浙江大学

ZHEJIANG UNIVERSITY

设计报告

物联网应用平台

沈乐明　3180103654　软件工程

浙江大学 网络空间安全学院

指导教师：胡晓军

目录

1 引言

1.1 编写目的

本设计报告对物联网应用平台作了具体详细的设计分析。

本文档主要为使软件产品和软件项目满足规定的软件规格要求而确定软件系统的体系结构、组成部分、数据组织、模块、内外部接口。主要任务有:

- 建立软件产品和软件项目目标系统的总体结构
- 总体设计
- 接口设计
- 运行设计
- 系统数据结构设计
- 系统异常处理设计

本文档的预期读者有用户（拥有各种物联网终端设备的人员等），项目经理，开发人员以及跟该项目相关的其他竞争人员和无关人员。

1.2 项目背景

本项目开发的软件为一个物联网应用平台。

自 21 世纪，我们的生活就全面迈入了全新信息化时代，尤其是数字产业的发展，我们的生活离不开各行各业的数据。而物联网作为新一代信息技术的重要组成部分，其用户端延伸和扩展到了任何物品与物品之间，通过射频识别、红外感应器、全球定位系统、激光扫描器等信息传感设备，按约定的协议，把任何物品与互联网相连接，进行信息交换和通信，以实现对物品的智能化识别、定位、跟踪、监控和管理的一种网络。

物联网的应用领域涉及到方方面面，在工业、农业、环境、交通、物流、安保等基础设施领域的应用，有效的推动了这些方面的智能化发展，使得有限的资源更加合理的使用分配，从而提高了行业效率、效益。 在家居、医疗健康、教育、金融与服务业、旅游业等与生活息息相关的领域的应用，从服务范围、服务方式到服务的质量等方面都有了极大的改进，大大的提高了人们的生活质量; 在涉及国防军事领域方面，虽然还处在研究探索阶段，但物联网应用带来的影响也不可小觑，大到卫星、导弹、飞机、潜艇等装备系统，小到单兵作战装备，物联网技术的嵌入有效提升了军事智能化、信息化、精准化，极大提升了军事战斗力，是未来军事变革的关键。

2020 年 5 月 7 日，工信部发布工信厅通信〔2020〕25 号文，即《工业和信息化部办公厅关于深入推进移动物联网全面发展的通知》。通知中首先表示：移动物联网（基于蜂窝移动通信网络的物联网技术和应用）是新型基础设施的重要组成部分。为贯彻落实党中央、国务院关于加快5G、物联网等新型基础设施建设和应用的决策部署，加速传统产业数字化转型，有力支撑制造强国和网络强国建设。

在这样一个大背景下，个人的物联网终端设备越来越多，管理也越来越麻烦，因此一款基于 Web 平台的物联网终端管理网站亟待开发。本物联网应用平台可以实现单用户物联网终端的有效管理、信息监控与在线跟踪等功能。

1.3 相关定义

- HTML
  HTML（Hyper Text Markup Language）称为超文本标记语言，是一种标识性的语言。它包括一系列标签．通过这些标签可以将网络上的文档格式统一，使分散的 Internet 资源连接为一个逻辑整体。HTML 文本是由 HTML 命令组成的描述性文本，HTML 命令可以说明文字，图形、动画、声音、表格、链接等。
- CSS
  层叠样式表（Cascading Style Sheets）是一种用来表现 HTML（标准通用标记语言的一个应用）或 XML（标准通用标记语言的一个子集）等文件样式的计算机语言。CSS 不仅可以静态地修饰网页，还可以配合各种脚本语言动态地对网页各元素进行格式化。
- UML
  统一建模语言（Unified Modeling Language）是一种为面向对象系统的产品进行说明、可视化和编制文档的一种标准语言，是非专利的第三代建模和规约语言。UML 是面向对象设计的建模工具，独立于任何具体程序设计语言。
- B/S 系统
  B/S 结构（Browser/Server，浏览器/服务器模式），是 WEB 兴起后的一种网络构模式，WEB 浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。客户机上只要安装一个浏览器如 Netscape Navigator 或 Internet Explorer，服务器安装 SQL Server、Oracle、MYSQL 等数据库。浏览器通过 Web Server 同数据库进行数据交互。
- Node.js
  Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。 Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型。Node 是一个让 JavaScript 运行在服务端的开发平台，它让 JavaScript 成为与 PHP、Python、Perl、Ruby 等服务端语言平起平坐的脚本语言。发布于 2009 年 5 月，由 Ryan Dahl 开发，实质是对 Chrome V8 引擎进行了封装。Node 对一些特殊用例进行优化，提供替代的 API，使得 V8 在非浏览器环境下运行得更好。V8 引擎执行 JavaScript 的速度非常快，性能非常好。Node 是一个基于 Chrome JavaScript 运行时建立的平台， 用于方便地搭建响应速度快、易于扩展的网络应用。Node 使用事件驱动， 非阻塞 I/O 模型而得以轻量和高效，非常适合在分布式设备上运行数据密集型的实时应用。
- React
  React 是用于构建用户界面的 JavaScript 库，起源于 Facebook 的内部项目，因为该公司对市场上所有 JavaScript MVC 框架，都不满意，就决定自己写一套，用来架设 Instagram 的网站。做出来以后，发现这套东西很好用，就在 2013 年 5 月开源了。具有声明式设计：React 采用声明范式，可以轻松描述应用；高效：React 通过对 DOM

的模拟，最大限度地减少与 DOM 的交互；灵活：React 可以与已知的库或框架很好地配合等特点。

- Ajax
  Ajax 即 Asynchronous JavaScript And XML（异步 JavaScript 和 XML）在 2005 年被 Jesse James Garrett 提出的新术语，用来描述一种使用现有技术集合的“新”方法，包括: HTML 或 XHTML, CSS, JavaScript, DOM, XML, XSLT, 以及最重要的 XMLHttpRequest。使用 Ajax 技术网页应用能够快速地将增量更新呈现在用户界面上，而不需要重载（刷新）整个页面，这使得程序能够更快地回应用户的操作。
- Django
  Django 是一个开放源代码的 Web 应用框架，由 Python 写成。采用了 MTV 的框架模式，即模型 M，视图 V 和模版 T。它最初是被开发来用于管理劳伦斯出版集团旗下的一些以新闻内容为主的网站的，即是 CMS（内容管理系统）软件。并于 2005 年 7 月在 BSD 许可证下发布。这套框架是以比利时的吉普赛爵士吉他手 Django Reinhardt 来命名的。2019 年 12 月 2 日，Django 3. 0 发布。

1.4 参考文献

- Early Approach to Software Engineering, Pallavi Gore, Kritika Saxena.
- Practical File of Software Engineering and Testing Laboratory, Aakash Raj.
- Software Engineering, Principles and Practice, 3rd Edition, Hans van Vliet.
- Program Manager’s Guidebook for Software Assurance, Dr. Kenneth E. Nidiffer, Timothy A. Chick, Dr. Carol Woody.
- Experimentation in Software Engineering, Claes Wohlin, Per Runeson, Martin Host, Magnus C. Ohlsson, Bjorn Regnell, Anders Wesslen.
- IEEE Computer Society/Software Engineering Institute Software Process Achievement (SPA) Award 2009, Satyendra Kumar, Ramakrishnan M.
- Michael Felderer, Wilhelm Hasselbring, Rick Rabiser, Reiner Jung: Software Engineering 2020, Fachtagung des GI-Fachbereichs Softwaretechnik, 24.-28. Februar 2020, Innsbruck, Austria. LNI P-300, Gesellschaft für Informatik e.V. 2020, ISBN 978-3-88579-694-7.
- Regina Hebig, Robert Heinrich: Combined Proceedings of the Workshops at Software Engineering 2020 Co-located with the German Software Engineering Conference 2020 (SE 2020), Innsbruck, Österreich, March 05, 2020. CEUR Workshop Proceedings 2581, CEUR-WS.org 2020.
- Steffen Becker, Ivan Bogicevic, Georg Herzwurm, Stefan Wagner: Software Engineering and Software Management, SE/SWM 2019, Stuttgart, Germany, February 18-22, 2019. LNI P-292, GI 2019, ISBN 978-3-88579-686-2.
- Stephan Krusche, Kurt Schneider, Marco Kuhrmann, Robert Heinrich, Reiner Jung, Marco Konersmann, Eric Schmieders, Steffen Helke, Ina Schaefer, Andreas Vogelsang, Björn Annighöfer, Andreas Schweiger, Marina Reich, André van Hoorn: Proceedings of the Workshops of the Software Engineering Conference 2019, Stuttgart, Germany, February 19, 2019. CEUR Workshop Proceedings 2308, CEUR-WS.org 2019.
- Peter Liggesmeyer, Gregor Engels, Jürgen Münch, Jörg Dörr, Norman Riegel: Software Engineering 2009: Fachtagung des GI-Fachbereichs Softwaretechnik 02.-06.03. 2009 in Kaiserslautern. LNI P-143, GI 2009, ISBN 978-3-88579-237-6.

- Jürgen Münch, Peter Liggesmeyer: Software Engineering 2009 - Workshopband, Fachtagung des GI-Fachbereichs Softwaretechnik 02.-06.03.2009 in Kaiserslautern. LNI P-150, GI 2009, ISBN 978-3-88579-244-4.
- 《软件工程——实践者的研究方法》，Roger S.Pressman，机械工业出版社
- 《软件需求（第三版）》，Karl Wiegers，Joy Beatty，清华大学出版社
- 《计算机软件产品开发文件编制指南》（GB 8567-88）
- Information Technology Project Management, Second Edition, Kathy Schwalbe, Course Technology.
- Successful Project Management, Gido, J. and Clements, J. South-Western Publishing.
- On Time and Within Budget: Software Project Management Practices and Techniques, 3rd Edition, Bennatan, E., Wiley.
- Software Project Management: A Unified Framework, Walker Royce, Addison-Wesley.
- IS Project Management Handbook, Doss, G., Prentice Hall.
- CMMI: Guidelines for Process Integration and Product ImprovementMary Beth Chrissis, Mike Konrad, Sandy Shrum.
- CMMI® Distilled: A Practical Introduction to Integrated Process Improvement, Second Edition, By Dennis M. Ahern, Aaron Clouse, Richard Turner.
- CMMI® SCAMPI Distilled Appraisals for Process Improvement, By Dennis M. Ahern, Jim Armstrong, Aaron Clouse, Jack R. Ferguson, Will Hayes, Kenneth E. Nidiffer.
- 军用软件能力成熟度模型可重复级实施指南，石柱，中国标准出版社
- 战略管理(原书第 6 版)，Greey Johnson & Kevan Scholes，王军等译，人民邮电出版社
- 复杂产品系统创新管理，陈劲，科学出版社
- Product Management，4thedition，Donald R. Lehmann & Russell S. Winer，McGraw-Hill Companies，Inc.
- 基于 ITIL®的 IT 服务管理基础篇，Jan van Bon，章斌译，清华大学出版社
- 创新管理-获取持续竞争优势，宁钟，机械工业出版社
- 软件编档导论，金波，清华大学出版社
- 计算机软件工程规范国家标准汇编，中国标准出版社

2 总体设计

2.1 运行环境

2.1.1 软件层面

2.1.1.1 服务器端层面

本项目使用 Oracle MySQL 数据库来存储项目相关的各项数据，版本为 MySQL Ver 8.0.23-0 Ubuntu0.20.04.4 for Linux on x86_64 ((Ubuntu))

2.1.1.2 客户端层面

本项目物联网应用平台主要通过 PC 端或移动手机端浏览器访问以及进行相关的操作。建议使用主流浏览器（Google Chrome, Microsoft Edge, Mozilla Firefox, Safari, Opera etc.）

2.1.2 硬件层面

2.1.2.1 服务器端

表 2-1 服务器 1 硬件接口需求

项目	信息
服务器	阿里云
处理器	Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz
核数	4
内存	8GB RAM
缓存	33792KB
网络	5Mbps
操作系统	Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-31-generic x86_64)
地址大小	46 bits physical, 48 bits virtual

表 2-2 服务器 2 硬件接口需求

项目	信息
服务器	天翼云
处理器	Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
核数	1
内存	2GB RAM
缓存	35840KB
网络	1Mbps
操作系统	Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-72-generic x86_64)
地址大小	46 bits physical, 48 bits virtual

2.1.2.2 客户端

表 2-3 客户端硬件接口需求

项目	信息
供应商	LENOVO
操作系统	系统一：Windows 10 Pro 64-bit (10.0, Build 19042) 系统二：Ubuntu 20.04.1 LTS (GNU/Linux 5.8.0-31-generic x86_64)
系统模型	20LBA01KCD
BIOS 版本	N27ET32W (1.27) (type: UEFI)
处理器	Intel(R) Core(TM) i5-8350U CPU @ 1.70GHz (8 CPUs), ~1.9GHz
内存	20.00 GB RAM
硬盘	512 GB SSD 1.0 TB HDD
显卡	Intel(R) UHD Graphics 620 NVIDIA Quadro P500
显示器	• 名称：Wide viewing angle & High density FlexView Display 1920x1080 分辨率（刷新频率）：1920 x 1080(p) (59.977Hz) • 名称：Generic PnP MonitorAOC2701 分辨率（刷新频率）：1920 x 1080(p) (60.000Hz) • 名称：Generic PnP Monitor DELL U2518D 分辨率（刷新频率）：2560 x 1440(p) (59.951Hz)
网络	Microsoft ATSC Network Provider,0x00200000,0,1,MSDvbNP.ax,10.00.18362.0001Microsoft DVBC Network Provider,0x00200000,0,1,MSDvbNP.ax,10.00.18362.0001Microsoft DVBS Network Provider,0x00200000,0,1,MSDvbNP.ax,10.00.18362.0001Microsoft DVBT Network Provider,0x00200000,0,1,MSDvbNP.ax,10.00.18362.0001Microsoft Network Provider,0x00200000,0,1,MSNP.ax,10.00.18362.0001
鼠标	Logitech M330
键盘	MOTOSPEED GK89

2.2 测试环境

2.2.1 操作系统

Windows 10 Family, Windows 10 Pro, Mac OS, Linux (Ubuntu)

2.2.2 浏览器

Google Chrome, Microsoft Edge, Mozilla Firefox, Safari, Opera

2.3 系统模块与结构设计

2.3.1 系统模块

2.3.1.1 系统模块架构

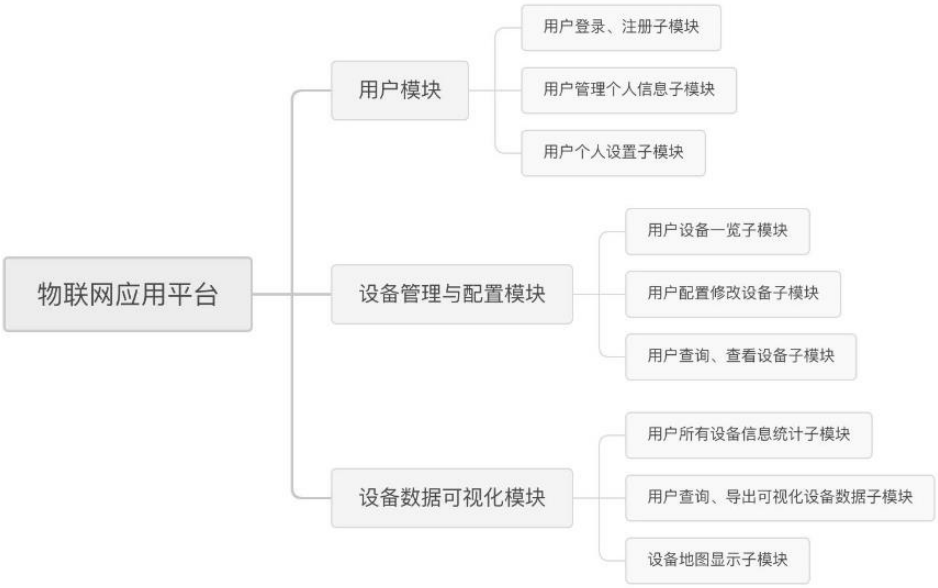

图 2-1 系统模块架构图

2.3.1.2 系统模块功能

2.3.1.2.1 用户模块

- 用户登录
  - 用户邮箱
  - 密码
- 用户注册
  - 用户邮件地址（需要通过邮箱验证）
  - 用户名
  - 密码
- 用户设置（修改个人信息）
  - 真实姓名
  - 性别
  - 用户名
  - 邮件（修改需要认证）
  - 密码（修改有两种方式，输入旧密码或邮件认证）
  - 单位（学校或企业）
  - 个人简介（Markdown 格式）

2.3.1.2.2 设备管理与配置模块

- 展示个人设备列表

- 展示某设备的基本信息
  - 设备名称
  - 设备 ID
  - 设备状态
  - 设备简介
- 展示设备详细数据（可视化界面）
- 设备数据的查询、统计、筛选、监控等

2.3.1.2.3 设备数据可视化模块

- 首页提供统计信息（设备总数量、在线数量、接收的数据等）
- 地图界面展示设备信息，区分正常和警告信息，展示历史轨迹

2.3.2 B/S 体系结构

2.3.2.1 B/S 体系结构简介

随着网络技术的发展，特别随着 Web 技术的不断成熟，B/S 这种软件体系结构出现了。B/S（Browser/Server）架构也被称为浏览器/服务器体系结构，这种体系结构可以理解为是对 C/S 体系结构的改变和促进。由于网络的快速发展，B/S 结构的功能越来越强大。这种结构可以进行信息分布式处理，可以有效降低资源成本，提高设计的系统性能。B/S 架构是有更广的应用范围，在处理模式上大大简化了客户端，用户只需安装浏览器即可，而将应用逻辑集中在服务器和中间件上，可以提高数据处理性能。在软件的通用性上，B/S 架构的客户端具有更好的通用性，对应用环境的依赖性较小，同时因为客户端使用浏览器，在开发维护上更加便利，可以减少系统开发和维护的成本。面向未来，连排级单位可通过掌上电脑（安卓系统），在训练场、演习场等环境下访问并使用该系统。

B/S 的特征和基本结构：在 B/S 结构中，每个节点都分布在网络上，这些网络节点可以分为浏览器端、服务器端和中间件，通过它们之间的链接和交互来完成系统的功能任务。三个层次的划分是从逻辑上分的，在实际应用中多根据实际物理网络进行不同的物理划分。

浏览器端：即用户使用的浏览器，是用户操作系统的接口，用户通过浏览器界面向服务器端提出请求，并对服务器端返回的结果进行处理并展示，通过界面可以将系统的逻辑功能更好的表现出来。

服务器端：提供数据服务，操作数据，然后把结果返回中间层，结果显示在系统界面上。

中间件：这是运行在浏览器和服务器之间的。这层主要完成系统逻辑，实现具体的功能，接受用户的请求并把这些请求传送给服务器，然后将服务器的结果返回给用户，浏览器端和服务器端需要交互的信息是通过中间件完成的。

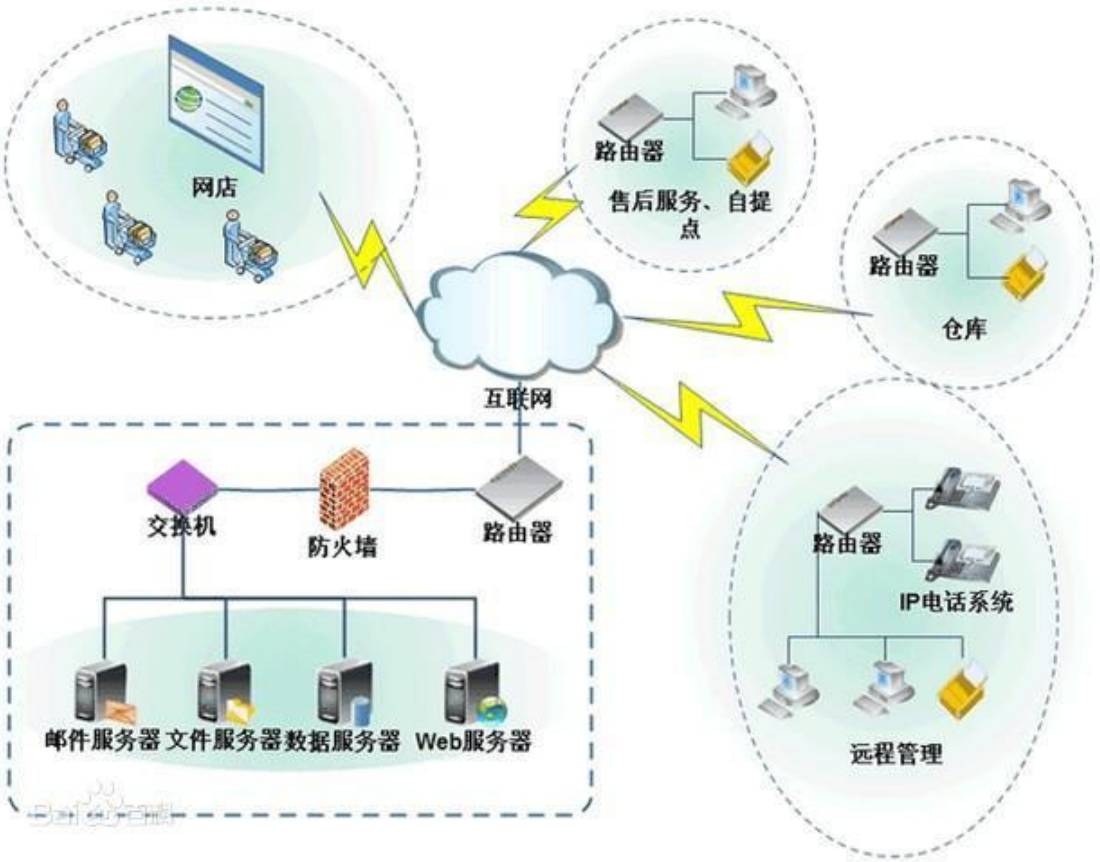

图 2-2 B/S 体系结构示意图

2.3.2.2 B/S 体系结构优劣

- 维护和升级
  软件系统的改进和升级越来越频繁，B/S 架构的产品明显体现着更为方便的特性。对一个稍微大一点单位来说，系统管理人员如果需要在几百甚至上千部电脑之间来回奔跑，效率和工作量是可想而知的，但 B/S 架构的软件只需要管理服务器就行了，所有的客户端只是浏览器，根本不需要做任何的维护。无论用户的规模有多大，有多少分支机构都不会增加任何维护升级的工作量，所有的操作只需要针对服务器进行；如果是异地，只需要把服务器连接专网即可，实现远程维护、升级和共享。所以客户机越来越“瘦”，而服务器越来越“胖”是将来信息化发展的主流方向。今后，软件升级和维护会越来越容易，而使用起来会越来越简单，这对用户人力、物力、时间、费用的节省是显而易见的，惊人的。因此，维护和升级革命的方式是“瘦”客户机，“胖”服务器。
- 成本与选择

大家都知道 windows 在桌面电脑上几乎一统天下，浏览器成为了标准配置。但在服务器操作系统上，windows 并不是处于绝对的统治地位。软件的趋势是凡使用 B/S 架构的应用管理软件，只需安装在 Linux 服务器上即可，而且安全性高。所以服务器操作系统的选择是很多的，不管选用那种操作系统都可以让大部分人使用 windows 作为桌面操作系统电脑不受影响，这就使得最流行免费的 Linux 操作系统快速发展起来，Linux 除了操作系统是免费的以外，连数据库也是免费的，这种选择非常盛行。

- 负荷比

  由于 B/S 架构管理软件只安装在服务器端（Server）上，网络管理人员只需要管理服务器就行了，用户界面主要事务逻辑在服务器（Server）端完全通过 WWW 浏览器实现，极少部分事务逻辑在前端（Browser）实现，所有的客户端只有浏览器，网络管理人员只需要做硬件维护。但是，应用服务器运行数据负荷较重，一旦发生服务器“崩溃”等问题，后果不堪设想。因此，许多单位都备有数据库存储服务器，以防万一。

2.3.2.3 B/S 体系结构前景

C/S 和 B/S 各有优势，C/S 在图形的表现能力上以及运行的速度上肯定是强于 B/S 模式的，不过缺点就是他需要运行专门的客户端，而且更重要的是它不能跨平台，用 C++在 windows 下写的程序肯定是不能在 Linux 下跑的。而 B/S 模式就不同了，它不需要专门的客户端，只要浏览器，而浏览器是随操作系统就有的，方便就是他的优势了。而且，B/S 是基于网页语言的、与操作系统无关，所以跨平台也是它的优势，而且以后随着网页语言以及浏览器的进步，B/S 在表现能力上的处理以及运行的速度上会越来越快，它的缺点将会越来越少。比如，未来的 HTML5，在图形的渲染方面以及音频、文件的处理上已经非常强大了。不过，C/S 架构也有着不可替代的作用。

2.4 基本设计概念和处理流程

2.4.1 服务器

使用阿里云 ECS 云服务器作为服务器，Python 语言的 Django 框架编写后端代码，JavaScript 语言的 React 框架作为前端代码，并且采用 Ant Design 作为 UI 框架，数据库采用 MySQL。当用户通过浏览器使用网站系统时，浏览器接收用户的请求，并传送到服务器或调用内嵌的脚本，分析用户请求，通过数据库接口函数向数据库发送 SQL 查询语句，数据库接收 SQL 查询语句后执行，返回查询结果，处理查询结果后返回给前端，并显示在网站页面上。

2.4.2 客户端

浏览器采用常用的 Microsoft Edge, Google Chrome, Mozilla Firefox 等。客户端在不频繁的操作页面时完成操作后断开与数据库的连接以减轻服务器负荷；在操作频繁时保持连接以增加访问速度。

2.5 人工处理过程

在本系统的运行过程中，可能会出现一些系统无法自动解决的问题，需要人工处理介入来解决。

其中密码重置过程的人工处理：

当用户忘记密码且未填写邮箱或忘记所填写的邮箱时，可以找到有管理该账户权限的教师或管理员，请求重置密码，并及时填写邮箱信息以便于以后的密码找回。

2.6 尚未解决的问题

无

3 数据结构设计

3.1 数据存储

项目产品使用标准 Oracle MySQL 数据库系统作为引擎，按照数据产生、转换和存储的策略，通过将数据导入数据库的方式进行数据的存储操作。

3.2 数据安全

保证以下完整性、保密性以及可用性三个特性来保护用户的数据安全:

- 完整性要求数据未经授权不得进行修改，确保数据在传输和存储过程中不被篡改，盗用和丢失。通过利用安全的框架，通过使用 MD5 或 SHA256 加密算法加密的基础上，运用多种方案和技术实现。
- 保密性要求对数据进行加密，只有授权者才能使用。在本项目中仅超级管理员才有此权限。这一特性要求加密技术必须自动、实时、精确、可靠。
- 可用性要求做到避免因为系统数据泄露而使得合法使用者无法接触可用数据，通过对使用者身份的验证，为合法使用者提供更加安全便捷的使用。力求做到保证安全的同时又尽量减少用户的操作认证负担。

3.3 数据库设计

3.3.1 逻辑结构设计

- 用户表 user
  `user (email, password, user_name, name, gender, work_school, description, phone, address)`
- 登录 Token 表 token
  `token (email, token_value, expire)`
- 设备表 device
  `device (device_id, state, longitude, latitude, class, type)`
- 设备数据表 device_information
  `device_information (device_id, message_content, message_time)`
- 用户-设备拥有关系表 own
  `own (email, device_id)`
- 设备历史信息表 history
  `history (device_id, device_state, message, longitude, latitude, timestamp)`

3.3.2 物理结构设计

- 用户表 user

```
create table user
(
    email varchar(80) not null,
    password varchar(80) not null,
    user_name varchar(80) not null,
    name varchar(80),
    gender int, /* 1 for male, 0 for female */
    work_school varchar(80),
    description text,
    phone varchar(20),
    address varchar(255),

    primary key (email)
)
```

- 登录 Token 表 token

```
create table token
(
    email varchar(80) not null,
    token_value varchar(32) not null,
    expire datetime not null,

    primary key (email, token_value, expire)
)
```

- 设备表 device

```
create table device
(
    device_id varchar(20) not null,
    state boolean not null, /* true for normal, false for warning */
    longitude float not null,
    latitude float not null,
    class int not null, /* 0 for iot device, 1 for cloud computing device, 2 for database device, 3 for cloud storage device, 4 for satellite device */
    type varchar(255) not null, /* for example, 'ECS', 'GPU Cloud Host', 'Elastic High Energy Calculation' and so on */

    primary key (device_id)
)
```

- 设备数据表 device_information

```
create table device_information
(
    device_id varchar(20) not null,
    message_content text not null,
    message_time datetime not null,

    primary key (device_id, message_time)
)
```

- 用户-设备拥有关系表 own

```
create table own
(
    email varchar(80) not null,
    device_id varchar(20) not null,

    primary key (email, device_id),
    foreign key (email) references user(email),
    foreign key (device_id) references device(device_id)
)
```

- 设备历史信息表 history

```
create table history
(
    device_id varchar(20) not null,
    device_state boolean not null,
    message text not null,
    longitude float not null,
    latitude float not null,
    timestamp datetime not null,

    primary key (device_id, timestamp),
    foreign key (device_id) references device(device_id)
)
```

3.3.3 数据库 ER 图

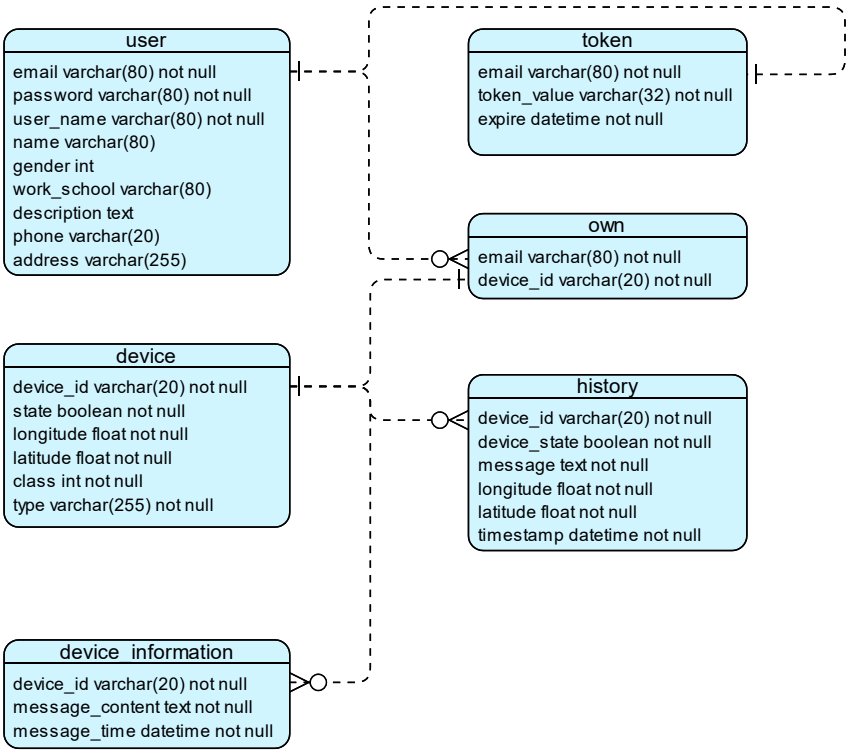

图 3-1 数据库 ER 图

3.4 数据结构与程序的关系

数据结构为关系型数据库，所以在程序中可以用标准的 SQL 语句与数据结构进行交互，交互过程中采用通用的数据反问接口。为了保持良好的程序结构，对数据库访问采用 Python 的 pymysql 封装类库实现，提高维护性和扩张性。

4 接口设计

4.1 用户接口

- 登录接口
- 注册接口
- 忘记密码接口
- 发送邮件验证码接口
- 修改个人基本信息接口
- 修改密码接口
- 查看个人设备列表接口
- 查看某台设备的详细信息接口
- 增添设备接口
- 修改设置设备接口
- 查询设备数据接口
- 查询设备状态接口
- 查看设备地图接口
- 导出可视化数据接口

4.2 外部接口

4.2.1 数据库接口

本系统的所有数据存储在服务器及数据库中，包括账号密码、通知、设备信息、设备数据信息、数据地理位置信息等数据。

资源文件及不适宜数据库表项存储的超长文本存储在文件中。

网页前端获取用户输入后，由网页后端完成与服务器及数据库的交互。利用 Python 的 pymysql 库与 MySQL 之间的接口完成网站外部接口设计。

本系统的初始数据依靠人工导入存储。

4.2.2 MQTT 服务器接口

本人在自己的服务器上搭建了一个基于 MQTT 协议的服务器——EMQ X Borker（https://www.emqx.cn/products/broker），服务器管理网页端口为 18083，提供 MQTT WebSocket 接口端口为 8083.

该服务器的管理界面如下图所示：

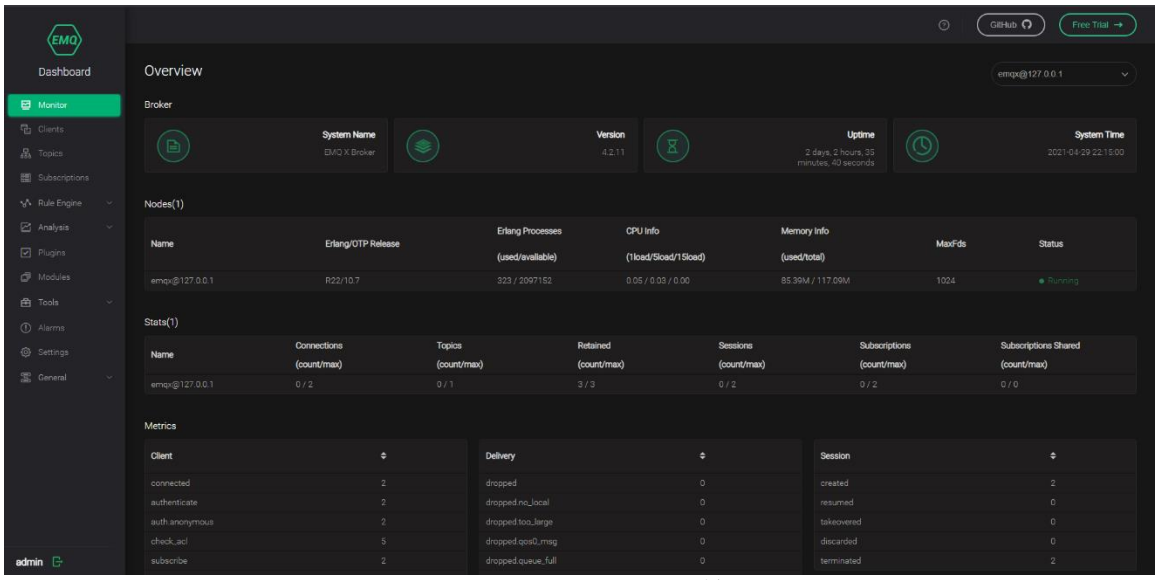

图 4-1 EMQ X Broker 服务器管理界面

4.3 内部接口（前后端接口）

本系统总体分为用户管理、设备管理、设备数据可视化三个模块，各个模块之间耦合度较低，主要通过用户邮件地址作为用户 ID 和设备 ID 进行连接。

但各模块共享数据库表项，各个模块之间相对可以独立开发，但必须先同一数据库数据和 HTTP 请求数据。

详细的前后端 HTTP 请求接口（Json 格式数据）规范见本人后期的文档——《物联网应用平台 前后端接口说明》。

5 运行设计

5.1 运行模块的组合

本系统总体分为用户管理、设备管理、设备数据可视化三个模块。

用户登录成功后，会获得后端返回的一个 token，作为每次发送强求的通行凭证，只有通过了后端的 token 验证才能继续在前端进行相关的操作。

一名用户可以自己添加多台不同种类、不同地区的设备，每台设备都具有自己的基本信息，并且会以随机时间向 MQTT 服务器发送数据。

每台设备的状态、数据、信息等都会同步更新到数据库中。

每次用户修改设备信息、导出设备相关数据的时候，都会以日志的形式记录到数据库中。

因此，虽然各个模块之间不会共享界面，但是共享数据库的数据，通过后端的业务逻辑代码进行相应的操作。后台程序只共享数据库连接。

5.2 运行控制

- 界面
  用户界面是用户直接与系统交互的部分，界面要求做到简约但不简陋，能引导用户进行无障碍、连续操作。设计时，在为用户提供便捷、快速的操作基础上适当增加细节部分，使得界面具有一定的美观性。
- 运行控制的条件与限制
  本项目的开发要求能及时保质保量完成任务。且项目开发过程中一定会存在技术上的难点和设备方面的欠缺，需要开发个人合理利用现有设备和资源，积极查找资料解决问题，在完成项目开发的基础上，同时保证项目的可用性、安全性、可维护性、可扩展性等。
- 前台与后台的关系
  - 前台主要展示用户个人基本信息、用户拥有的设备数据、设备状态和设备地理位置等基本信息。
  - 后台主要负责业务逻辑， 控制前台显示信息，负责与数据库和前端的数据交互 。
- 运行时的安全控制
  本系统因为涉及到用户个人物联网终端设备数据的隐私问题，对数据库的保护比较重要。系统将对重要信息加密（一般采用不可逆加密算法，如 MD5、SHA256 等）。数据库操作前预处理参与数据库查询的输入信息，尽量多采用选项选择的方式输入信息，如果采用文本框输入的方式应当对文字进行一定程度的检查，防止脚本注入等常见攻击。同时对于设备信息的传输将进行再次加密。

5.3 运行时间

前端通过 WebSocket 与 MQTT 服务器进行数据的持续监听与传输，会占用较多的浏览器资源，而这和用户自身设备配置有关。

同时，后端一方面需要与 MQTT 服务器进行数据的持续监听与传输，一方面需要将监听到的数据记录到数据库内，因此这也会占用较多的数据库资源和后端计算机的资源。

6 系统异常设计

6.1 出错信息

表 6-1 出错信息及处理表

数据输入	处理方法	说明
数据库泄露	使用加密手段，如 sha1 和 md5 双层加密	后台服务器被入侵，数据库中用户信息被窃取
数据库连接失败	修改数据库配置，尝试重新连接	配置出错，或并发访问的用户过多
数据库丢失	定期备份数据库	由于物理因素，数据库部分数据丢失
数据库乱码	统一编码方式	客户端和服务器端数据库读取过程编码不一致
Cookie 出错	强制清楚 cookie，重新登录	用户 cookie 的 token 与数据库中用户信息不一致，可能由于用户 cookie 被劫持，账号被盗取

6.2 补救措施

表 6-2 补救措施表

项目	措施
数据备份	定期备份系统数据，防止当系统数据因不可抗拒力丢失造成的损失。
分布存储	将系统部署到不同计算机上，减小硬件损坏造成的数据丢失的影响。

7 系统维护设计

7.1 概述

我们将从三个方面进行系统维护：

表 7-1 维护方式表

项目	手段
数据库	销毁数据库连接时使用 try catch 语句捕获异常，对不同的错误信息区分输出
管理员	对整个网站的状况进行控制，以防系统出现不可预计的错误，避免系统显示不合法信息
维护人员	每次维护后留下完备可读的系统维护日志便于管理员和其他维护人员查看

7.2 检测点设计

7.2.1 用户模块

- 用户登录、注销
- 用户注册
- 用户验证邮件地址
- 用户修改密码
- 用户修改个人信息

7.2.2 设备管理与配置模块

- 用户查看所有设备
- 用户新增设备
- 用户修改、配置设备信息
- 用户导出设备数据

7.2.3 设备数据可视化模块

- 用户查看某设备的实时数据（可视化）
- 用户查看设备地图（可视化）