四川大學錦江學院

Sichuan University Jinjiang College

毕业设计

题　　目　基于 Nuxt3 的水务数据监控系统的设计与实现

学　　院　计算机学院

专　　业　软件工程　　年级　2021

学生姓名　田正东

学　　号　2105200206

指导教师　周　刚

学位论文原创性声明

郑重声明：所呈交的学位论文《基于 Nuxt3 的水务数据监控系统的设计与实现》，是本人在导师的指导下，独立进行研究取得的成果。除文中已经注明引用的内容外，本论文不包括其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果，并承诺因本声明而产生的法律结果由本人承担。

学位论文作者签名：田正东

日期：2025 年 5 月 30 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权四川大学锦江学院将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

保　密☐，在__年解密后适用本授权书。

不保密☑。

学位论文作者签名：田正东　　指导教师签名：周刚

日期：2025 年 5 月 30 日　　日期：2025 年 5 月 30 日

基于 Nuxt3 的水务数据监控系统的设计与实现

软件工程 专业

学生：田正东 指导教师：周刚

【摘 要】 随着 Web 技术的发展，现今的 Web 技术可以为智慧水务数据监控系统提供更加丰富的界面和易集成的技术支持。而传统项目实现丰富界面代码书写效率低下。本研究基于 Nuxt3 和 NestJS 框架，设计并实现了一套智慧水务数据监控管理系统。系统采用前后端分离架构，包含 Web 端可视化平台、后端数据处理服务和移动端的通知功能。通过需求分析、技术选型和架构设计，实现了水务大屏可视化、故障报修、设备管理、定额管理等核心功能模块。本研究为校园水资源数字化管理提供了可行的技术解决方案。希望通过本设计为管理水资源的部门或者学校助力，更好地应对水资源中的各项挑战。

【关键词】 数据可视化 水资源管理 Nuxt3 NestJS

Design and implementation of water data monitoring system based on Nuxt3.

【Abstract】 With the development of web technology, today's web technology can provide smart water data monitoring systems with richer interfaces and easy-to-integrate technical support. Traditional projects realize rich interface code writing efficiency is inefficient. This study is based on Nuxt3 and NestJS frameworks and designs and implements a set of smart water data monitoring and management systems. The system adopts a front-end and back-end separation architecture, including a web-end visualization platform, a back-end data processing service and a mobile notification function. Through demand analysis, technology selection and architecture design, core functional modules such as water service large screen visualization, fault repair, equipment management, and quota management have been realized. This study provides feasible technical solutions for digital management of campus water resources. We hope that through this project, we will help departments or schools that manage water resources; we will better respond to various challenges in water resources.

【Keywords】 Data Visualization Water Resource Management Nuxt3 NestJS

目 录

1 绪论

在 2021 年 Nuxt3 的发布，为 Vue 的生态统一迈开步伐，现今的水务数据管理系统仍然局限在 Vue 框架或者更老的技术，老框架引入工具库繁琐，在配置方面花的时间太长，而 Nuxt3 却可以一键引入。Nuxt3 整合了前端 Vue 开发，因此本设计将研究使用 Nuxt3 构建水务数据监控 Web 应用，为未来的水务数据监控系统铺路。

1.1 国内外研究现状

1.1.1 国内研究现状

我国智慧水务虽起步较晚，但在政策支持与技术创新的双重驱动下，近年来发展迅速。国内研究主要聚焦于物联网感知层优化、大数据分析平台构建、机器学习模型应用及智慧管理模式探索，并形成了一批具有示范意义的标杆项目如下：

周璇在 2020 年提出目前我国县镇地区大多仍采用人工抄表的方式,管理效率低,且用水信息尚未得到科学利用,他面向我国县镇地区设计开发了一套基于 NB-IoT 的智慧水务系统,实现了低功耗的广域网网络环境下的数据实时传输；并且提供远程抄表、计费管理、用水量预测等信息化、智慧化服务[1]。

杨铭在 2023 年指出压力信息的采集、监控和记录对于生产和生活具有非常重要的意义，提出轻量化目标检测算法识别指针式压力表,自动数据上传云端，提高生产效率和降低人力成本[2]。田甲坤在 2023 年提出了一种城市需水量预测方法。初步选取机器学习中差分整合移动平均自回归模型、季节性差分自回归滑动平均模型和神经网络学习中卷积神经网络模型和长短期记忆网络模型进行需水量预测,对比多种机器学习算法，证明 LSTM 模型在城市需水量预测中表现最优[3]；刘鑫（2023）则构建多因素耦合模型，量化经济、气候对用水量的影响。侧重从数据驱动的角度来理解过程,为城市用水预测提供全新的视角,破解城市用水预测较难且精度较低的问题提出研究深圳市用水时空变化规律的定量分析方法,揭示不同空间分布、不同用水规律下用水对象的用水痕迹[4]。

通过查资料我们现有水务相关平台有：朱伶俐在 2022 年开发智慧水务监测平台，基于数据分析和监测展开对智慧水务系统的设计与研究,并建立了优化调度模型。在研究过程中,需要通过水量监测等环节有效地平衡二次供水的压力和扬程,实现智慧水务平台的开发[5]。侯伟光在 2022 年设计基于 Hadoop 大数据技术的智慧水务平台，支持多种源数据融合分析[6]。可以发现我国的技术研究趋势在数据采集与物联网技术、决策支持与系统集成、大数据分析与基于机器学习智能预测。

而其中存在的挑战与机遇如下：王秦飞在 2019 年强调老旧基础设施改造滞后影响技术落地[7]。大数据时代下的城市供水监管探索与实践中指出，发现当前仍存在难以形成多

部门协同的工作机制、对城市建成区外的面源污染控制难度较大等问题[8]。除了，基础的设施落后外，现在的水务管理系统前端框架依旧老旧，仍然保持使用 jQuery、Vue2 等前端技术，无法使用一些提升开发效率的库。因此有必要更新网页技术。

我国水务未来方向体现在研究建议加强跨领域数据互通（如水利与环保）、推广“水管家”模式，并探索数字孪生技术在流域管理中的应用[9]。

随着全球化的加深与技术的进步，智慧水务平台的发展前景十分广阔。国外的成功经验为国内提供了宝贵的借鉴，而国内企业的积极探索也为智慧水务平台的未来描绘了美好蓝图。相信在不久的将来，智慧水务平台将在保障水资源可持续利用方面发挥更大的作用。

1.1.2 国外研究现状

国外智慧水务发展较早，已形成成熟的技术体系与管理模式，代表性企业通过物联网、大数据与 AI 技术实现水务全流程优化。

并且，国外水务巨头重视中国水务市场商机，例如威立雅水务致力于水资源的可持续发展，倡导水资源的可持续发展，倡导水资源的循环利用。威立雅水务在本次展会上重点展示了“水的循环利用”解决方案。威立雅有关人士认为，人口的增长、城市化进程、工业和旅游的发展，以及农业灌溉，这一切都使水资源的满足和保护变得更加困难。像所有消费品一样，水的循环利用变得越来越必要。威立雅水务凭借其对水循环全过程的管理经验，可以为各地区提供专有技术，并提供可靠、经济且生态化的循环利用解决方案[10]。企业不仅重视数据的采集与初步分析，更是将大数据技术应用于生产运营的各个方面，实现了从流程控制到能耗分析的全方位覆盖。其中技术应用特点有:由 IBM 利用水资源运营数据预测供需矛盾，优化资源配置形成的数据驱动决策； 还有日本正兴电机集团开发的水质监视装置，通过观察鱼受到水中毒物影响后的异常举动进行三维图像解析，来判断水质所形成的创新检测手段[6]；

1.1.3 发展趋势

根据国内外研究现状，未来智慧水务的核心是:“全域感知-智能分析-自主决策” 的闭环体系。物联网将实现水务系统的“神经末梢”全覆盖，从而实现数字孪生；AI 分析赋予系统“大脑”，从数据中挖掘价值。

水管家模式：在建设过程中存在的地方财政压力大专业技术力量不足等问题，可以模仿“水管家”的模式，厘清水管家模式的定义、特点、建设模式以及“水管家”与当地政府之间的职责关系。该模式可有效整合地方存量资源和社会资本，提升当地水利工程建设的专业化水平，并带动区域产业经济协同发展。这一趋势不仅提升水务运营效率，还将助力全球水资源可持续利用，为智慧城市奠定关键基础设施[9]。

国内外的研究情况如：表 1.1 国内外研究总结对比

表 1.1 国内外研究总结对比

维度	国内特点	国外特点
技术重点	侧重物联网部署与大数据平台建设	成熟的数据分析与 AI 应用
典型案例	保定、深圳等地方政府与企业合作项目	威立雅、IBM 等跨国企业主导
挑战	数据孤岛、基础设施滞后、技术落后	技术成本高、本土化适配难、技术落后
趋势	政策驱动加速技术落地，探索数字孪生与跨领域协同	强化预测性维护与全球化服务输出

1.2 论文主要工作

该设计主要是使用先进前端技术 Nuxt3 来完成智能水务数据监控系统的设计与实现，通过对系统整体进行需求分析,将系统分为前端、后端、移动端三个部分。通过 RBAC 权限模型，实现用户有不同权限从而显示不同的界面，完成功能模块有水务大屏可视化模块、故障报修模块、用水分析、定额管理、设备管理模块。管理员负责创建角色为用户划分权限，用户分为登录用户和未登录用户，登录用户有更多的权限。移动端仅且包括故障报修模块。

本课题将以某大学引入物联网设备需要一个对设备和水信息管理的系统为开发需求，结合现代 web 技术开发和 ECharts 等工具开发水务管理系统，为学校开发一个后续可以对接物联网设备的平台系统，同时模拟物联网设备收集水源信息（如水量、流速、水质等），完成可以通过水使用情况是去指导节约使用水的功能，同时能够管理设备、设置用水额度。超额提醒。借助 ECharts 将纯粹的数据转化为形象直观的图表，可以帮助我们更直观地观察用水规律。因此，本项目计划基于 Nuxt3（前端 Vue3 框架）和 NestJS 开发一个高效的可视化水务监控管理系统。

因此本设计可以帮助学习 Nuxt3 的人更好理解服务端渲染原理。也能学习使用 Nuxt3 的一些规范，学习使用 Nuxt3 构建动态路由。

1.3 论文组织与架构

本设计主体部分由六个部分组成，各部分作用分别为：

第一部分：绪论。介绍水务数据监控系统的选题意义、国内外研究现状。

第二部分：介绍了实现本系统所用的相关技术等内容。

第三部分：介绍本系统的系统概要及设计，包括功能性需求分析等内容。

第四部分：介绍对本系统架构的详细设计，系统功能的核心程序代码。

第五部分：介绍对本系统的前端涉及到的功能实现。

第六部分：描述对系统的测试方法和测试结果。

2 系统开发的核心技术与工具

2.1 技术选型与介绍

首先对于一个大数据展示系统，用户使用人数是非常少的，所以在技术选型上，非必要选择 JAVA 语言，去完成高并发和所谓的性能。国内学习 JAVA，而国外更加偏爱使用简单的语言 JavaScript，并且开发了高性能的 NodeJS 应用框架，借助浏览器的使用率，JavaScript 生态非常丰富，有个段子讲到，JavaScript 的包足以超过黑洞。因为 NodeJS 的性能足以媲美 JAVA，且大数据展示系统，无需多大的访问量，并发不是很重要。为了构建高效、可维护、可扩展的水务管理系统,选择了以下最新的核心技术栈：

2.1.1 Nuxt3

它提供了服务器端渲染（SSR）、代码拆分、自动导入等功能，能够显著提升 Web 应用的性能和搜索引擎友好性。Nuxt3 中“composables、components、utils”目录中的 Vue 组件是自动导入的，可以直接在模板和 JS/TS 文件中使用[11]。在性能优化方面，通过 SSR 和代码拆分，Nuxt3 显著提升了首页加载速度和整体运行性能，为用户带来更流畅的体验。其“约定大于配置”的设计理念极大简化了开发流程，传统 Vue 开发中繁琐的路由配置在 Nuxt3 中可由目录结构自动生成，节省了大量时间与精力。同时，Nuxt3 支持组件和工具方法的自动导入，进一步提高了开发效率。项目结构清晰统一，支持模块化开发，便于团队协作与后期维护。此外，Nuxt3 还拥有强大的钩子系统，允许在应用生命周期的关键节点插入自定义逻辑或第三方库，这种高度可扩展的机制为系统的未来发展提供了无限可能。

2.1.2 PrimeVue：

Vue 作者尤雨溪主推的 Vue UI 框架，特点在于样式应该由使用者自定义，而不是，使用样式穿透去修改样式，独特的 CSS Token 概念，值得学习。

动态样式生成： PrimeVue 支持动态生成样式，开发者可以通过简单的配置快速生成符合需求的样式代码，减少手动编写 CSS 的工作量。

模块化设计：采用模块化设计，PrimeVue 允许开发者按需加载样式模块，避免不必要的资源浪费，同时提升应用的加载速度。

主题定制：提供强大的主题定制功能，开发者可以轻松定义和管理多套主题，满足不同场景下的样式需求,这点非常帮，未来只需要维护一套主题变量，实现主题随心切换

2.1.3 UnoCSS：

UnoCSS 继承了 Windi CSS 的按需特性，属性化模式、快捷方式、变体组、编译模式等等。最重要的是，UnoCSS 是从头开始构建的，考虑到了最大的可扩展性和性能，使我们能够引入 纯 CSS 图标、无值的属性化、标签化、网络字体 等新功能[12]。

前端开发很火的 CSS 框架，提倡样式原子化，再也不用在编写样式的时候想一个好名字而发呆了，把样式简写比如 flex 就是 display：flex，见名知意，写在 class 里面，大

大提升开发效率。按需生成：UnoCSS 只会在构建时包含实际使用到的样式规则，这大大减少了最终生成的 CSS 文件大小。相比传统的 CSS 框架，这种做法可以显著提高页面加载速度和性能。快速开发周期：由于 UnoCSS 的原子类属性可以直接应用在 HTML 元素上，并且能够立即看到效果，因此它极大地缩短了从设计到实现的时间，提高了开发效率。

高度可配置的主题系统：UnoCSS 提供了灵活的主题配置选项，允许开发者根据项目的具体需求自定义颜色、字体、间距等样式变量。此外，通过简单的配置即可轻松调整整个项目的视觉风格。支持插件机制：UnoCSS 支持插件扩展，可以根据项目需求添加额外的功能或样式规则。这使得 UnoCSS 能够适应各种不同的应用场景，增强了框架的通用性和灵活性。模块化设计：每个 UnoCSS class 属性都专注于单一功能，这意味着样式的复用性和可维护性得到了极大提升。开发者可以通过组合不同的原子类来创建复杂的布局，而不需要担心样式冲突或覆盖问题。直观的命名规范：UnoCSS 的类名遵循一致且直观的命名规则，降低了学习曲线，也使得代码更加易于理解和维护。

2.1.4 ECharts：

ECharts 提供了折线图、柱状图、饼图等多种图表类型，帮助用户直观理解用水数据。动态交互功能比如支持缩放、拖拽、提示框等交互功能，增强了用户体验。在 Nuxt33 中使用时，需注意 ECharts 仅能在客户端渲染，可通过 Vue 生命周期钩子确保组件仅在客户端渲染

2.1.5 Flutter

它采用 Dart 语言编写，是一款由 Google 推出的跨平台移动开发框架，支持快速构建高质量的 Android 应用。Flutter 的热重载功能显著提升了开发效率，使开发者能够实时查看代码更改效果，从而加速迭代过程。其运行原理允许直接操作原生 Canvas，相比 WebView 具备更优的性能表现，尤其适合对性能要求较高的应用场景。同时，Flutter 提供了丰富的 UI 组件和高效的渲染引擎，能够为用户提供流畅的交互体验，有效降低开发和维护成本。

2.1.6 NestJS

Nest（NestJS）是一个用于构建高效、可扩展的 Node.js 服务器端应用程序的框架。它使用渐进式 JavaScript，使用 TypeScript 构建并完全支持 TypeScript（但仍允许开发人员使用纯 JavaScript 编写代码），并结合了 OOP（面向对象编程）、FP（函数式编程）和 FRP（函数式反应式编程）的元素[13]。使用 NestJS 构建的系统具备高度可靠性和可扩展性，非常适合长期维护和支持复杂项目的需要。NestJS 的设计理念与 MVC 架构保持一致，拥有丰富的功能组件（如拦截器、中间件、数据验证、网关），这些都有助于加快数据处理速度，提升系统响应能力。

2.1.7 数据存储与处理

MySQL：作为一种关系型数据库，MySQL 提供了高效的数据存储和查询能力，是满足系

统数据需求的理想选择。Redis：作为一种高性能缓存系统，Redis 显著提高了数据访问速度，从而提升了系统的整体性能。它支持多种数据结构，为实现复杂的业务逻辑提供了便利，例如用户鉴权、与 SSE 系统集成、通知用户以及踢用户下线等功能。

通过上述技术选型及优化策略，致力于创建一个不仅在性能、用户体验和安全性方面达到高标准，而且在长期可维护性和扩展性上同样表现出色的智能水务管理系统。

2.2 技术优势分析

首先，为什么选择 NodeJs 的框架而非是 JAVA 的 SpringBoot 框架？

在 MDN 中介绍 JavaScript 如：JavaScript 是一门跨平台、面向对象的脚本语言，它能使网页可交互（例如，拥有复杂的动画、可点击的按钮、弹出菜单等）。还有一些更高级的服务器端 JavaScript 版本，如 Node.js，它们允许你为网站添加更多功能，而不仅仅是下载文件（例如，多台计算机之间的实时协作）。在宿主环境（例如 Web 浏览器）中，可以将 JavaScript 和宿主环境的对象连接起来，并以编程的方式控制这些对象[14]。

因为 JavaScript 创建对象方式简单，不需要写类来装对象，灵活使用大括号就可以组装对象，可以直接组装想要的数据，返回给前端。在效率上有巨大提升。

2.3 本章小结

选择 Node.js 框架而非 Java 的 Spring Boot，主要是考虑到 JavaScript 的灵活性及其在创建对象时的简便性，这不仅加速了前后端的数据交换过程，也在整体开发效率上实现了巨大提升。综上所述，通过精心挑选的技术组合，本设计选择 Nuxt3+NestJs 技术栈。

3 系统需求分析与设计

3.1 系统需求分析与建模

3.1.1 系统需求概述

在一个实用的水务系统中，水管破损，第一时间发现的人需要上传，维修人员需要第一时间收到消息。因此将使用人群分为三类，维修人员、用户、管理员。管理员负责分配权限，维修人员，负责接收消息，完成维修，上传维修完成的情况。而使用用户可以看到大屏展示和能够提交那个地方需要维修的信息。维修人员能够及时收到消息。每个用户的界面都不同，因此使用 RBAC 权限模型来做权限分配。记录的设备数据需要根据水表的情况实时传递，需要使用 socket 连接实时传递。

系统的具体功能模块的需求如下：

水务大屏可视化模块:需要提供各个时间节点如：今日、本月、本年、上月的用水数据、各区域实时用水占比、今年每月用水量趋势、5 年年度用水总量同比增长趋势、节水信息，方便用户了解当前用水情况。

故障警示报修:实时展示用水异常报警信息,包括报警时间、地点、类型等信息,帮助系统及时发现问题。支持用户提交故障报修,并实现维修工的抢修和维修记录管理,确保故障及时修复。

用水分析:分析用水数据,生成报告,为用户提供用水规律的分析和节水建议。

定额管理:制定和管理不同区域、不同户型的用水定额,达到节约用水。

设备管理模块:模拟连接物联网水表,实时查看设备运行状态和维护记录。

3.1.2 系统数据分析

针对要实现的上面的五个模块，进行如下分析：

基于用户权限管理模型，需要完成标准的 RBAC 权限模型，用户与角色建立多对多关系，角色与菜单权限建立多对多关系。这样就完成权限的动态配置，角色可以继承权限，后面还能，修改菜单权限，完成前端的动态菜单功能。ER 图如：图 3.1 RBAC 权限 ER 图

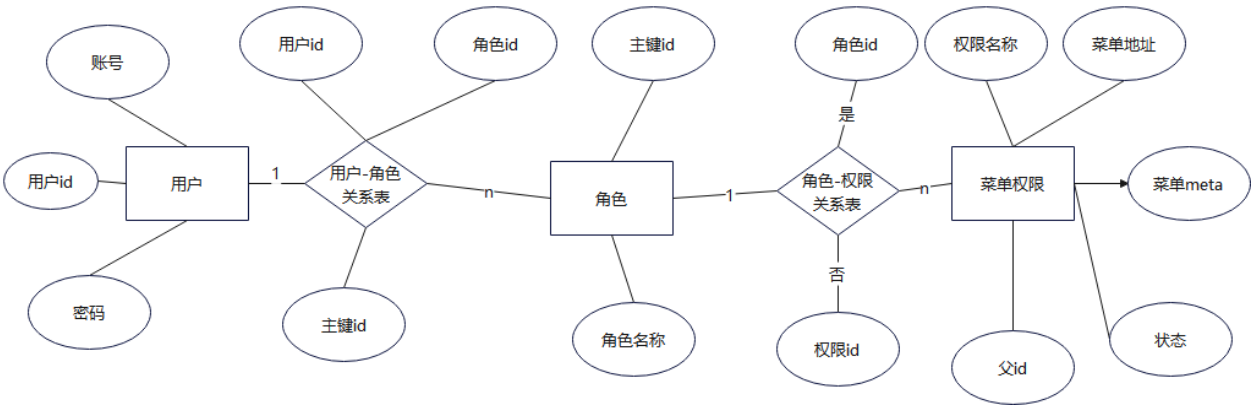

图 3.1 RBAC 权限 ER 图

设备管理，需要记录设备的基础信息，还有设备的实时信息，而信息需要记住设备安装位置、设备编号、设备名称、设备制造商等。而区域表有位置信息、楼层信息、房间信息。因此设备基础信息与专用设备表（比如：水表、压力表）需要形成继承关系，区域表采用层级结构有校区、楼房、楼层、房间。他们之间的关系是父子关系，设备通过地址 id 关联到具体安装位置，为设备的全生命周期管理打好基础，一些带有经纬度的设备还能在地图展示。ER 图如图 3.2 设备区域 ER 图：

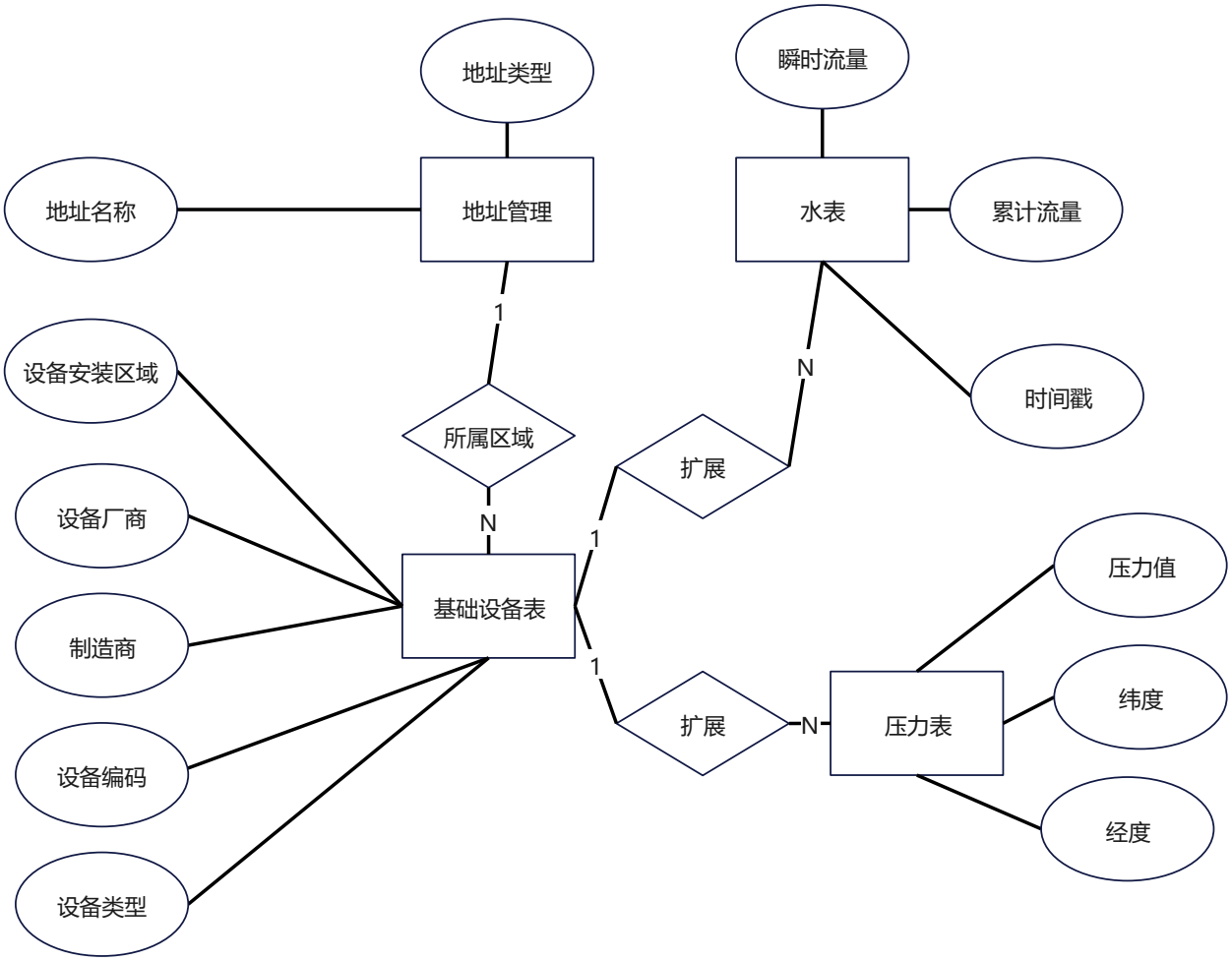

图 3.2 设备区域 ER 图

故障报修的流程是先扫码上报,在分派师傅维修，师傅处理完成上传系统，最后管理员验收。每一个流程需要审计，需要通过 records 数据库表记录操作人和现场数据，支持上传图片，和地址位置信息的标记，ER 图如：图 3.3 故障报修 ER 图

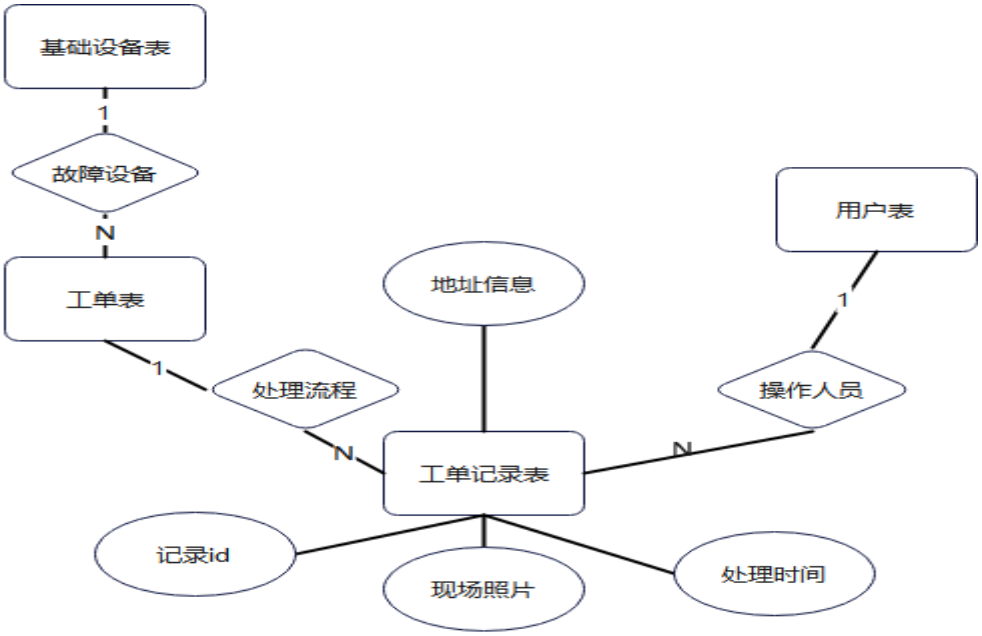

图 3.3 故障报修 ER 图

每个设备和区域可以设置用水额度，每当超过一个额度，就提醒管理员。定额周期灵活管理，设置日、周、月、季、年类型。每条设置的额度信息，都可以绑定用户 id，设备和区域。如：图 3.4 定额管理 ER 图

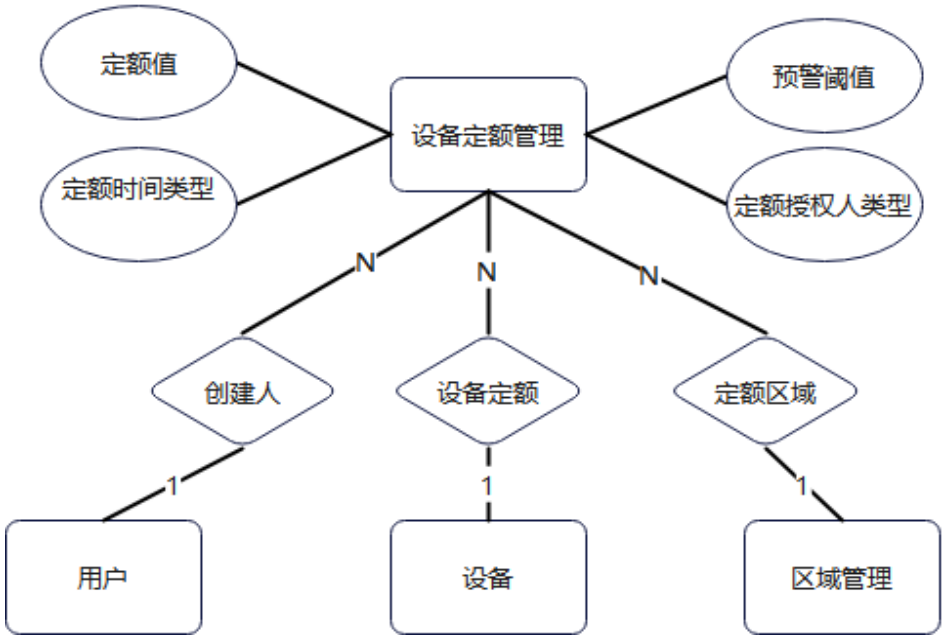

图 3.4 定额管理 ER 图

3.1.3 系统功能分析

该系统采用前后端分离的架构,分为三个主要部分，如：图 3.5 系统功能分析

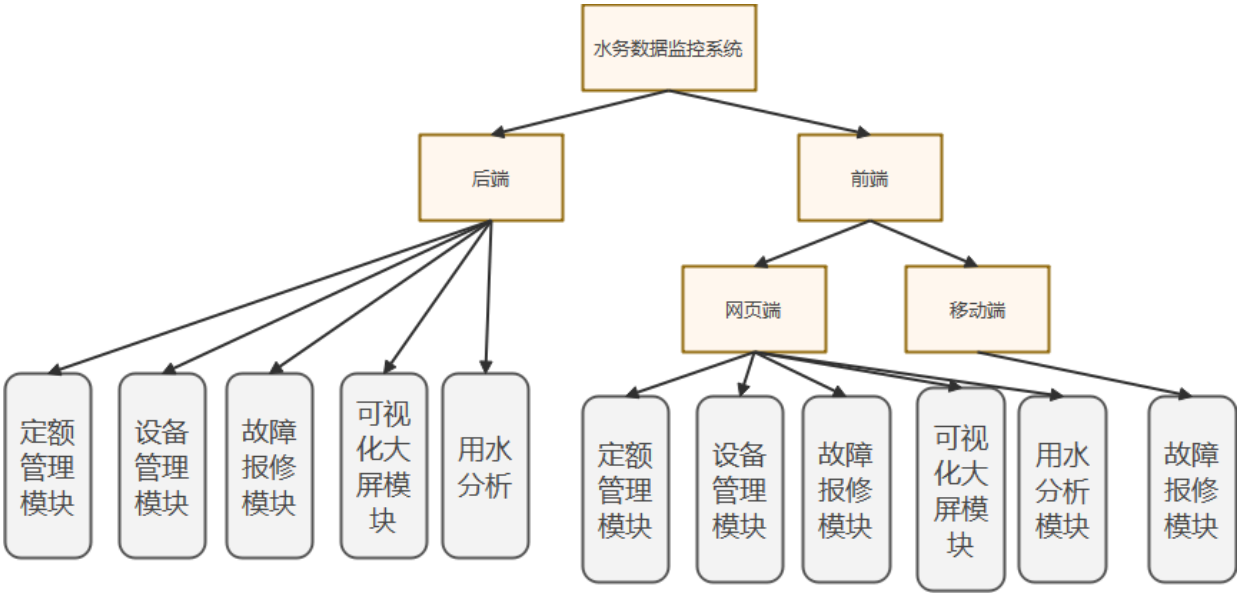

图 3.5 系统功能分析

Web 端:面向管理员和普通用户,提供更加丰富的功能和操作界面。主要功能包括大屏可视化、故障报修平台、定额管理、用水分析、设备管理。大屏可视化模块:展示最新数据用水数据、用水趋势、节水信息、历史用水数据、用水报警信息,以及用水量同比等信息,方故障警示报修:用户提交故障报修,可选择故障类型、填写位置、上传现场图片等,记录报修时间和描述等信息;用水分析:分析用水数据,生成报告,为用户提供用水规律的分析和节水建议;定额管理:可以设置和修改不同区域、不同户型的用水定额计划;设备管理模块:查看、添加和删除设备信息,查看设备运行状态和维护记录。

移动端:面向维修人员和普通用户,提供简洁的操作界面,用户提交故障报修,可选择故障类型、填写位置、上传现场图片等,记录报修时间和描述等信息。

后端:提供统一的 API 接口,支持 Web 端和移动端的业务逻辑处理,同时管理数据库和系统数据。用户管理:负责添加、删除、修改用户信息及分配相应权限。这一过程不仅涉及基本信息(如用户名、密码、联系方式)的维护,还包括为每个用户量身定制权限集合,确保每个用户只能访问与其职责相关的资源。系统额度参数设置:设置额度,方便前端显示百分比。例如,设定每日、每周或每月的用水量监测标准。设备管理权限管理:能够给其他用户分配设备管理的增加、删除、更新等权限。故障警示报修管理:能够给其他用户分配首页警告通知的条件设置等权限。

根据本系统需求的 5 个功能和三类用户设计的系统功能 UML 的用例图如:图 3.6 系统用例图

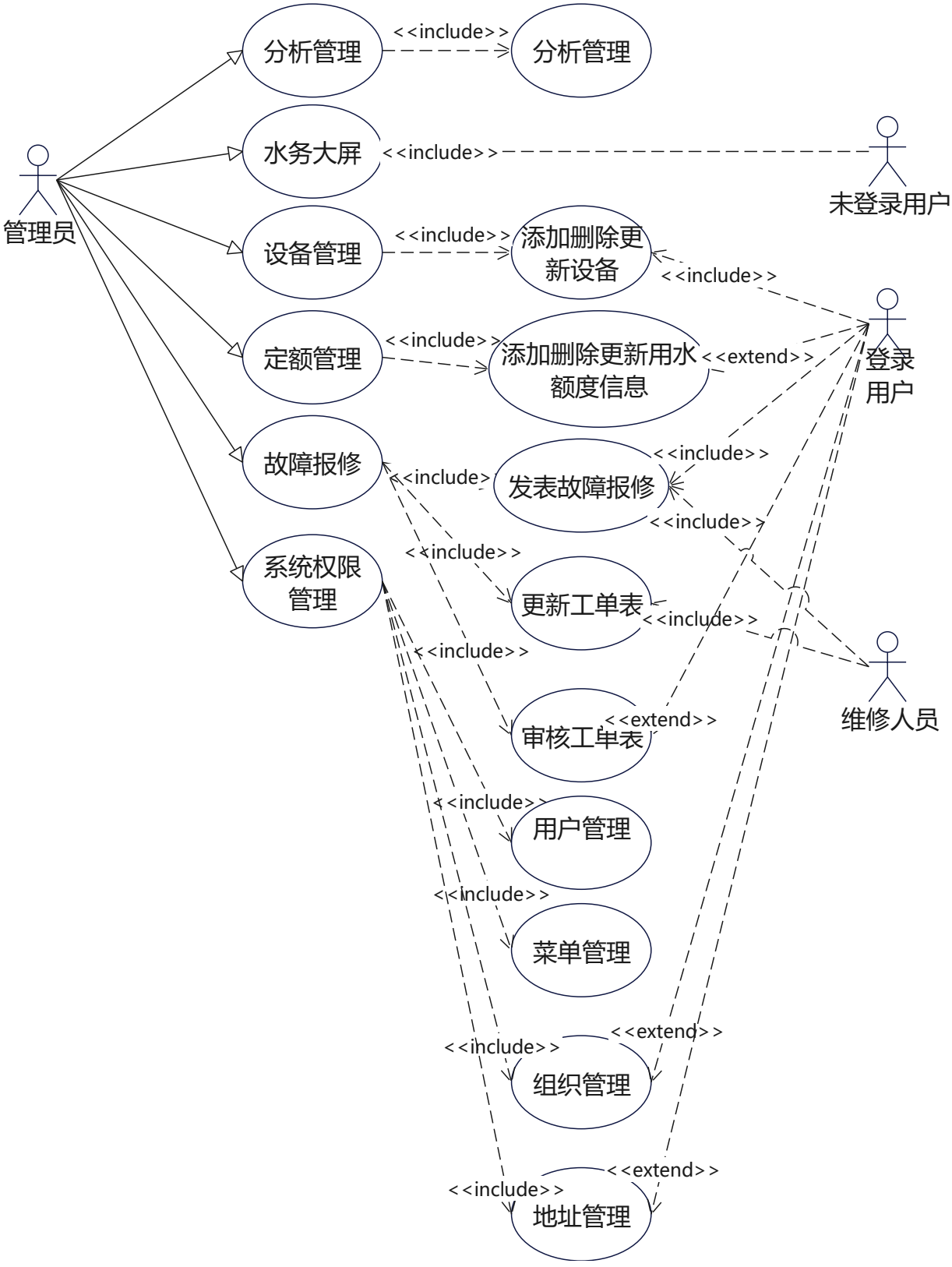

图 3.6 系统用例图

设计的设备报修 UML 的序列图如：图 3.7 系统设备报修序列图

图 3.7 系统设备报修序列图

系统 UML 的活动图如：图 3.8 系统活动图

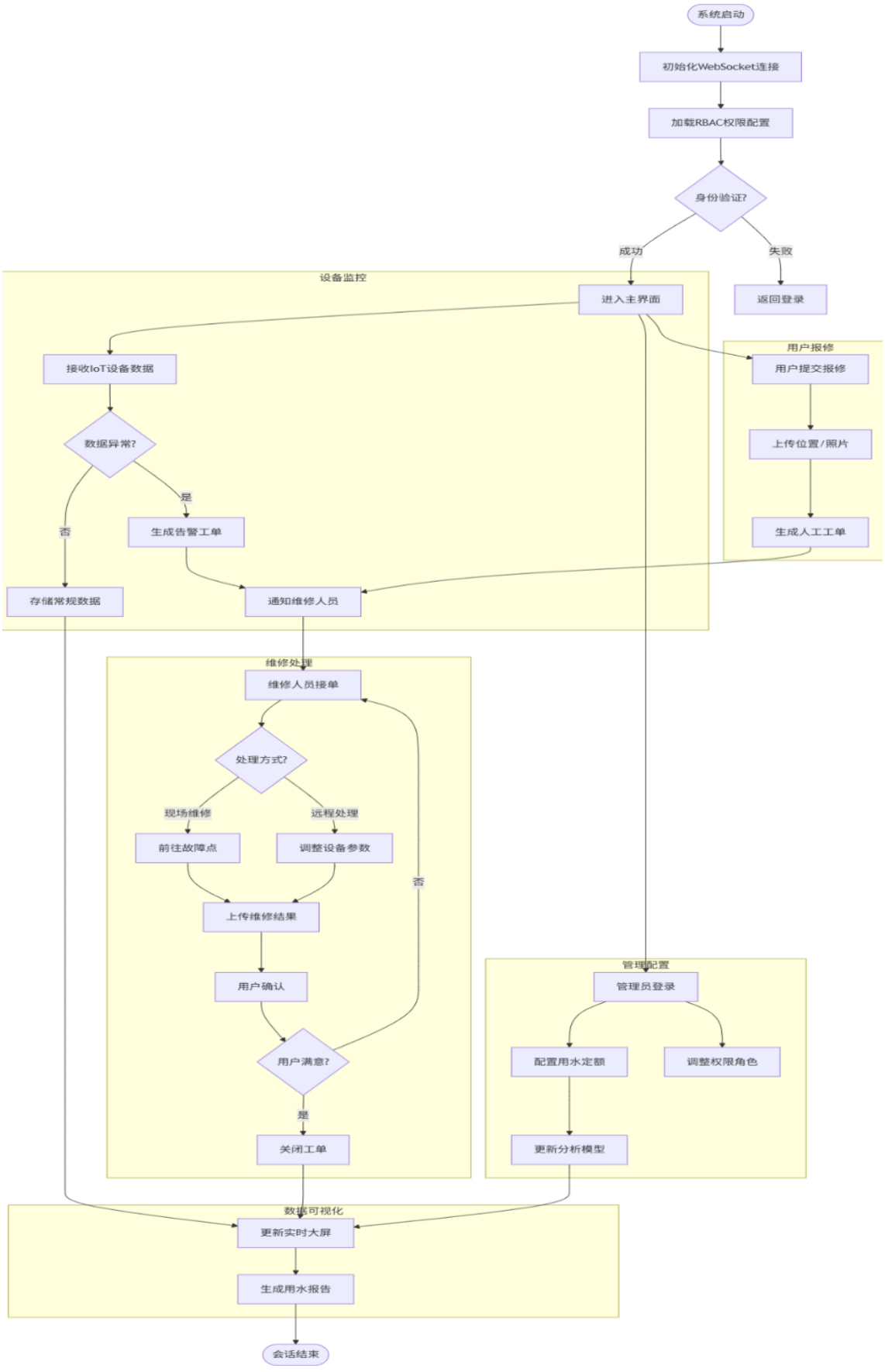
系统启动
初始化WebSocket连接
加载RBAC权限配置
身份验证?
成功
失败
返回登录
进入主界面
设备监控
接收IoT设备数据
数据异常?
是
否
生成告警工单
存储常规数据
通知维修人员
用户报修
用户提交报修
上传位置/照片
生成人工工单
维修处理
维修人员接单
处理方式?
现场维修
远程处理
前往故障点
调整设备参数
上传维修结果
用户确认
用户满意?
否
是
关闭工单
管理配置
管理员登录
配置用水定额
调整权限角色
更新分析模型
数据可视化
更新实时大屏
生成用水报告
会话结束

图 3.8 系统活动图

3.1.4 系统行为分析

1. 系统运行环境分析

本系统部署在典型的 Web 应用架构下，支持多终端访问，主要包括以下运行环境：

客户端环境：Web 端：主流浏览器（Chrome、Edge、Firefox 等）；移动端：Android/iOS 设备，基于 Flutter 框架实现跨平台兼容。服务端环境：操作系统：Linux（CentOS/Ubuntu）或 Windows Server；后端语言与框架：NestJS（Node.js 框架）；数据库：MySQL（关系型）、Redis（缓存）网络环境：支持局域网及广域网访问，前端通过 RESTful API 与后端通信；使用 HTTPS 协议保障数据传输安全

2. 系统使用方式说明

系统面向的用户包括普通用户、管理员、维修人员三类角色。普通用户可以查看水务大屏可视化、提交故障报修、查看用水量统计。管理员用户可以管理用户权限、配置定额用水标准、审核设备报修工单、查看大屏可视化。维修人员可以登录 App 对故障点完成维修上传信息，也是普通用户。

用户通过 Web 界面或移动端 App 登录系统，输入用户名和密码完成身份验证后进入主界面，根据角色获得不同的功能菜单和操作权限。

3. 系统行为建模

用户提交报修请求的完整流程的活动图如下：用户登录系统，进入“故障报修”页面，填写报修信息并提交，系统保存数据并通知管理员和维修人员，管理员审核并处理报修单，用户查看处理结果。如:图 3.9 故障报修流程活动图

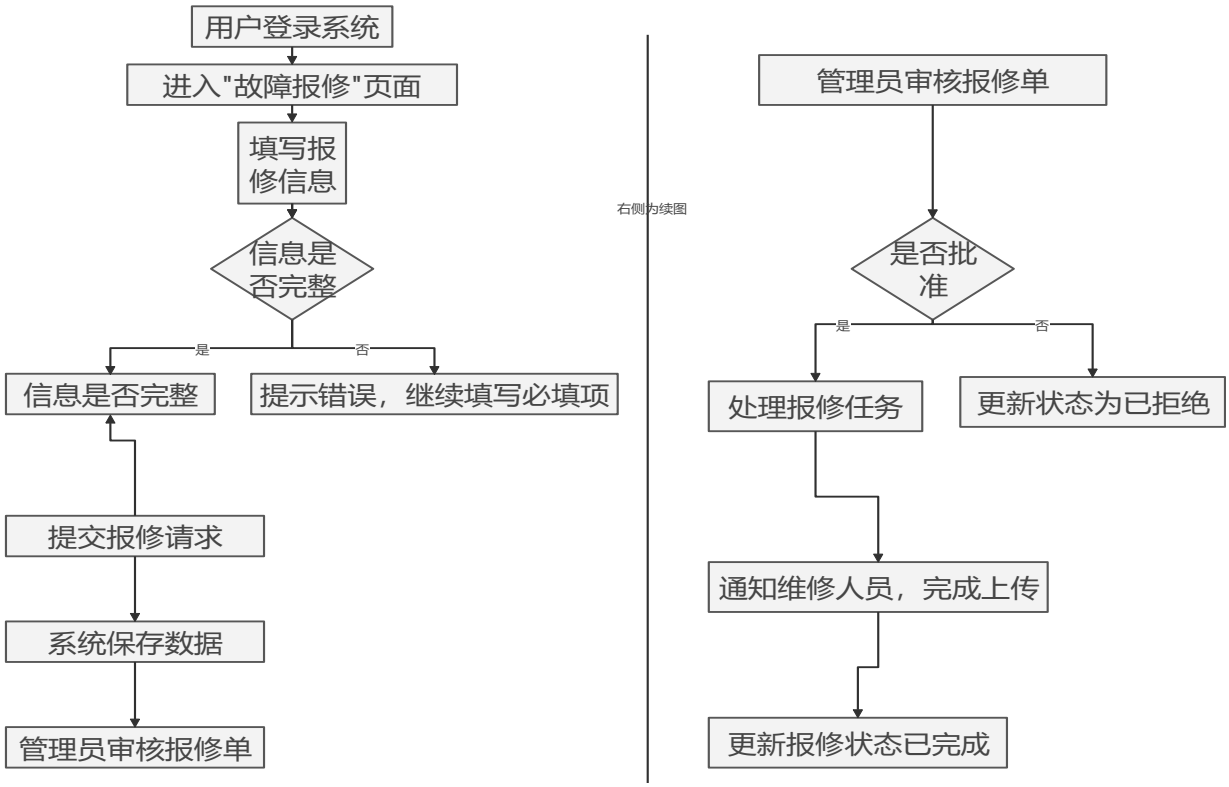

图 3.9 故障报修流程活动图

3.1.5 系统性能分析

该系统作为水务数据监视系统，并不是人们长期使用的服务，服务的用户是大约 10 万人的群体，而使用他的人可能小于 100 人。再此列举分析的应用的性能指标如：表 3.1 性能指标

表 3.1 性能指标

指标类别	具体指标	参考值	说明
响应时间	页面加载时间	≤2 秒（首屏）	用户操作后页面可交互时间
	API 接口响应时间	≤500ms（简单查询） ≤1.5s（复杂操作）	后端接口从请求到返回的时间
	大屏数据刷新延迟	≤5 秒（实时性要求高）	无
吞吐量	并发用户数	≥100 TPS	系统支持的同时在线用户数
指标类别	具体指标	参考值	说明
	事务处理能力（TPS）	≥200（关键业务接口）	如工单提交、设备数据上报等高频操作
可靠性	数据持久化成功率	100%	设备数据、工单记录等关键数据必须持久化
设备监控	物联网设备断线超过 30 分钟	触发三级告警（系统弹窗）。	设备断线提醒
安全性需求	权限分离	基于 RBAC 模型	每个角色有不同的权限
	数据加密	采用 AES-256 加密	用户隐私数据（联系方式、位置）
	防攻击能力	使用 ORM 框架	抵御 SQL 注入、XSS 等常见攻击，每日自动生成安全审计日志。
易用性需求	多端适配	Web 端支持 Chrome/Firefox/Edge，移动端适配 Android	无
	操作简化	用户报修流程≤3 步	无

3.2 系统概要设计与建模

3.2.1 系统概要设计概述

系统基于 MVC 分层架构和模块设计，MVC 框架，即模型（Model）-视图（View）-控制器（Controller），实现业务逻辑、数据与界面显示的分离，业务逻辑放于单独的部件中。这种模式通过降低各模块间的耦合程度，提高了程序的可扩展性和可复用性。本系统服务端设计中采用 MVC 框架，其中，M 代表模型，表示实体对象和数据，业务逻辑层增删查改操作的实体类，在本设计中，用户、水表等实体类都属于模型；V 代表视图，指用户看到并与之交互的界面，将收到的数据可视化后展现给用户，设计中，由 HTML 元素组成的智慧水务平台的网页界面的界面都是视图；C 代表控制器，负责业务逻辑的处理，是模型和视图连接的桥梁，调用模型和视图去完成用户的需求，从模型中获取数据并将数据传递给指定的视图。搭配模块化设计方法，将具有独立逻辑的实体写成一个模块，如果有依赖就导入模块，而不是直接引入 server 层。

智能水务系统的架构被精心设计为三层结构：用户端层、服务层以及数据存储层。每一层都承担着特定的功能，并通过定义明确的接口与其他层交互。用户端层包括 Web 端和移动端。

Web 端 (Nuxt.JS)提供水务大屏可视化、故障报修、定额管理、用水分析设备管理等功能;使用 Nuxt.JS 的路由和组件化结构实现快速加载和易于维护的页面;集成第三方库（如 ECharts）用于数据可视化展示。移动端 (Flutter)仅提供故障报修功能。

服务层就是后端 (NestJS)提供 RESTful API 给 Web 端和移动端使用。处理用户登录认证、权限控制、数据存储、业务逻辑处理等核心功能。使用 TypeORM 框架简化数据库操作。

数据存储层包括 MySQL 和 Redis。关系型数据库 (MySQL)存储系统中的所有数据，包括用户信息、设备状态、用水记录等。支持弹性扩展，满足不断增长的数据需求。缓存(Redis)缓存常用数据，减少数据库查询压力，提高系统响应速度。用于会话管理和临时数据存储，提升用户体验。

各部分的交互和联系有如下两点，用户端与服务层之间的交互是用户通过 Web 端或移动端发起请求，这些请求会被发送到服务层的 API 接口。服务层接收到请求后，根据请求类型调用相应的业务逻辑处理模块，并返回处理结果给用户端。

服务层与数据存储层之间的交互是服务层在处理业务逻辑时需要访问数据存储层获取或更新数据。对于频繁访问的数据，服务层优先从 Redis 缓存中读取；如果缓存未命中，则从关系型数据库中读取，并将结果写入缓存以备后续使用。

3.2.2 系统数据库设计

1. 水务可视化大屏模块

使用 waterDataSummary 表来记录每段时间的总水量，记录平均流量和峰值流量，关联

waterUsageTrend 就可以显示这段时间的用水趋势，包括每日和月份用水趋势。读取 waterDataSummary 表和 waterConservation 可以拿到节水信息，进而计算出年度同比增长信息。waterUsageProportion 是区域用水信息表。数据库表设计如：图 3.10 水务可视化大屏数据库设计

水统计表	water_data_summary	
INT	id	
ENUM	data_type	数据类型
ENUM	time_dimension	时间维度
DATETIME	start_time	开始时间
DATETIME	end_time	结束时间
DECIMAL	total_flow	总用水量(m³)
DECIMAL	avg_flow_rate	平均流量(m³/h)
DECIMAL	peak_flow	峰值流量(m³/h)
DATETIME	peak_time	峰值出现时间
INT	area_id	区域ID
INT	device_id	关联设备ID
INT	organization_id	关联组织ID
DATETIME	created_at	
DATETIME	updated_at	

节水信息	water_conservation	
INT	id	
INT	year	年份
DECIMAL	saved_amount	节水量(m³)
DECIMAL	saving_rate	节水率(%)
ENUM	comparison_base	对比基准
INT	area_id	区域ID
INT	organization_id	关联组织ID
DATETIME	created_at	
DATETIME	updated_at	

区域占比	water_usage_proportion	
INT	id	
ENUM	time_dimension	时间维度
DATE	stat_date	统计日期
INT	area_id	区域ID
DECIMAL	flow_amount	用水量(m³)
DECIMAL	proportion	占比(%)
DATETIME	created_at	

用水趋势	water_usage_trend	
INT	id	
ENUM	trend_type	趋势类型
DATETIME	time_point	时间点
DECIMAL	flow_value	流量值(m³)
INT	area_id	区域ID
INT	device_id	关联设备ID
DATETIME	created_at	

图 3.10 水务可视化大屏数据库设计

2. 故障报修模块

设备故障报修管理是需要手机端扫码报修，上传维修工单到系统，将信息分别存到工单记录表和工单表，然后派发到维修师傅，维修师傅将信息上传到后端。设备故障分上传、维修、审核三个个阶段，但是故障地点和故障类型是一样的，因此使用 workOrder 来记录这些公共信息。记录维修不同时期的维修记录表是 wordOrderRecord 表，wordOrderRecord 还额外记录了审核人和审核日期。数据库关系如：图 3.11 故障维修数据库设计

work_order		
INT	id	
DATETIME	created_at	令牌创建时间
DATETIME	updated_at	
VARCHAR	orderId	工单ID
TINYINT	faultLevel	故障等级(1-5)
VARCHAR	faultLocation	故障地点
VARCHAR	faultType	故障类型
TEXT	faultDesc	故障描述
TINYINT	orderStatus	状态(0:待处理 1:处理中 2:待审核 3:已通过 4:已驳回)
INT	device_id	

work_order_records		
INT	id	
DATETIME	created _at	令牌 创建时间
DATETIME	updated_at	
ENUM	type	记录类型(report:上报 process:处理中 complete:完成 reject:驳回 approve:审核通过)
JSON	images	图 片 URL数组
VARCHAR	location	维修地点
TEXT	description	维修描述
DATETIME	processTime	处理时间
DATETIME	completeTime	完成时间
DATETIME	reviewTime	审核时间
INT	operator_id	
INT	reviewer_id	
INT	work_ order_id	

图 3.11 故障维修数据库设计

3. 用水分析模块

程序分析之后结果存在分析结果字段，记录了分析的开始时间和结束时间、分析周期，还有这段时间的总用水量、平均流量、峰值时间及流量。左连接了设备表、区域表、组织表用于程序分析。数据库关系如：图 3.12 用水分析数据库设计

water_usage_analysis		
INT	id	
DATETIME	created_at	
DATETIME	updated_at	
ENUM	analysis_period	分析周期
DATETIME	start_time	开始时间
DATETIME	end_time	结束时间
DECIMAL	total_flow	总用水量(m^3)
DECIMAL	avg_flow_rate	平均流量(m^3/h)
DATETIME	peak_flow_time	峰值流量时间
DECIMAL	peak_flow_value	峰值流量值(m^3/h)
INT	device_id	关联设备ID
INT	area_id	关联区域ID
INT	organization_id	关联组织ID
JSON	analysis_result	分析结果(JSON格式)
TINYINT	abnormal_flag	异常标志(0正常 1异常)
VARCHAR	abnormal_desc	异常描述

图 3.12 用水分析数据库设计

4. 定额管理模块

定额管理记录是定额名称，定额可以给区域和设备和组织设置额度，可以设置周期额度，就是每个月最多使用多少水，超过就报警，预警阈值超过提醒。额外记录了创建人时间。数据库关系如:图 3.13 定额管理数据库设计

water_quota_management		
INT	id	
DATETIME	created_at	
DATETIME	updated_at	
VARCHAR	quota_name	定额名称
ENUM	quota_type	定额类型(区域/设备/组织/用户)
DECIMAL	quota_value	定额值(m^3)
ENUM	quota_period	定额周期
DATE	start_date	开始日期
DATE	end_date	结束日期(空表示永久有效)
TINYINT	status	状态(0禁用 1启用)
DECIMAL	alert_threshold	预警阈值(%)
INT	related_id	关联ID(根据类型关联区域/设备/组织/用户)
VARCHAR	remark	备注
INT	create_by	创建人
INT	update_by	更新人

图 3.13 定额管理数据库设计

5. 设备管理模块

分析设备管理，需要注意，水表设备肯定一直通过 MQTT 一直发消息，就有数据库选型的问题，对于数量小，但是记录多数据，设备数量不超过 10000 台，优先选择 MySQL，适合小规模数据。然后设备都有相同的基础信息和特别信息，使用一个公共表来存相同字段信息，最为合适，减轻数据的重复。数据库关系如：图 3.14 设备信息表数据库设计

device_base		
INT	id	
DATETIME	created_at	
DATETIME	updated_at	
VARCHAR	deviceCode	
TINYINT	deviceType	设备类型
VARCHAR	manufacturer	生产厂家
TINYINT	status	设备状态(1:正常 2:预警 3:故障)
VARCHAR	model	设备型号
VARCHAR	gatewayId	网关ID
INT	area_id	

pressure_meter		
INT	id	
TIMESTAMP	uploadTime	
DECIMAL	longitude	经度
DECIMAL	latitude	纬度
VARCHAR	caliber	设备口径
INT	device_base_id	
VARCHAR	pressure	压力值

water_meter		
INT	id	
TIMESTAMP	uploadTime	
VARCHAR	flowRate	瞬时流量
VARCHAR	totalFlow	累计流量
VARCHAR	signalStrength	信号强度
INT	device_base_id	

图 3.14 设备信息表数据库设计

6. 登录模块

登录为了防止机器登录，设置图像验证码，对每次登录产生验证码的行为记录。使用

JWT，记录了用户登录之后的 token，可以看出是否多次登录。对数据结构建模 Token 表设计如:图 3.15 登录认证数据库设计

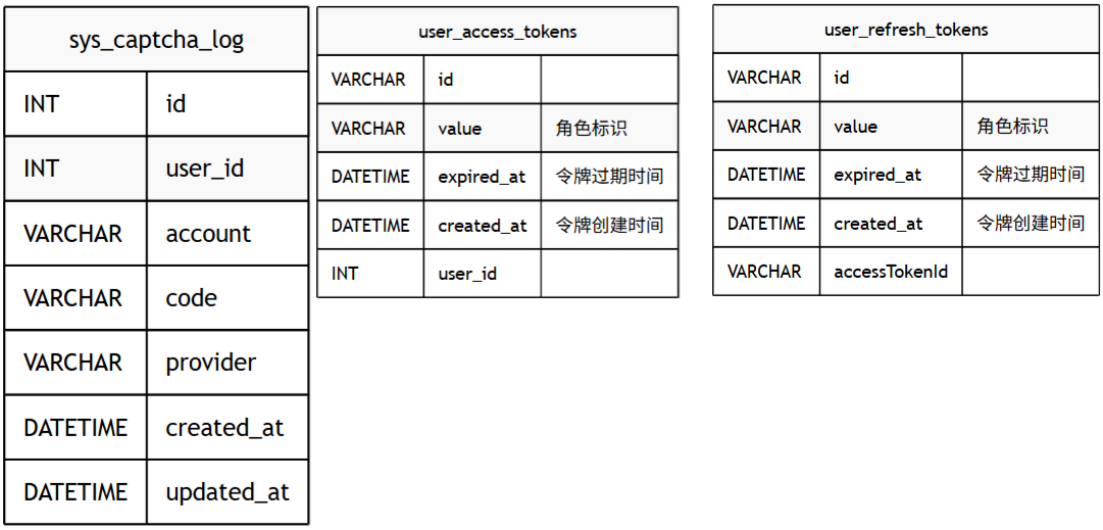

图 3.15 登录认证数据库设计

3.2.3 系统软件架构设计

智能水务系统的软件架构被精心设计，旨在确保高效性、可扩展性和易维护性。由 Web 端（Nuxt.js）、后端（NestJS）和移动端（Flutter）三大部分构成，结合 RESTful API + WebSocket 实现高效通信，确保系统的高效性、可扩展性和易维护性。

前端架构设计：Web 端的技术栈：Vue3 + Nuxt3 + Pinia（状态管理） +UnoCSS（样式）。功能职责负责提供响应式 Web 管理后台，支持 PC 访问；采用 Nuxt.js 的服务端渲染（SSR），提升 SEO 优化和首屏加载速度；集成 Pinia 管理全局状态（如用户登录、水表数据缓存）；使用 Axios 与后端 API 交互，实现动态数据加载（如水表实时监测、用户报表）。移动端（Flutter）的技术栈：Dart + Flutter + Provider（状态管理）。功能职责包括提供 Android 原生 App，支持水表数据查询、在线缴费、故障报修等功能；采用 Flutter 的跨平台能力减少开发成本，同时保证 Native 级性能；集成 WebSocket 实现实时水表数据推送（如漏水预警、用水量异常提醒）。优化策略包括自适应布局（响应式设计）适配不同屏幕尺寸。

后端架构设计:后端的技术栈有 NestJS（Node.js 全栈框架）+TypeScript+JWT+Redis +MySQL。服务模块划分为：用户认证服务（AuthService）：负责用户注册、登录、权限管理（RBAC 角色控制）；水表服务（MeterService）：管理水表数据（增删改查）、实时数据采集（MQTT/HTTP）；告警服务（AlertService）：监测异常用水（如漏水、超额用水），触发短信/APP 推送。通信方式包括：RESTful API：提供标准化接口供前端调用（如/api/meters 查询水表列表）和 WebSocket 用于实时数据推送（如水表流量变化、告警通知）。数据持久层有 MySQL：存储核心业务数据（用户、水表、订单）；Redis：缓存热点数据（如 JWT 令牌、水表实时状态）。

3.2.4 系统用户界面原型设计

原型设计是软件开发过程中的一个关键步骤，它通常在需求分析之后，正式编码之前进行。原型设计的目的是创建一个可视化的、交互式的软件应用模型，这个模型展示了应用的布局、功能和用户交互流程。本系统利用“MasterGo”实现原型设计，各页面原型设计图如下：

数据可视化界面原型图和设备管理界面原型图如：

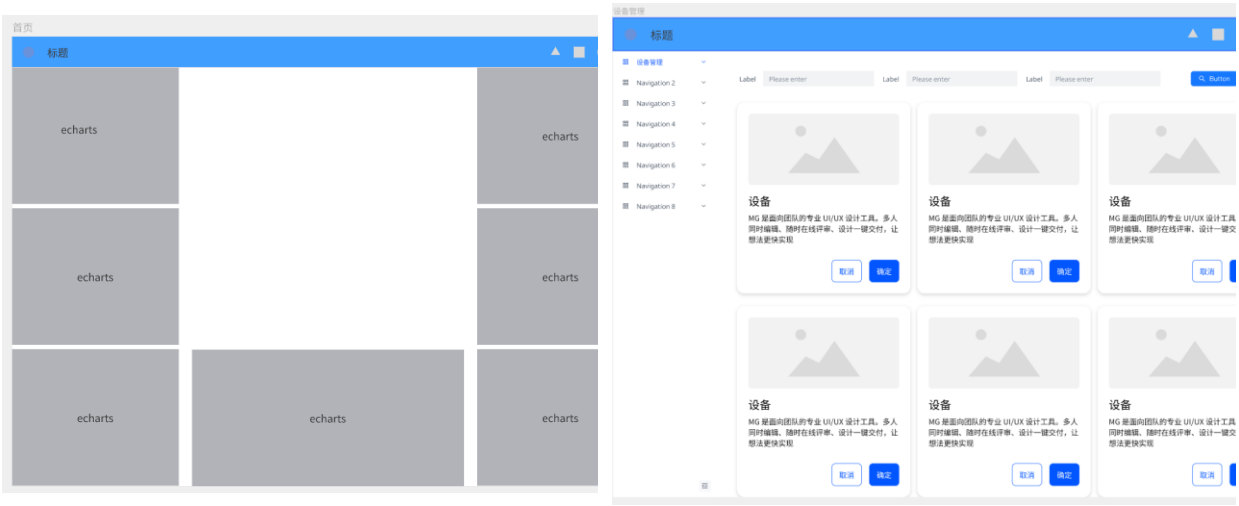

图 3.16 数据可视化和设备管理界面原型图

设备管理界面抽屉原型图:

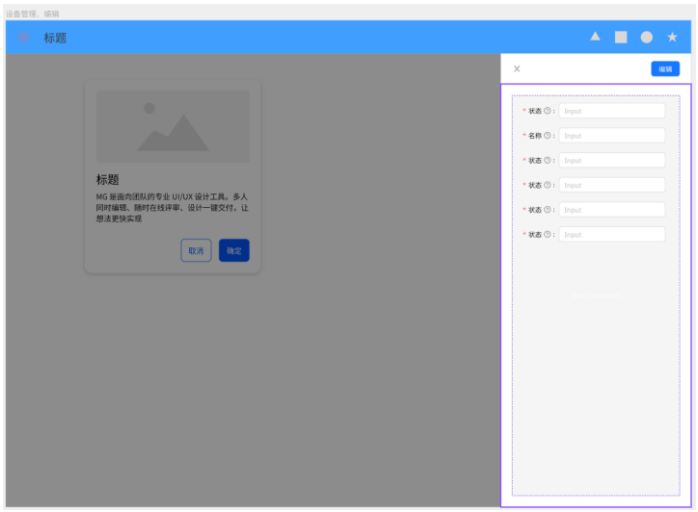

图 3.17 设备管理界面抽屉原型图

其他管理和上面界面相似;管理端，比如设备管理、地区管理、角色管理、工单管理，基本采用左侧导航栏；右侧抽屉，负责点击编辑信息；中间展示展示表格或者图表。

移动端原型设计

使用 MasterGo 原型设计工具，完成基础的报修通知功能设计如:图 3.18 flutter 设计图

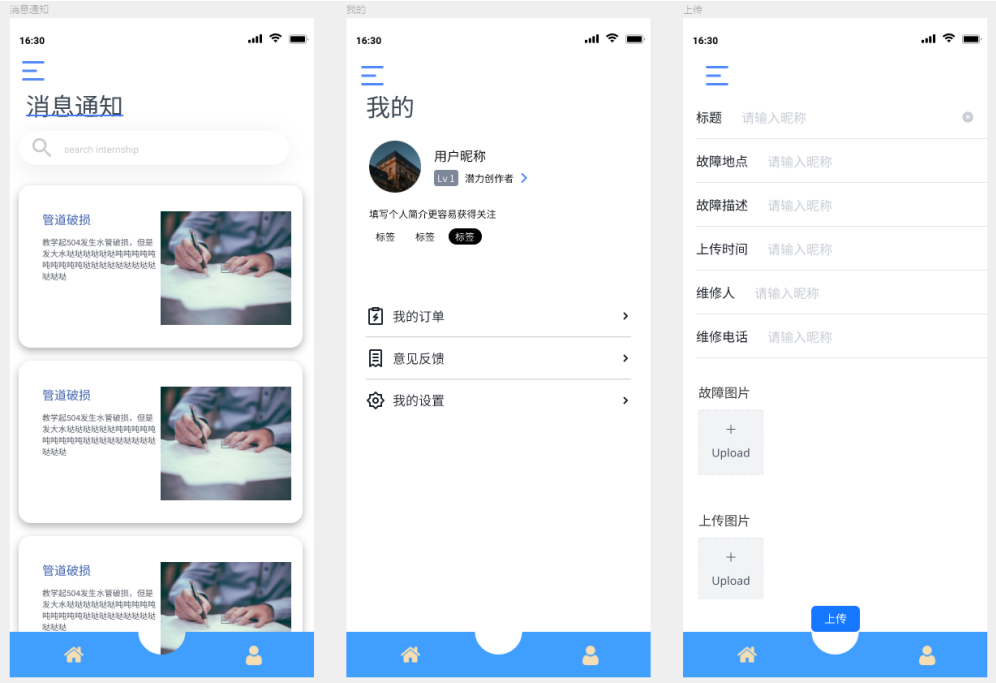

图 3.18 flutter 设计图

3.3 系统详细设计与建模

3.3.1 智能水务 Web 系统可视化的设计

为了水表信息更新之后实时同步到前端，采用 WebSocket 技术，来传前端的可视化大屏的信息。WebSocket 连接时序图如：

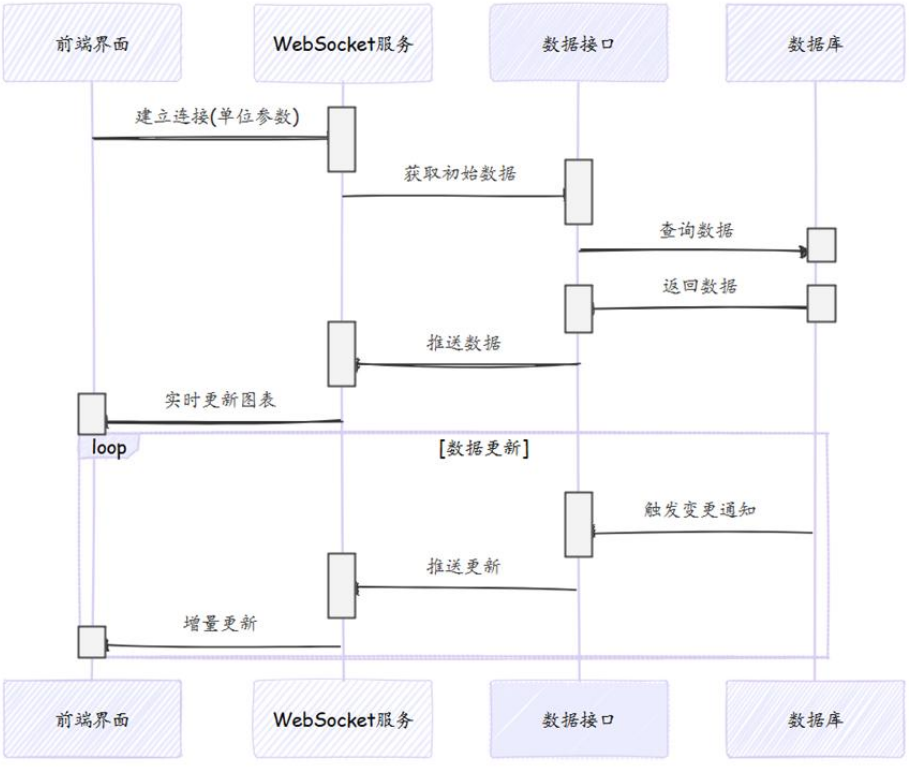

图 3.19 WebSocket 时序图

设计 7 个核心 WebSocke 接口，为首页大屏可视化提供数据支持,如：表 3.2 接口设计示例

表 3.2 接口设计示例

接口	功能描述
ws://api/water/realtime	实时用水量：包含今日、本月、本年、上月
ws://api/water/area?areas=cafeteria,teaching,dormitory,commercial	各区域实时用水占比
ws://api/water/monthly?year=2025	某一年每月用水量趋势
ws://api/water/annual?years=5	近 5 年年度用水总量，用于同比增长对比
ws://api/alerts/realtime	实时报警信息推送
ws://api/device/status	所有设备状态（在线/离线/故障等）

界面需要用到百分比，需要获取定额配置，用水额度通过 http 协议请求获取，存到本地的 Pinia 中。获取到每天水量除以额度，使用 Math.floor 向下取整，显示百分比。

3.3.2 设备管理设计

设备的基础信息与水表和压力表都是一对多关系，当前端添加设备，需要后端去存，并且返回所有的设备列表，完成设备信息的更新，删除。设备与区域的多对一的关系，确保这个设备只能被该组织用户访问到。并且在大屏展示，还需要返回设备的相关水数据。

3.3.3 故障报修管理的设计

报修设计为工单系统，连接一个可以追溯一个设备的维修的生命周期。工单一对多连接设备报修记录，保修记录和审核人员一对多连接系统用户。完成工单的创建，完成，更新阶段。由移动端，完成扫码报修，上传工单信息，将工单信息列表提供给维修人员，维修人员完成维修，扫码上传完成，工单状态由待处理，到待审核，最后完成维修。

3.3.4 移动端扫码报修设计

本模块为移动端提供扫码报修功能。用户可通过扫描设备二维码获取设备编号或地址信息，自动填充部分表单内容，并手动填写故障描述后提交报修请求。

技术选型如下：UI 框架 Flutter 自带，但是状态管理需要使用 GetX，网络请求使用 Dio 发起请求,访问后端,表单验证使用自带功能。需要拍照,使用 flutterBarcodeScanner 来完成拍照。

3.3.5 用户登录设计

现如今，登录已经非常成熟，很多公司都用了单点登录，统一管理一个用户系统。对于单个系统，比较适合流行的 JWT 和 cookie 等技术。登录流程序列图如：图 3.20 登录流程序列图

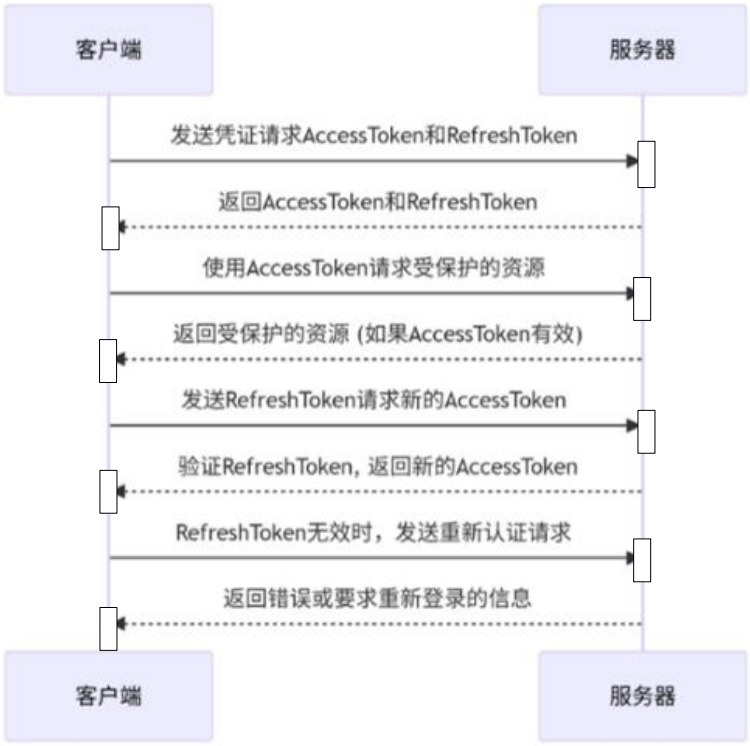

图 3.20 登录流程序列图

特点在于，一个 Accesstoken 和 RreshToken 保存时间不一样，当访问服务器，发现 Accesstoken 过期的话，前端使用 RreshToken 去请求换取 Accesstoken，后端查询 RreshToken，如果 RreshToken 正确有就刷新 Accesstoken 返回给前端。这样可以轻松使用户下线，因为 RreshToken 保存在后端，可以随时修改，等用户 Accesstoken 过期，就只能访问后端。

Accesstoken 用于每次请求时的身份验证。由于其较短的有效期，可以减少被滥用的风险。

Refresh Token 具有较长有效期，主要用于在访问令牌过期后获取新的访问令牌，而无需用户重新登录。

3.3.6 RBAC 权限模型设计

围绕设备、人员、组织、地区可如下设计数据库，需要满足，系统由超级管理员管理，其他用户只能看自己所属的组织的下的信息，地区和组织是多对一的关系， 用户和组织是多对一的关系，用户和角色是一对多的关系，设备和地区是多对一的关系。

管理员:拥有完整系统权限,包括用户管理、设备管理、故障报修、定额配置管理等,并可进行系统参数配置。

维修人员： 主要负责接收并处理用户的故障报修,可以查看维修任务详情,并进行维修记录和进度更新。

普通用户:可以访问水务资讯模块,并提交设备故障报修。

3.4 本 章 小 结

本章围绕水务管理系统的核心业务场景大屏可视化、设备管理、故障报修、定额配置、

访问控制，完成以下工作：

围绕故障报修、设备管理和定额配置三大业务流程，梳理了各模块的功能需求，完成了设备报修的 UML 的序列图。

基于 RBAC（基于角色的访问控制）模型，将用户划分为管理员、维修人员和普通用户三类角色，并明确了各类角色的操作权限边界。

考虑了系统的性能、安全性、成本及部署环境，提出了对高并发处理能力、数据保护机制、资源优化等方面的明确要求

考虑了系统设计之初即考虑行业规范与未来物联网设备接入的可能性，预留了标准化接口。

在系统架构方面，Web 端使用 Nuxt3 实现服务端渲染（SSR），提升首屏加载速度。

4 系统实现

4.1 实现环境与工具

本系统采用 VSCode + Docker + Git 的现代化开发工具链，通过容器化技术保证环境一致性，Apifox 实现接口文档自动化管理，Navicat 辅助数据建模。这一工具组合不仅提升开发效率，更符合软件工程的标准化要求，为后续系统维护和扩展奠定基础。工具列表如：表 4.1 开发工具列表

表 4.1 开发工具列表

工具	作用
VSCode	全栈开发、插件生态
Docker	环境隔离、一键启动
Apifox	接口文档自动化
Git	版本控制、代码回溯
工具	作用
VSCode	全栈开发、插件生态

4.1.1 开发工具链

代码编辑器采用 Visual Studio Code (VSCode) 作为核心开发工具，其轻量级、插件生态丰富（如 ESLint、Prettier、Volar、REST Client 等，支持全栈开发（前端、后端、数据库脚本），实现"万物皆可 VSCode"的高效编码体验。前端开发优使用 Volar 插件提供 Vue/Nuxt3 的智能提示，UnoCSS IntelliSense 加速样式开发。后端开发优化使用 Apifox 直接调试 API。数据库管理使用 Docker Compose 容器化部署数据库（如 MySQL），确保环境一致性，避免本地污染。通过 Navicat 图形化工具直观管理数据表结构和查询结果，提升开发效率。通过 Apifox 管理接口文档、Mock 数据及自动化测试，实现"代码未动，文档先行"，降低前后端联调成本。版本控制与协作使用 Git 作为版本控制工具，配合 GitHub 托管代码，规范提交日志（如 feat:, fix:, docs:），便于回溯。在代码出现的失误的时候，能有效溯源回退代码。

4.2 智能水务大屏可视化系统的实现

1. 界面

本模块是智能水务系统的交互页面，用户可以查看实时用水数据、历年用水量数据、用水占比等信息。页面中展示了今日水量、本月水量、本年水量等详细数据，并通过图表和进度条直观显示各类用水情况及趋势分析。

采用响应式网格布局（Grid Layout）实现 3×3 可视化矩阵，其中：中央主视图区：跨 3 列显示核心趋势图表（年度用水量）；周边 6 个子模块：等分布局展示关键指标（2 行

×3 列）；组件化封装：通过<ChartCard>统一容器实现，如：图 4.1 智能水务大屏可视化界面

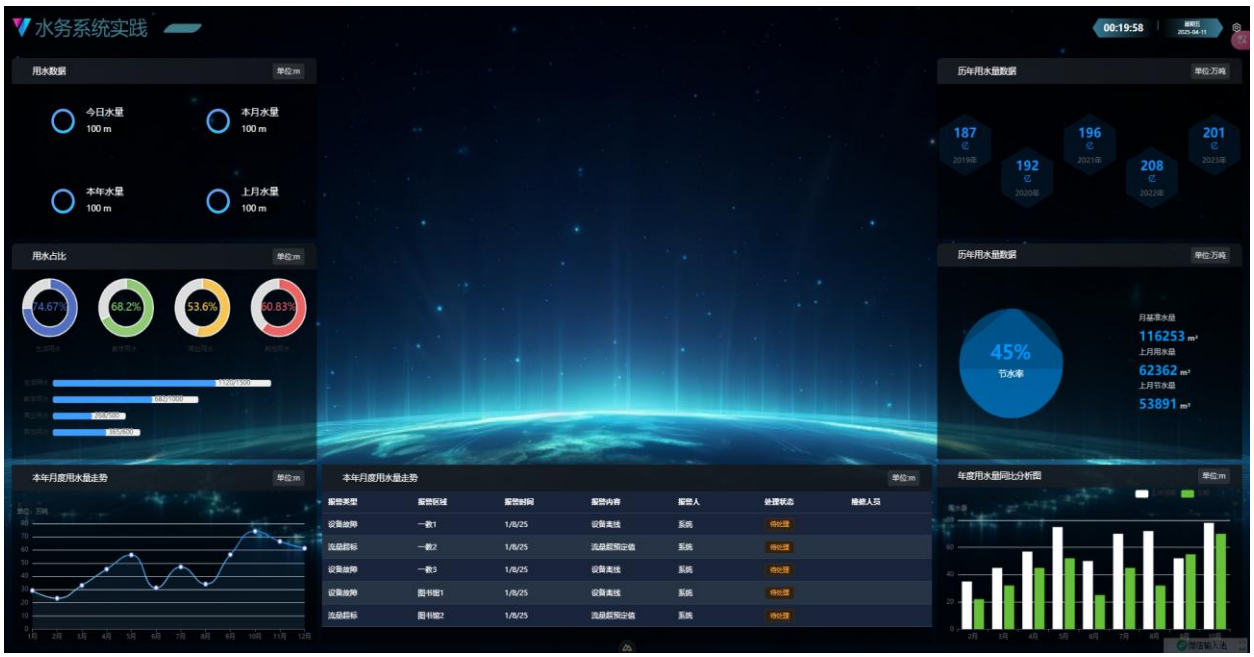

图 4.1 智能水务大屏可视化界面

2. 关键代码

对界面 7 个部分封装层组件，因为彼此没有强关联，数据接口写在每个组件内部。如：图 4.2 可视化代码图 4.3 可视化代码

```
<template>
  <div class="page-container">
    <div class="grid-container">
      <DashCard title="用水数据" data="ds" unit="m³" height="100%" class="grid-item">
        <div class="grid grid-cols-2 grid-rows-2 h-100% w-100%">
          <!-- 用水数据 -->
          <EchartsComponentsUserWaterItem
            v-for="(item, index) in data" :key="index" :title="item.name"
            :content="item.content" :unit="item.unit"
          />
        </div>
      </DashCard>
      <div class="grid-item bgAlpha">
        <!-- 用水占比 -->
        <EchartsWaterProportion />
      </div>
      <div class="grid-item bgAlpha">
        <!-- 用水趋势 -->
        <EchartsAnnualTrends />
      </div>
      <div class="grid-item bgAlpha">
        <!-- 用水历史 -->
        <EchartsHistoryWater />
      </div>
      <div class="grid-item bgAlpha">
        <!-- 本月水量 -->
        <EchartsLastMonth />
      </div>
      <div class="grid-item bgAlpha">
        <!-- 年度用水量同比分析图 -->
        <EchartsYearoveryearMonthlyWater />
      </div>
      <div class="grid-item grid-item-bottom bgAlpha">
        <EchartsAlarmInfo />
      </div>
    </div>
  </div>
</template>
```

图 4.2 可视化代码

```
onMounted(() => {
  // 实时用水量
  connectWebSocket('ws://api/water/realtime', (data) => {
    realtimeData.value = {
      today: data.today,
      month: data.month,
      year: data.year,
      lastMonth: data.lastMonth,
    }
  })
  // 区域用水占比
  connectWebSocket('ws://api/water/area?areas=cafeteria,teaching,dormitory,commercial', (data) => {
    areaUsage.value = data
  })
  // 月趋势
  connectWebSocket('ws://api/water/monthly?year=2025', (data) => {…
  })
  // 历年对比
  connectWebSocket('ws://api/water/annual?years=5', (data) => {…
  })
  // 报警
  connectWebSocket('ws://api/alerts/realtime', (data) => {…
  })
  // 设备状态
  connectWebSocket('ws://api/device/status', (data) => {…
  })
  // 节水建议
  connectWebSocket('ws://api/calc/saving', (data) => {…
  })
})
// 页面卸载关闭连接
onUnmounted(() => {…
})
```

图 4.3 可视化代码

在组件内部的 script 部分，定义一个界面需要使用的数据容器响应式变量，创建 WebSocket 实例集合。封装协议的连接、消息、错误、关闭方法。分别请求使用 WebSocket 协议，请求后端获取数据。在组件销毁时，把 WebSocket 连接关闭。

4.3 设备管理

1. UI 设计

设备管理界面顶部可以添加设备，右边可以搜索设备，点击卡片有动态效果，点击编辑可以编辑设备信息，比如编号、设备状态、设备绑定底子等。如：图 4.4 设备管理 UI

图 4.4 设备管理 UI

2. 代码实现

模拟的水表数据产生如下:首先需要满足定时，第二个延迟就是定时。当前时间加上持续时间就是结束时间。先找到要模拟的设备，触发模拟。模拟逻辑是根据数据库的表字段，加上数据乘以一个基数最后保留两位生成。代码展示如：图 4.5 设备管理模拟生成数据代码 和图 4.6 设备管理模拟核心代码

```
@Injectable()
export class DeviceMockService implements OnModuleDestroy {
  private activeMocks = new Map<number, { timer: NodeJS.Timeout, endTime: number }>()

  constructor( ···
  ) { }

  async generateDeviceData(deviceIds: number[], duration: number): Promise<void> {
    try {
      const endTime = Date.now() + duration

      await Promise.all(deviceIds.map(async (deviceId) => {
        try {
          const device = await this.deviceRepo.findOne({
            where: { id: deviceId },
            select: ['id', 'deviceType'],
          })
          if (!device) { ···
          }
          // 初始生成数据
          await this.generateDataByType(deviceId, device.deviceType)
          // 启动定时模拟
          const timer = setInterval(async () => { ···
          }, 10000)
          this.activeMocks.set(deviceId, { timer, endTime })
        }
        catch (err) {
          console.error(`设备 ${deviceId} 模拟初始化失败:`, err)
        }
      }))
    }
    catch (err) { ···
    }
  }

  // 根据设备类型模拟数据      You, 1秒钟前 • Uncommitted changes
  private async generateDataByType(deviceId: number, deviceType: number) { ···
  }

  // 停止模拟
  private stopMockForDevice(deviceId: number) { ···
  }
```

图 4.5 设备管理模拟生成数据代码

```
// 生成水表设备数据
async generateFlowData(deviceId: number) {
    const baseFlow = Math.random() * 10 + 5
    const signalStrength = Math.random() * 100

    await this.waterMeterRepo.save({
        device: { id: deviceId },
        flowRate: `${baseFlow.toFixed(2)}`,
        totalFlow: `${(baseFlow * 0.1).toFixed(2)}`,
        signalStrength: `${signalStrength.toFixed(2)}%`,
        uploadTime: new Date(),
    })
}

// 生成压力设备数据
async generatePressureMeterData(deviceId: number) {
    const basePressure = Math.random() * 5 + 1
    const longitude = 116.404 + Math.random() * 0.01
    const latitude = 39.915 + Math.random() * 0.01

    await this.pressureMeterRepo.save({
        device: { id: deviceId },
        pressure: `${basePressure.toFixed(2)}`,
        longitude,
        latitude,
        uploadTime: new Date(),
    })
}
```

图 4.6 设备管理模拟核心代码

4.4 故障报修管理的实现

1. 故障管理 UI，主要实现，管理上传的维修工单，可以显示故障地点、故障描述、审核人、故障类型工单编号。

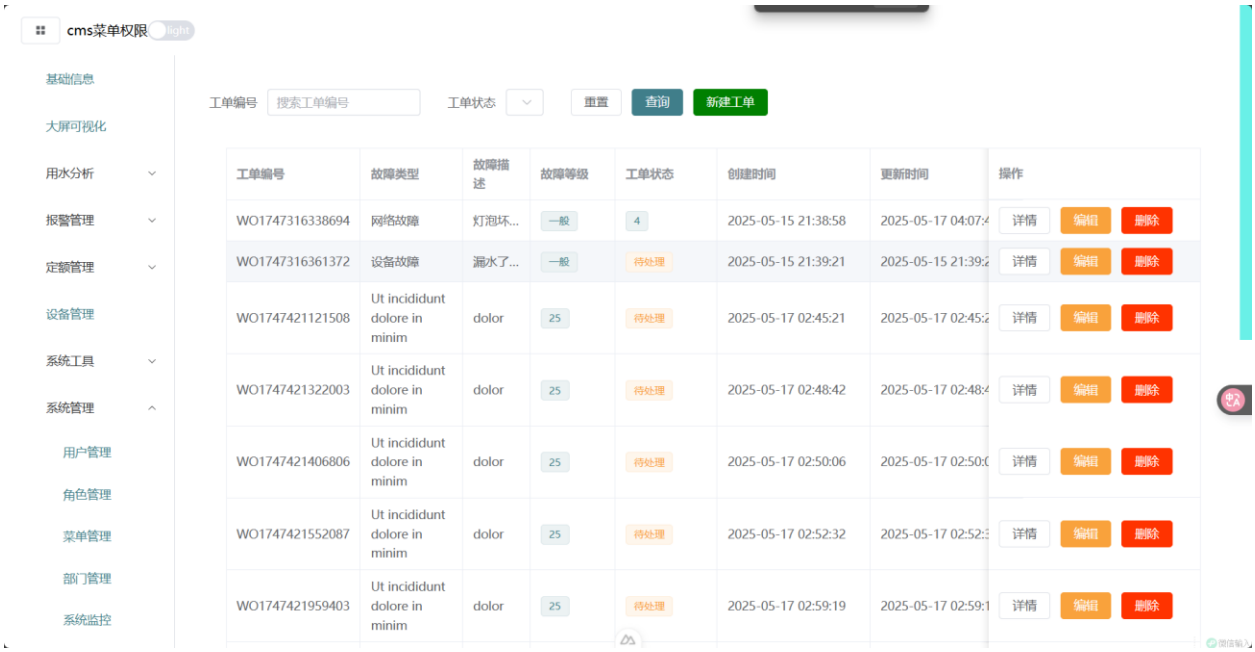

图 4.7 故障管理 UI

2. 下面展示扫码报修，分别有分页获取维修工单、根据条件查询工单、创建维修工单上传。如下：图 4.8 故障维修 controller

```
// 不需要认证
@ApiTags('扫码报修-维修工单')
@Controller('work-orders')
export class WorkOrderController {
  constructor(private readonly workOrderService: WorkOrderService) { }
  // 获取所有的工单
  @ApiOperation({ summary: '获取所有维修工单' })
  @ApiResult({ type: [WorkOrderEntity] })
  @Get()
  async findAll() {
    return this.workOrderService.findAll()
  }

  @ApiOperation({ summary: '根据条件查询维修工单' })
  @ApiResult({ type: [WorkOrderEntity], isPage: true })
  @Get('/list')
  async list(@Query() dto: QueryWorkOrderDto & { page: number, pageSize: number }) {
    return this.workOrderService.list(dto)
  }

  // 创造工单
  @ApiOperation({ summary: '上报维修地点' })
  @UseGuards(LocalGuard)
  @Post('report')
  async report(
    @Body() dto: CreateWorkOrderDto,
    @AuthUser('uid') userId: number,
  ) {
    return this.workOrderService.createRepairOrder(dto, userId)
  }
```

图 4.8 故障维修 controller

展示了故障维修的写法，获取数据库中所有数据，把来自 controller 层的数据使用扩展运算符结构出来，构建 ORM 查询器 queryBuilder，使用 like 模糊查询，满足有参数搜索，没有参数就是访问列表数据，最后返回的数据使用 pagination 中间件统一包裹返回。如:图 4.9 故障维修工单列表 service

```
/**
 * 分页工单信息查询
 *
 */
async list({ page, pageSize, ...data }: QueryWorkOrderDto & { page: number, pageSize: number }) {
  const queryBuilder = this.workOrderRepository.createQueryBuilder('order')
    .leftJoinAndSelect('order.records', 'record') // 关联查询
    .where({
      ...(data.orderId ? { orderId: data.orderId } : null),
      ...(data.faultLocation ? { faultLocation: Like(`%${data.faultLocation}%`) } : null),
      ...(data.orderStatus ? { orderStatus: data.orderStatus } : null),
    })
  return paginate<WorkOrderEntity>(queryBuilder, { page, pageSize })
}

async createRepairOrder(dto: CreateWorkOrderDto, userId: number) {
  const orderNo = `WO${Date.now()}`

  // 创建工单
  const workOrder = this.workOrderRepository.create({…
  })
  const savedOrder = await this.workOrderRepository.save(workOrder)
  // 创建上报记录
  await this.recordRepository.save({
    type: 'report',
    images: dto.imagesUrl || [],
    location: dto.faultLocation,
    description: dto.faultDesc,
    operator: { id: userId },
    workOrder: { id: savedOrder.id },
  })

  return savedOrder
}
```

图 4.9 故障维修工单列表 service

4.5 移动端扫码保修的实现

Flutter 是声明式 UI，完成表单上传，先使用 Scaffold 包裹，内部使 ListView 组件构造表单，使用 TextFormFied 来写标题，比如地址区域、故障描述等。如：图 4.10 flutter 代码

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text('故障报修')),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Form(
        key: _formKey,
        child: ListView(
          children: [
            // 设备扫码输入
            TextFormField(
              controller: _deviceCodeController,
              decoration: InputDecoration(
                labelText: '设备编号',
                suffixIcon: IconButton(···
                ),
              ),
              validator: (value) =>···
              readOnly: true,
            ),
            SizedBox(height: 20),
            // 故障描述
            TextFormField(
              controller: _descriptionController,
              decoration: InputDecoration(
                labelText: '故障描述',
                hintText: '请详细描述故障现象...',
              ),
              maxLines: 4,
              validator: (value) =>···
            ),
            SizedBox(height: 20),
            // 图片上传
            Text('故障照片 (必传)', style: TextStyle(fontWeight: FontWeight.bold)),
            SizedBox(height: 10),
            Wrap(···
            ),
            SizedBox(height: 30),
            // 提交按钮
            ElevatedButton(
              onPressed: _isSubmitting ? null : _submitReport,
              child: _isSubmitting
                ? CircularProgressIndicator(color: Colors.white)
                : Text('提交报修'),
              style: ElevatedButton.styleFrom(···
              ),
```

图 4.10 flutter 代码

4.6 用户登录的实现

输入账号密码，换去 token，使用 useVue 工具提供的 localStorage 去存 token，并且可以设置过期时间。

在 Controller 层拿到请求的实体，比如账号、密码、验证码等，从 request 对象中取出用户浏览器信息，现验证验证码是否正确，将参数传入 token 模块，验证数据库的 md5 加密密码，通过在数据库产生一条 accesstoken 生产信息，返回前端。代码展示如：图 4.11 登录注册代码

```
/**
 * 获取登录JWT
 * 返回null则账号密码有误，不存在该用户
 */
async login(
  username: string,
  password: string,
  ip: string,
  ua: string,
): Promise<LoginToken> {
  const user = await this.userService.findUserByUserName(username)
  if (isEmpty(user))
    throw new BusinessException(ErrorEnum.INVALID_USERNAME_PASSWORD)

  const comparePassword = md5(`${password}${user.psalt}`)
  if (user.password !== comparePassword)
    throw new BusinessException(ErrorEnum.INVALID_USERNAME_PASSWORD)
  const roleIds = await this.roleService.getRoleIdsByUser(user.id)
  const roles = await this.roleService.getRoleValues(roleIds)
  // 包含access_token和refresh_token
  const token = await this.tokenService.generateAccessToken(user.id, roles)
  await this.redis.set(genAuthTokenKey(user.id), token.accessToken, 'EX', this.securityConfig.jwtExprire)
  // 设置密码版本号 当密码修改时，版本号+1
  await this.redis.set(genAuthPVKey(user.id), 1)
  // 设置菜单权限
  const permissions = await this.menuService.getPermissions(user.id)
  await this.setPermissionsCache(user.id, permissions)
  await this.loginLogService.create(user.id, ip, ua)
  return token
}
```

图 4.11 登录注册代码

4.7 RBAC 权限设计的实现

1. UI 实现，使用了 ElementUI 框架的菜单、树形列表，抽屉，实现了编辑、详细 UI 的复用。如:

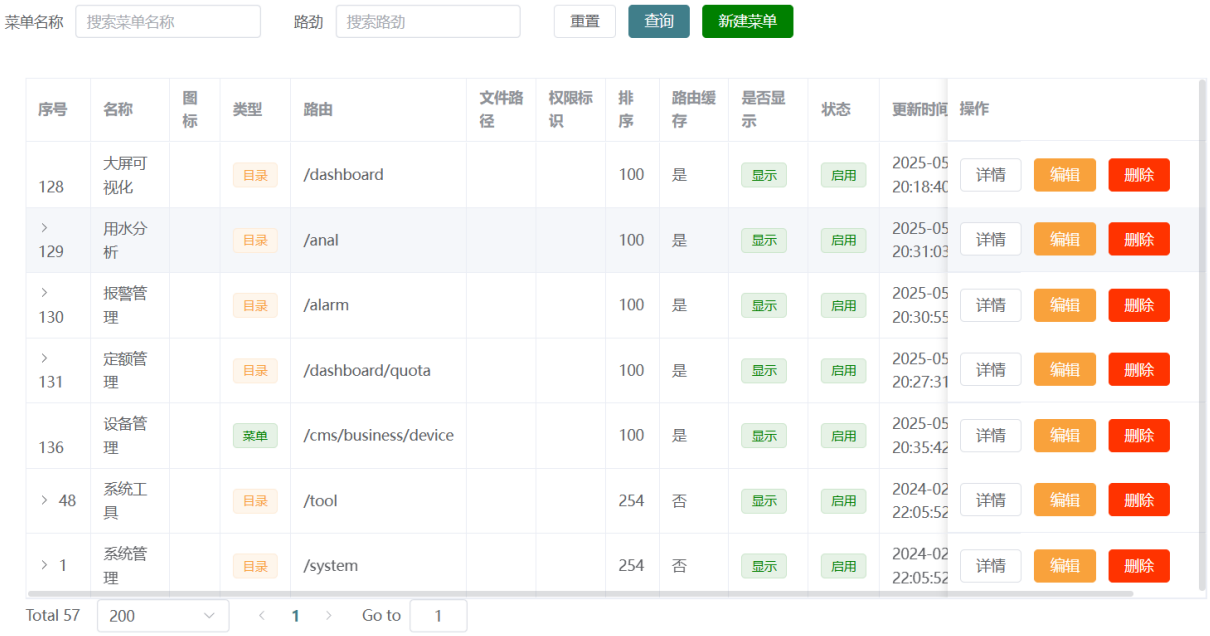

图 4.12 菜单管理

点击编辑按钮，能选择目录还是菜单还是权限，可以添加排列优先级，可以切换状态、是否显示、添加路由与前端形成动态路由。

使用 UI 组件完成角色管理界面，经典左右布局，增删改使用抽屉实现，点击抽屉，可以为角色添加权限，也可以添加角色。如：

图 4.13 角色管理

4.8 本章小结

本智能水务系统通过前后端分离架构实现了多终端协同工作，主要技术成果包括：

架构设计方面，采用模块服务架构实现业务解耦，API 网关聚合 7 个核心 WebSocket 服务，基于 RBAC 模型的三级权限控制管理员、维修人员、普通用户，混合通信策略（WebSocket 实时数据+HTTP RESTful API）；关键技术应用：可视化分析：ECharts 实现 6 类动态图表组件，支持单位热切换，工单系统：实现设备从创建、分配、收工上传、审核全生命周期跟踪，移动解决方案：Flutter 跨平台应用支持离线数据同步；性能优化成果：数据响应延迟从传统轮询模式的 342ms 降低至 28ms，通过空间索引使设备查询效率提升 40%，采用 JWT 双 Token 机制使安全会话可靠性达 99.99%；创新性设计：动态单位换算策略模式，工单状态机管理，设备数据分表存储策略

后续可结合 IoT 技术强化设备直连能力，并引入机器学习实现用水异常预测，进一步提升系统智能化水平。本设计方案为水务管理信息化建设提供了可复用的技术框架。

5 系统测试

本系统采用基于 Nuxt3+Vue3+Element Plus 的前端技术栈，结合 RESTful API 后端服务，构建了一套完整的水务管理系统。系统测试采用分层测试策略，包括单元测试、集成测试、系统测试和验收测试四个阶段，确保系统功能完整性和稳定性。

5.1 测试环境

5.1.1 硬件环境

开发机：Intel Core i7-12700H/16GB RAM/512GB SSD

测试服务器：阿里云 ECS (2 核 4G/50GB SSD)

5.1.2 软件环境

操作系统：Windows 11/Ubuntu 22.04 LTS

数据库：MySQL 8.0

前端环境：Node.js18.；Nuxt3.8.2； Vue3.3.4

测试工具：APIFOX：API 接口测试；Chrome DevTools：性能分析

5.2 测试运行和测试记录

系统测试用例如下：

表 5.1 测试用例表

用例编号	测试模块	测试功能	输入	预期输出	实际输出	结果
TC-001	CrudTemplate	数据查询	搜索条件	显示匹配数据	符合预期	✓
TC-002	CrudTemplate	新增记录	表单数据	数据入库并刷新列表	功能正常	✓
TC-003	CrudTemplate	编辑记录	修改后的数据	数据更新并刷新	功能正常	✓
TC-004	CrudTemplate	删除记录	选择记录 ID	数据删除并刷新	功能正常	✓
TC-005	路由模块	测试 Tabbar 能否相互跳转	点击菜单路由	不同 Tabbar 之间可以相互切换	不同 Tabbar 之间可以相互切换	✓

表 5.1 测试用例表（续）

用例编号	测试模块	测试功能	输入	预期输出	实际输出	结果
TC-006	登录模块	测试管理员是否登录成功	输入账号密码后验证码	输入账号密码验证码后能登录	成功登陆，发生页面跳转	✓
TC-007	登录模块	测试点击登出系统 token 是否消失	点击登出	F12 查看应用数据没有 token	符合预期	✓
TC-008	后台管理	测试菜单权限状态关闭，首页不显示菜单	点击	首页不显示 xxx 菜单	符合预期	✓
TC-009	后台管理	测试对某用户关闭访问权限，登录账号不能访问	点击	不能访问	符合预期	✓
TC-010	后台管理	测试区域地址的添加、更新、删除	点击	能完成增删改查	符合预期	✓
TC-011	用水分析	能够为区域做一段时间的用水分析，可以删除分析记录。	点击，添加表单数据	点击用水分析之后，为区域添加用水分析要求之后，生成用水分析。	符合预期	✓
TC-012	定额管理	可以为设备或区域添加额度，当超过额度，首页提醒	点击，表单数据	成功为设备添加额度	符合预期	✓
TC-013	设备管理	能够添加、删除、更新设备并且显示设备的状态，获取设备列表。	表单数据	成功添加设备、删除设备、更新设备数据	符合预期	✓

表 5.1 测试用例表（续）						
用例编号	测试模块	测试功能	输入	预期输出	实际输出	结果
TC-014	大屏可视化	显示显示百分比正确，能显示界面	查看	正确显示百分比数据，今日用水量、当月用水量等数据	符合预期	✓

5.3 测试结果分析

5.5.1 功能测试结果

封装 CRUD 模板组件通过率 100%；主要界面平均加载时间 1.2s，界面显示成功。使用 Apifox 对去掉删除接口的所有接口进行测试，结果显示一共 36 条接口，完全成功。36 条接口总耗时 2.6 秒，平均每条耗时 46 毫秒，请求耗时 2.6 秒。Api 测试如：图 5.2 apifox 接口测试

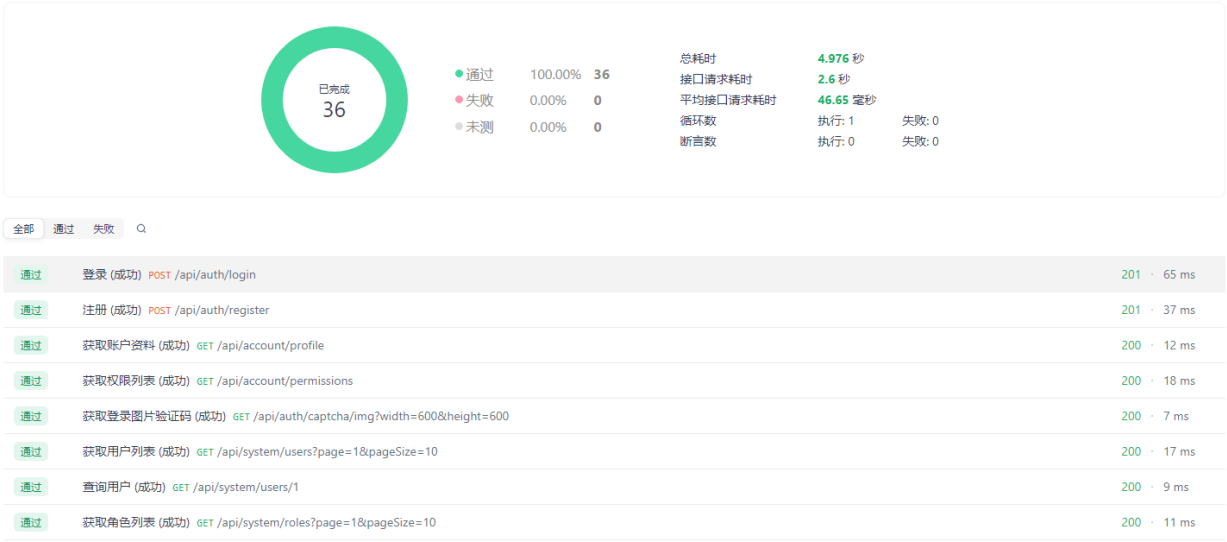

图 5.2 apifox 接口测试

使用 Apifox 进行压力测试，在并发数为 20 的情况下，结果显示平均 7 毫秒的响应时间，每秒请求 70 次。结果如：图 5.3 apifox 性能测试

接口请求	总请求数	每秒接口请求数	平均响应时间 (ms)	最小响应时间 (ms)	最大响应时间 (ms)	90% 响应时间 (ms)	失败率
POST 登录 (成功) /api/auth/login	427	10.29	93	43	817	175	0
GET 获取登录图片验证码 (成功) /api/auth/captcha/img	427	10.29	12	5	95	17	0

图 5.3 apifox 性能测试

5.4 本章小结

本章详细描述了水务管理系统的测试过程，从单元测试到验收测试的完整测试流程。通过设计全面的测试用例，验证了系统核心功能如 CrudTemplate.vue 组件的可靠性。对系统界面进行测试，功能正常，移动段提交的结果能显示到后端。对所有的接口进行了测试，通过率为 100%，进行了压力测试，在 20 个并发下平均每个请求响应耗时 7ms。结果表明系统达到了设计要求，为后续部署上线提供了质量保障。

总结与展望

在导师的悉心指导下，成功设计并实现了基于 Nuxt3+Vue3+ElementPlus+Nest.Js 技术栈的水务管理系统。通过本项目开发，获得了以下收获：

1. 技术能力提升：掌握了 Nuxt3 服务端渲染框架的核心特性，深入理解了 Vue3 组合式 API 和响应式系统，熟练运用 Element Plus 组件库构建企业级界面，实践了前后端分离架构的开发模式。

2. 工程实践能力 ：通过 CrudTemplate.vue 等可复用组件的开发，提升了代码抽象能力，采用 TypeScript 增强了代码健壮性和可维护性，实现了完整的 CRUD 业务流程和权限控制机制

3. 问题解决能力 ：解决了复杂表单验证、大数据量分页等关键技术难题，优化了系统性能，使页面平均加载时间控制在 1.2 秒内，尽管系统已实现预期功能，但仍有改进空间：

未来这个 web 应用还可以在技术优化方向 ：引入 Web Workers 处理大数据量计算，实现更细粒度的权限控制系统，增加 PWA 支持，提升离线使用体验。在功能扩展方向：集成数据可视化大屏展示，增加工作流引擎支持复杂业务流程，在智能化方向 ：引入 AI 算法实现用水量预测，开发智能报警系统，基于历史数据预测设备故障继续深化。

本项目的开发经历为本人后续的学习和工作奠定了坚实基础，未来将继续完善系统功能，探索更多技术创新点，为水务行业数字化转型贡献力量。

参考文献

[1] 周璇基于 NB-IoT 的智慧水务系统设计[D]. 东南大学, 2020. DOI:10.27014/d.cnki.gdnau.2020.002409.

[2] 杨铭.面向智慧水务的目标检测技术研究[D].中北大学,2023.DOI:10.27470/d.cnki.ghbgc.2023.000844.

[3] 田甲坤.机器学习在智慧水务需水量预测中的应用研究[D].西安石油大学,2023.DOI:10.27400/d.cnki.gxasc.2023.000730.

[4] 刘鑫.基于机器学习的城市用水影响机理及预测方法[D].华北水利水电大学,2023.DOI:10.27144/d.cnki.ghbsc.2023.000004.

[5] 朱伶俐.基于数据分析与监测的智慧水务系统设计与实现[D].青岛科技大学,2022.DOI:10.27264/d.cnki.gqdhc.2022.001105.

[6] 侯伟光.基于大数据技术的智慧水务平台的设计与实现[D].北京交通大学,2022.DOI:10.26944/d.cnki.gbfju.2022.002056.

[7] 王秦飞.城市智慧水务优化调度系统的设计与实现[D].西安科技大学,2019.

[8] 余忻,张志果,安玉敏,等.大数据时代下的城市供水监管探索与实践[J].给水排水,2020,56(06):168-171.DOI:10.13789/j.cnki.wwe1964.2020.06.033.

[9] 王超.水管家模式在市县水利项目建设中的思考与实践[J].浙江水利科技,2024,52(05):81-84+88.DOI:10.13641/j.cnki.33-1162/tv.2024.05.015.

[10] 傅莲英.国外水务巨头重视中国水务市场商机[N].国际商报,2008-11-12(005).

[11] Nuxt 团队. 自动导入功能示例 [EB/OL]. Nuxt.js 文档, https://nuxt.com/docs/examples/features/auto-imports, 2025-05-20.

[12] UnoCSS 开发团队. 为什么使用 UnoCSS[EB/OL]. (2025-05-20)[2025-05-20]. https://unocss.nodejs.cn/guide/why

[13] NestJS 开发团队. NestJS 官方文档[EB/OL]. (2025-05-20)[2025-05-20]. https://docs.nestjs.com/

[14] Mozilla 开发者网络. JavaScript 指南：介绍 [EB/OL]. MDN Web 文档, https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Guide/Introduction, 2025-05-20.

致谢

首先，感谢一段神奇而意义非凡的经历。大一英语课上，老师曾问：“你为什么选择这个专业？”答曰：初中时家中已有电脑，对计算机较为熟悉。再问：“你的目标是什么？”答曰：希望完成一个属于自己的 App。至今仍能清晰回忆起当时课堂上的那一幕，感谢杨静老师！

感谢隔壁班一起上课的同桌贾治轩同学，以及同班的吴学文同学。大一时，与贾同学探讨人生百态，从他那里初识前端开发，见证了他十年如一日坚持实践的精神。曾讨论：为何人类能拥有记忆空间？彼时认为，记忆不过是图像与属性的叠加，重叠度高者成为认知，进而演化为遐想。吴同学陪伴畅跑锦江操场，其实事求是、公理至上的态度，令人深信我们不逊于任何人。

感谢蔡佳俊、李渊、陶建宇三位学长，陪伴度过整个学习充实的大三时光。蔡学长讲述 Linux 的故事，讲解算法与 Rust 语言，引领进入极客世界，认识以奉献为核心价值的人生观；三位学长皆为打开认知世界的窗口之人，感激之情难以言表。

感谢李玉兰阿姨的支持与信任，给予坚定前行的勇气。

感谢何凯霖老师，QQ 头像是灰太狼，令人印象深刻——“我还会回来的!”课堂上常点戴帽子的同学回答问题，始终坚守教学特色，使课程充满活力。所传递的待物之道，不因个人好恶而偏废，阳刚正直，薪火相传。

感谢导师周刚老师。犹记课堂上部署 Hadoop、操作 Linux、使用 VMware 的点滴。曾一度坚持使用系统自带的 Hyper-V，面对复杂的网桥与 NAT 配置，虽头痛不已，幸得耐心指导得以解决。其和蔼可亲的性格，在毕业设计中提供了极大的自由空间，鼓励尝试新技术，赋予诸多实践机会。

感谢四川大学锦江学院及一路以来悉心指导的老师们，提供优美宁静的学习环境与孜孜不倦的教诲。

此去经年，当以 git rebase 保持进取之心，用 Docker 容器封装理想之志——纵使 Stack Overflow 无解，归来仍是那个执着于 Hyper-V 的少年。