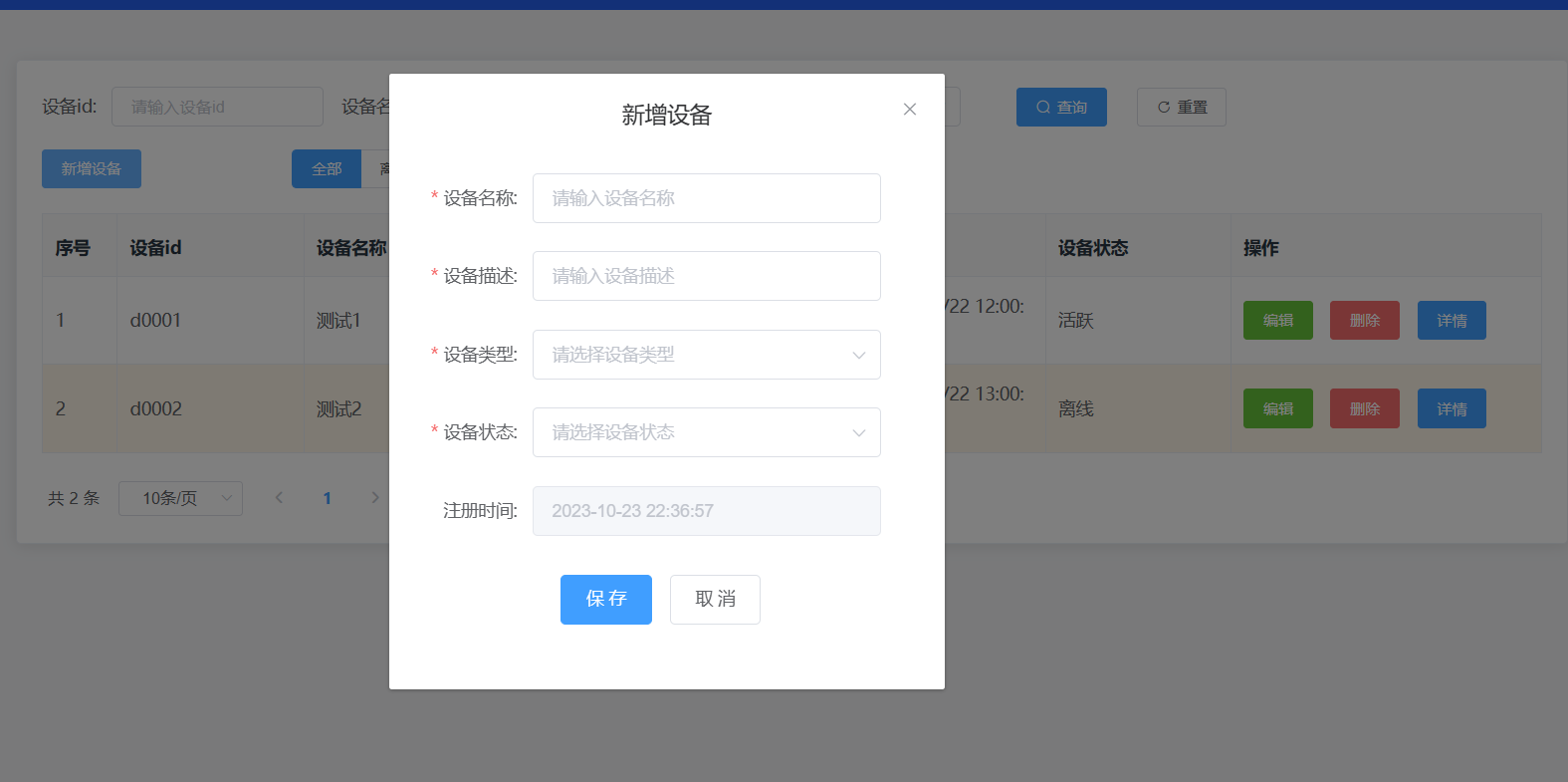
头部导航栏有一个下拉按钮，提供修改密码和退出系统的功能



* 1. **设备配置页面**
     1. **总体布局**



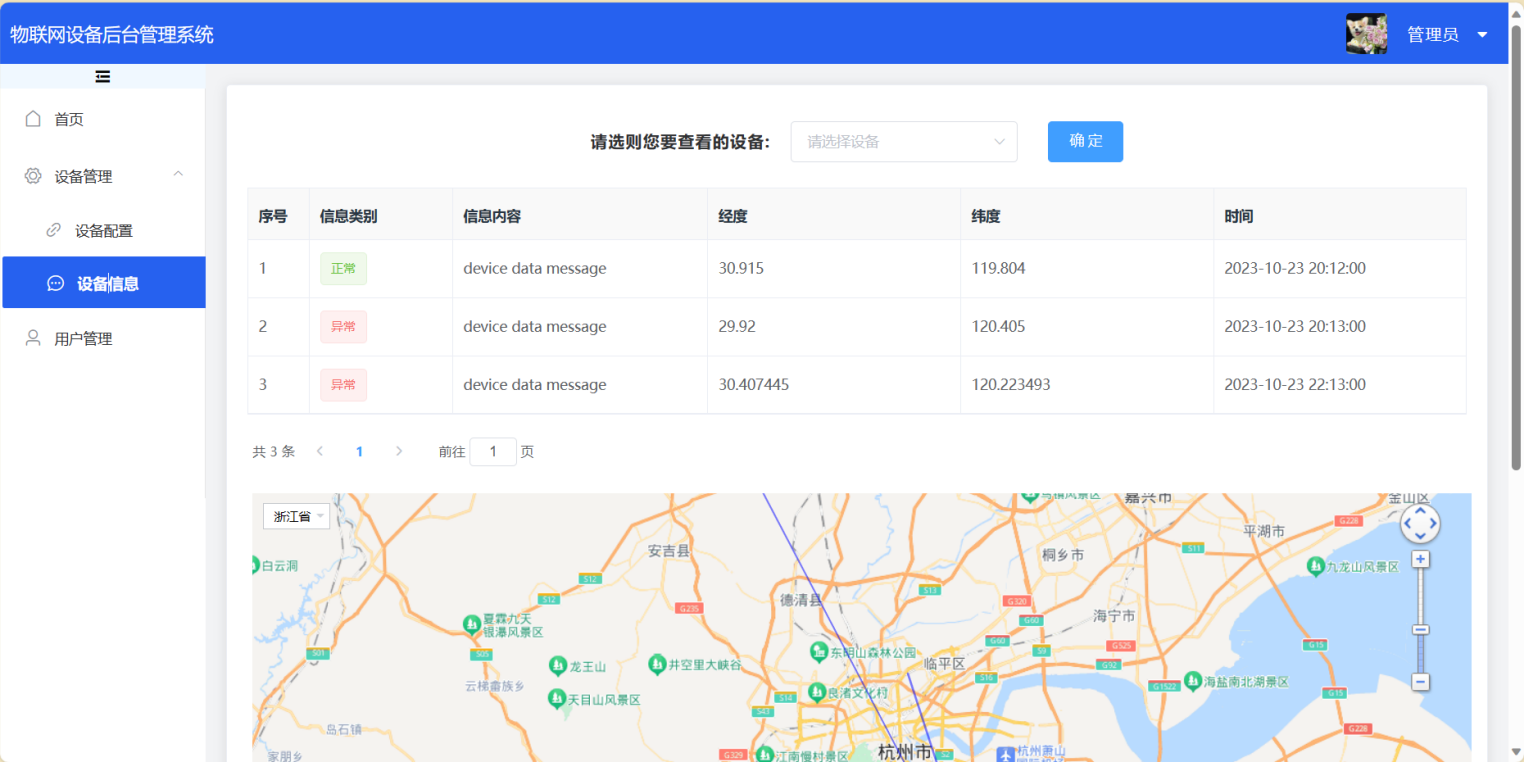
* + 1. **新增设备**

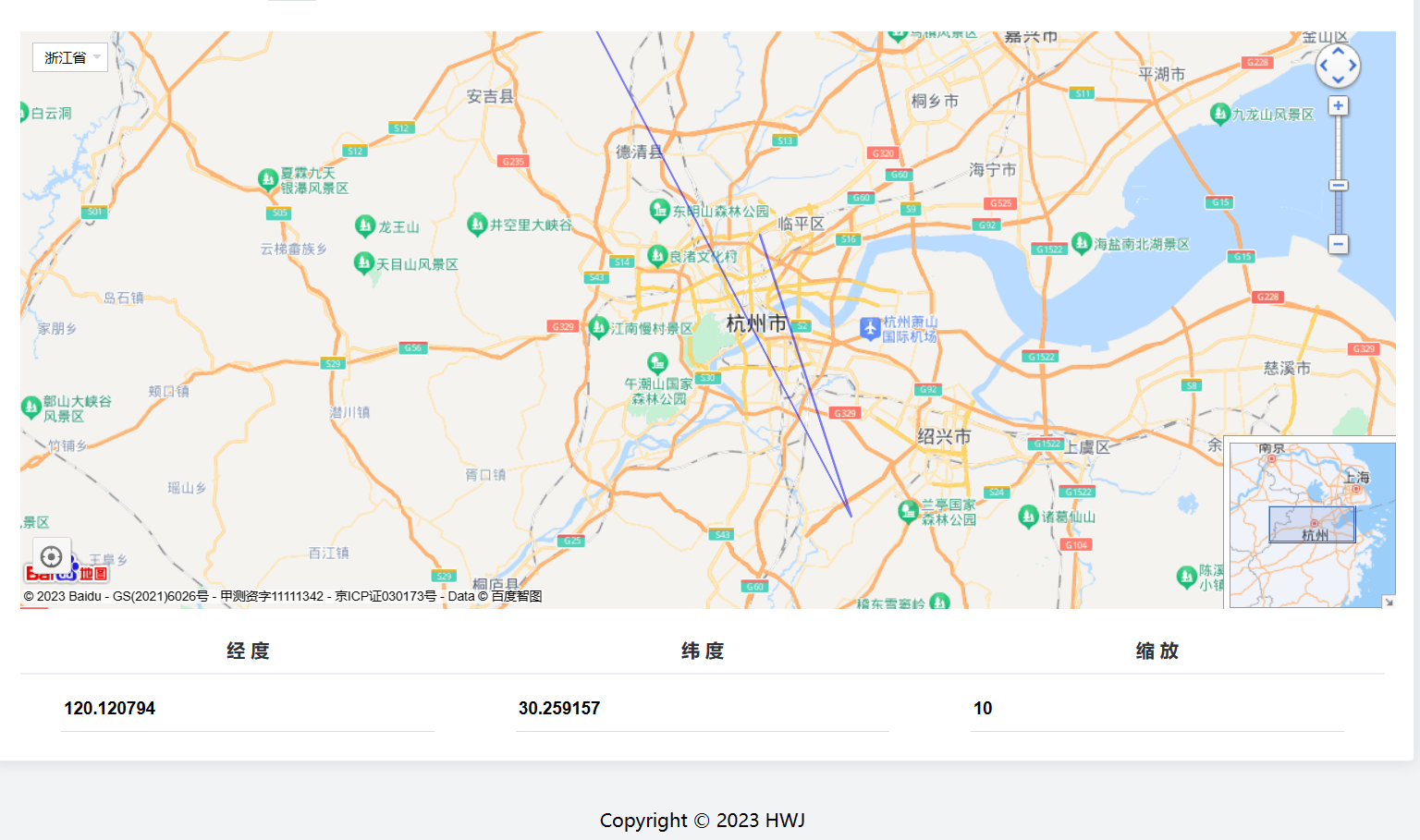


* + 1. **编辑设备**

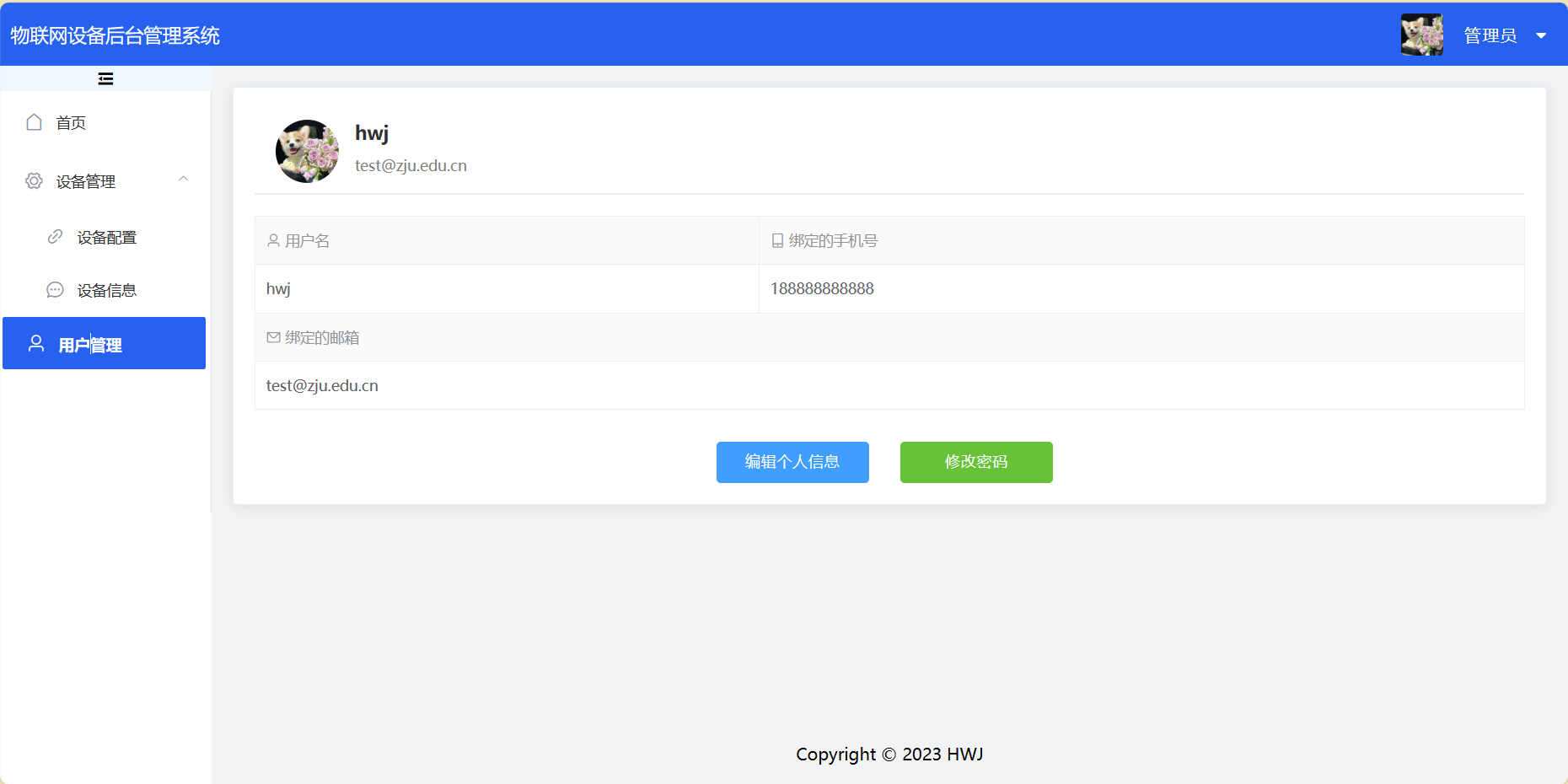


* 1. **设备信息页面**





* 1. **用户管理页面**
     1. **总体布局**



* + 1. **编辑个人信息页面**



* + 1. **修改密码页面**



1. **接口设计**

在该项目中，前端采用Axios 库与后端数据请求和交互，后端通过设计若干RESTful 的增删查改 api 供前端调用，其中具体的接口设计如下（此部分在后期可能会有修改，这里仅供参考）：

* 1. **用户相关接口**

用户相关的接口主要提供登录、注册、对User表进行增删改查的功能，其的接口 URL 均以 /user 开头。

* + 1. **用户登录**

|  |  |
| --- | --- |
| **请求路径** | /user/login |
| **请求方法** | POST |
| **请求参数** | ·username (String) - 用户名  ·password (String) - 密码 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·message (String) - 操作结果消息  ·user\_id (Int) - 用户的id  ·username (String) - 用户名  ·email (String) - 邮箱  ·phone (String) - 电话号码  ·token (String) - JWT令牌 |
| **接口简介** | 用户通过提供用户名和密码进行身份验证，成功登录后返回用户信息，并将token存储在客户端 |

* + 1. **用户注册**

|  |  |
| --- | --- |
| **请求路径** | /user/register |
| **请求方法** | POST |
| **请求参数** | ·username (String) - 用户名  ·password (String) - 密码  ·email (String) - 邮箱  ·phone (String) - 电话号码 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·message (String) - 操作结果消息  ·user\_id (Int) - 用户的id  ·username (String) - 用户名  ·email (String) - 邮箱  ·phone (String) - 电话号码 |
| **接口简介** | 用户通过提供用户名、密码、邮箱和电话号码进行注册，成功注册后返回用户信息以及生成的JWT令牌（前端实现自动跳转） |

* + 1. **修改密码**

|  |  |
| --- | --- |
| **请求路径** | /user/updatePassword |
| **请求方法** | POST |
| **请求参数** | **Header：**  "Authorization" : token  **Body:**  ·username (String) - 用户名  ·old\_password (String) - 旧密码  ·new\_password (String) - 新密码 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·message (String) - 操作结果消息 |
| **接口简介** | 用户通过提供JWT令牌以及旧密码和新密码来修改密码。 |

* + 1. **编辑个人信息**

|  |  |
| --- | --- |
| **请求路径** | /user/editUserInfo |
| **请求方法** | POST |
| **请求参数** | **Header：**  "Authorization" : token  **Body:**  ·user\_id (Int) - 用户的id  ·new\_username (String) - 新用户名  ·new\_email (String) - 新邮箱（可为空）  ·new\_phone (String) - 新手机号（可为空） |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·message (String) - 操作结果消息 |
| **接口简介** | 用户通过提供JWT令牌和用户id来修改用户名、手机号和邮箱。 |

* + 1. **根据用户id获取个人信息**

|  |  |
| --- | --- |
| **请求路径** | /user/getUserInfo |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·user\_id (Int) - 用户的id |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·message (String) - 操作结果消息  ·user\_id (Int) - 用户的id  ·username (String) - 用户名  ·email (String) - 邮箱  ·phone (String) - 电话号码  ·token (String) - JWT令牌 |
| **接口简介** | 用户通过提供JWT令牌和用户id来重新获取用户个人信息（一般是隐式调用，多用在修改个人信息之后） |

* 1. **设备相关接口**

设备相关的接口主要提供对设备配置信息的增删查改功能，这一类接口的 URL 都

用/device 开头。

* + 1. **获取某个设备信息**

|  |  |
| --- | --- |
| **请求路径** | /device/{device\_id} |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·device\_id (String) - 目标设备的id |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·device\_id (String) - 设备的id  ·user\_id (Int) - 用户的id  ·device\_name (String) - 设备名称  ·device\_type (Int) - 设备类型  ·device\_description (String) - 设备描述  ·registration\_time (Date) - 设备注册日期  ·last\_update (Date) - 设备信息上次更新日期  ·is\_active (Boolean) - 设备是否处于活动状态 |
| **接口简介** | 获取特定设备的详细信息。 |

* + 1. **查询用户设备列表**

|  |  |
| --- | --- |
| **请求路径** | /device/user/{user\_id} |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·user\_id (Int) - 用户的id |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·devices (Array) - 包含设备信息的数组  ·device\_id (String) - 设备的id  ·device\_name (String) - 设备名称  ·device\_type (Int) - 设备类型  ·device\_description (String) - 设备描述  ·registration\_time (Date) - 设备注册日期  ·is\_active (Boolean) - 设备是否处于活动状态  ·countList(Array) - 包含每种设备数量的数组 |
| **接口简介** | 获取特定用户拥有的设备列表。 |

* + 1. **条件查询设备列表**

|  |  |
| --- | --- |
| **请求路径** | /device/search-with-pagination |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·user\_id (Int) - 用户的id（不为空）  ·device\_id (String) - 设备的id (可为空)  ·device\_name (String) - 设备名称(可为空)  ·device\_type (Int) - 设备类型(可为空)  ·is\_active (Int) - 设备状态（可为空）  ·current (Int) - 当前页  ·size (Int) - 每页的大小 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·devices (Array) - 包含设备信息的数组  ·device\_id (String) - 设备的id  ·device\_name (String) - 设备名称  ·device\_type (Int) - 设备类型  ·device\_description (String) - 设备描述  ·registration\_time (Date) - 设备注册日期  ·is\_active (Boolean) - 设备是否处于活动状态  ·totalCount (Int) - 符合条件的数据总数  ·currentPage (Int) - 当前页  ·pageSize (Int) - 页的大小 |
| **接口简介** | 根据查询表单的相关字段和用户ID的值搜索符合条件设备列表数据（注意：表单中所有字段均可为空，后端需要作相应处理，为NULL或者空字符串的字段不作为查询条件，要考虑分页处理）。同时，根据设备名称搜索可以搜索到相近的结果（不要求完全匹配而是like） |

* + 1. **新增设备**

|  |  |
| --- | --- |
| **请求路径** | /device/add |
| **请求方法** | POST |
| **请求参数** | **Header：**  "Authorization" : token  **Body:**  ·user\_id (Int) - 用户的id  ·device\_name (String) - 设备名称  ·device\_type (Int) - 设备类型  ·device\_description (String) - 设备描述  ·is\_active (Boolean) - 设备是否处于活动状态  ·registration\_time (Date) - 设备注册日期 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·message (String) - 操作结果消息 |
| **接口简介** | 创建一个新设备。（注意，因为是新增的设备，所以后端需要自动将registration\_time的值赋给last\_update） |

* + 1. **修改设备配置**

|  |  |
| --- | --- |
| **请求路径** | /device/update/{device\_id} |
| **请求方法** | PUT |
| **请求参数** | **Header：**  "Authorization" : token  **Body:**  ·device\_name (String) - 修改后的设备名称  ·device\_type (Int) - 修改后的设备类型  ·device\_description (String) - 修改后的设备描述  ·last\_update (Date) - 设备信息上次更新日期  ·is\_active (Boolean) - 修改后的设备活动状态 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  · message (String) - 操作结果消息 |
| **接口简介** | 修改特定设备的配置信息。（传入的参数可以为空，后端需要做特殊处理） |

* + 1. **获取最近七天新增设备数量**

|  |  |
| --- | --- |
| **请求路径** | /device/new-devices-count |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·user\_id (Int) - 用户的id  ·today (Date) - 当天的日期 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·counts (Array) - 包含最近七天每天新增设备数量的数组  ·date (Date) - 日期  ·count (Int) - 该日期新增的设备数量 |
| **接口简介** | 获取当前用户最近七天内每天新增设备的数量。 |

* + 1. **删除设备**

|  |  |
| --- | --- |
| **请求路径** | /device/delete/{device\_id} |
| **请求方法** | DELETE |
| **请求参数** | **Header：**  "Authorization" : token |
| **响应数据** | ·status (Int) - 响应码（响应状态）  · message (String) - 操作结果消息 |
| **接口简介** | 删除相应的设备 |

* 1. **设备信息相关接口**

设备消息相关接口主要提供对Message表的增删改查功能，其接口的URL均以/message开头。

* + 1. **查询当前用户所有设备的总消息数**

|  |  |
| --- | --- |
| **请求路径** | /message/total-count |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·user\_id (Int) - 用户的id |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·total\_count (Int) - 所有设备的总消息数量 |
| **接口简介** | 获取当前用户所有设备的消息总数。 |

* + 1. **查询设备的历史轨迹**

|  |  |
| --- | --- |
| **请求路径** | /message/history-track/{device\_id} |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·device\_id (String) - 目标设备的id |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·tracks (Array) - 包含设备历史轨迹信息的数组  ·timestamp (Date) - 时间戳  ·latitude (Decimal) - 纬度  ·longitude (Decimal) - 经度 |
| **接口简介** | 获取特定设备的历史轨迹信息。 |

* + 1. **查询某一设备的历史消息**

|  |  |
| --- | --- |
| **请求路径** | /message/device-history/{device\_id} |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·device\_id (String) - 目标设备的id |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·messages (Array) - 包含设备历史消息的数组  ·message\_id (Int) - 消息的id  ·timestamp (Date) - 时间戳  ·message\_type (String) - 消息类型  ·message\_content (String) - 消息内容  ·latitude (Decimal) - 纬度  ·longitude (Decimal) - 经度 |
| **接口简介** | 获取特定设备的历史消息记录。 |

* + 1. **查询最近七天接受的消息数**

|  |  |
| --- | --- |
| **请求路径** | /message/received-count |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·user\_id (Int) - 用户的id  ·today (Date) - 当天的日期 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·counts (Array) - 包含最近七天内每天接收的消息数的数组  ·date (Date) - 日期  ·count (Int) - 该日期接收的消息数量  ·normalCount (Int) - 该日期收到正常消息的数量  ·abnormalCount (Int) - 该日期收到不正常的消息数量 |
| **接口简介** | 获取用户所属设备最近七天内每天接收的消息数量 |