

北京信息科技大学

毕业设计（论文）

题目： 集于浏览器组态软件---上层服务器数据汇总

学院： 自动化学院

专业： 自动化

学生姓名： 王畅 班级/学号 自控 1305/ 2013010653

指导老师/督导老师： 卢晶晶/周亚丽

起止时间： 2017 年 2 月 20 日至 2017 年 6 月 16 日

摘要

随着自动化技术的蓬勃发展，基于组态的自动化系统在工业领域得到了广泛的应用，如今组态监控系统已成为工业控制与自动化技术的重要组成部分。组态(configuration)早期用于在计算机控制系统中描述一些用于工业控制当中的软件工具的集合，随着 PLC 与相关工控技术的发展，组态软件的内涵慢慢演变成能够实现工业现场数据监测和过程控制功能的一些专用软件，通过使用组态软件，用户能够快速构建工业级的监控系统。

作为一种通用型的专业工具软件，组态软件按照所采用的服务架构可以分为 C/S 架构和 B/S 架构，C/S 的架构中，需要用户进行软件和相关驱动程序的安装，而 B/S 架构中，组态软件运行于浏览器之上，虽然 B/S 架构的组态软件相对于 C/S 架构软件的部分功能可能有缺失，但是随着互联网技术的发展，各种系统越来越云端化，手机作为移动的智能终端已经进入每一个人的生活中，因此基于浏览器的组态软件技术也越来越得到一些工控厂商的重视。

研华的 WebAccess 就是这样一款基于浏览器的组态软件，除了提供不逊于 C/S 模式下的组态软件的功能外，WebAccess 还提供了对手机端的支持，通过访问 IIS 服务器，用户可以实时的获取下位机的相关设备的运行参数，WebAccess 还提供了良好的用户界面进行实时的控制。本课题以完成一个风机监控系统作为工程背景，使用 WebAccess 完成了工程的搭建，通过 WebAccess 的 Web Service 接口，建立起一个运行在现代网络浏览器下的 Web 数据可视化的应用，从而完成了浏览器组态软件的上层架构，并且实现了数据更新与汇总功能，本课题的研究不仅完成了一个具体工程背景下 WebAccess 监控系统的搭建，还为组态软件的上层数据汇总实现提供了一种新的思路。

关键词：组态软件；监控系统；Web Service；数据可视化

Abstract

With the rapid development of automation technology, the automation system based on the configuration in the industrial field has been widely used, and now the configuration monitoring system has become an important part of industrial control and automation technology. The configuration is used to describe some of the software tools used in industrial control of the computer control system. With the development of PLC and related industrial control technology, the connotation of the configuration software has evolved into the industrial field data monitoring and process control functions of some special software, through the use of configuration software, users can quickly build industrial-grade monitoring system.

As a general-purpose professional tool software, configuration software in accordance with the service architecture can be divided into C / S architecture and B / S architecture, C / S architecture, users need to install software and related drivers. In the B / S architecture, the configuration software is running on the browser, although the B / S architecture configuration software relative to the C / S architecture software ,part of the function may be missing, but with the development of Internet technology, all kinds of systems is increasingly deployed on the cloud, the mobile phone as a mobile intelligent terminal has entered every person's life, so the browser-based configuration software technology is drawing more and more attention to some industrial manufacturers.

Advantech's WebAccess is such a browser-based configuration software, in addition to providing less than C / S mode configuration software functions, WebAccess also provides support for the mobile side, by accessing the IIS server, the user can real-time access to the relevant equipment operating parameters of the slave computer, WebAccess also provides a good user interface for real-time control. This project completes a windmill monitoring system as the engineering background, which uses WebAccess to complete the construction of the project, through WebAccess Web Service interface, establishes a web data visualization application which runs in the modern web browser, thus completing the browser configuration software. the implementation of this paper not only completed a specific engineering background WebAccess monitoring system, but also provides a new way of thinking for the implementation of upper configuration software in the upper data summary achievement.

Key words: The configuration software ; Monitoring system; Web service ;Data visualization

目录

摘要.....	I
Abstract.....	II
第一章 绪论.....	1
1.1 课题的背景与意义.....	1
1.1.1 课题的背景.....	1
1.1.2 课题的意义.....	1
1.2 国内外发展现状.....	2
1.3 课题研究的目标和内容.....	3
1.3.1 课题的意义.....	3
1.3.2 课题研究的内容.....	3
1.4 论文的组织结构.....	3
第二章 需求分析.....	4
2.1 工程背景需求.....	4
2.2 功能需求.....	4
2.3 非功能需求.....	5
2.4 本章小结.....	5
第三章 相关原理与技术.....	6
3.1 系统架构选型.....	6
3.2 前端架构.....	7
3.3 后端架构.....	9
3.4 下层架构.....	11
3.4.1 WebAccess 工程搭建与配置.....	11
3.4.2 下层数据的模拟.....	13
3.4.3 Web Service 服务介绍.....	13
3.5 本章小结.....	14
第四章 系统总体设计.....	15
4.1 系统分层设计.....	15
4.2 系统模块设计.....	16
4.3 本章小节.....	17
第五章 系统实现.....	18
5.1 服务器实现.....	18

5.2 数据实时更新实现.....	20
5.3 系统主要功能实现.....	21
5.4 软件部署与上云.....	23
5.5 本章小节.....	24
结束语	25
参考文献.....	26
致谢	27

第一章 绪论

1.1 课题的背景与意义

1.1.1 课题的背景

在 20 世纪 40 年代,大多数工业过程还处于工人手工操作的状态,工人们凭经验手动地控制工业生产过程,一些关键的参数只能依靠人工肉眼观察,到了 50 年代左右,一些较为先进的企业和工厂引进了一些工业仪表(多为气动仪表),虽然这种情况到了 60 年代左右,随着电子技术的发展,逐渐替换成了电动单元组合仪表甚至组装仪表,但不同仪表的控制相当分散,很多情况下都需要人工手动的去设定仪表的控制值。在这之后,随着计算机技术日新月异的发展,全厂或整个工艺流程进行计算机集中控制的技术越来越成熟,在这之后出现了集散型控制系统(Distributed Control System, DCS),就解决了这一问题[1],DCS 把计算机技术,控制技术和通信技术融为一体,组态的概念就是在此时逐渐的为技术人员所熟知。

组态软件是在工业自动化中被广泛采用的数据采集和监控的通用解决方案。组态软件(英文名称为 configuration),其本意就是设置或者配置,是指用户或者操作人员只需要对软件的功能进行组合即可达到应用的目的,组态是这个过程在国内普遍接受的称谓[2],在组态的过程中,系统提供了各种模块给用户进行选择,用户无需关注计算机底层的硬件和编写程序的情况下,通过组合不同模块的功能就可以构建出用户应用软件,同时,工程师也无需关注应用层面上的不同情况的复杂性,只需按照组态软件的模块标准进行开发,就可以满足应用层面的需求。

自上世纪 80 年代末被提出以来,组态软件到至今已经经历了三十多年的发展,早期的组态软件由于计算机技术的限制,只能在 DOS 环境下展示比较简陋的图形界面,随着视窗操作系统(Windows)的发展和普及,图形界面的技术也变得越来越受到关注[3],到如今,组态软件已经具有非常丰富的功能和人机界面,在国内,组态软件比较出名的有:组态王、力控、世纪星等。

在这些组态软件的使用过程中,通常需要安装平台相关的专业软件,在如今移动互联网日益发达的今天,无法满足跨平台需求的软件越来越显得难以开发和使用,因此市面上出现了不少能够进行 web 发布的组态软件,实现 web 发布有三种方式:第一种就是 ActiveX,但是这种技术必须要求用户指定计算机端口,一旦遇到路由器就不能被看到[4]。第二种采用 java 来手动开发 web 发布的功能,这种方式往往工作量较大并且数据刷新很慢,第三种方式,采用 .net framework 的 web service 技术,这种方式功能强大并且易于分离开组态软件的上下层结构,是目前组态软件 web 发布的最优解决方案。本课题将以研华公司的 WebAccess 为基础,利用 WebAccess 的 web service 功能,搭建起一个在浏览器环境下进行服务器数据汇总的组态软件。

1.1.2 课题的意义

本课题与一般组态软件区别在于本课题的组态软件全部搭建在浏览器平台上,在传统的客户端-服务器模式中,开发人员需要同时关注不同操作系统上不同的编程方式,但是随着互联网技术的长足进步,浏览器已经成为了不同系统上必备的一个软件,浏览器平台提供了一个无关于操作系统的统一平台,开发者只需要关注程序能够在浏览器上得到实现,而无需关注操作系统上的细节。

本课题的研究意义在于将流行的前端技术与组态软件实现在了一起，并提供服务器上云的解决方案，本课题解决了传统组态软件数据展示和采集耦合在一起的问题，随着互联网技术的发展和现代人对软件用户体验越来越高的要求，本课题实现的解决方案可以适用于大多数的浏览器环境下的组态软件。

随着国家对物联网技术的重视和各种云端服务器技术的成熟，浏览器上的组态软件相比于传统的组态软件有了更大的优势和前景，本课题实现的浏览器组态软件监控系统，正能够满足当下的这个需要，通过云端的服务器以及已经大规模普及的移动设备，管理者可以随时随地地查看工业现场的相关数据指标。

1.2 国内外发展现状

随着计算机信息技术的不断发展和用户对工业控制系统要求的不断提高，组态软件的发展也向着更高层次和更广范围发展.其发展趋势表现在以下三个方面：

1. 集成化、定制化。目前大多数桌面客户端的组态软件的代码规模已经超过了一百万行^[5]，已经超出了小型软件的范围。从组态软件的功能来看，数据的采集与通信、数据管理、数据统计分析等功能越来越强^[6]。组态软件作为一种通用的工业控制软件，需要具有很大的灵活性，但是实际上很多用户并不具备软件编程或者相关的专业技能进行维护，即用户需要能够凭借很少的定制工作量即可完成工程应用。因此需要既满足“通用性”又满足“定制性”。组态软件的开发必须进行组件化的方式，每个组件功能单一，并且只用于完成特定的功能，所以组件软件的集成化会越来越明显。
2. 接口标准化。厂商在开发组态软件时可以选择各种各样的利于自己的通信方式，但是在工业环境下，每一种通信方式必须要和硬件相匹配才能正确的工作，如果接口没有标准的统一，当硬件厂商采用一种新的通信方式时，程序设计人员就必须要重新进行学习^[7]，因此，接口的标准化是大势所趋。如已经被 OPC 国际基金组织提出的 OPC 工业标准，它定义了一个开放接口，在不同的客户机和服务器之间进行数据交换，在 OPC 标准下，服务器和客户之间通过 DCOM 接口进行通信，而不需要知道服务端或者客户端的内部实现细节。目前该标准已经得到大多数仪表以及控制厂商的支持^[8]。
3. 开放化、互联化。传统的组态软件在提供大量的现场数据的同时，无法对现场数据进行可靠的流程分析和数据统计，随着计算机信息管理系统相关技术的发展，在组态软件中集成信息管理系统已经成为一种趋势。
4. 应用扩大化。组态软件在技术上实现了数据的实时通信，在工业环境下只要对数据实时通信的有所要求，就间接地存在对组态软件的需求，因此组态软件的应用将会越来越多，越来越扩大化。

从以上可以看出，本课题研究的目标正是发展趋势的第三点的内容，目前国内外的组态软件发展大多向提供功能更多，配置更简单的方向发展，很大程度上没有解决组态软件只能在局域网或者工业现场使用的现状，在开放化和互联化的趋势下，这种情况将会逐渐的被改善。

1.3 课题研究的目标和内容

1.3.1 课题的意义

本课题是集于浏览器的组态软件—上层服务器数据汇总，本课题将以一个实际工程为背景，研究如何完成一个双层架构的组态软件系统，并且实现在浏览器环境下进行组态软件数据汇总的方法。本课题的意义在于在工程层面上实现了双层架构和浏览器上的组态软件，在应用层面上本系统实现的数据可视化系统可以方便的进行扩展和二次开发，为传统组态软件的开发企业提供了一个可以实现浏览器组态软件的实现思路。

1.3.2 课题研究的内容

本课题主要研究组态软件在浏览器环境下的实现和 WebAccess 上层服务器的数据可视化。在浏览器环境下，信息的传递变成了一次次的 Http 的请求，通过 HTML 语言和 Web 实时通信技术，可以做到数据的实时获取和可视化展示。根据实际工程的要求，将需求点抽离出来，搭建一个 Node.js 作为网页服务端、浏览器端实时更新的可视化系统。HTML 可以被所有的主流浏览器所解释执行，意味着本课题研究的系统具有很高的可移植性和扩展性。

本课题以搭建一个风机监控系统的上层服务为工程目标，编写前端的 Html 代码和 Css 代码完成数据可视化界面的设计，并通过固定时间间隔轮询下层服务器所采集的数据,通过目前主流的前端技术完成数据的可视化展示。

1.4 论文的组织结构

本论文的主体内容主要由以下几个方面构成。

首先，本论文会对课题的背景与意义进行简要的说明，并提出本课题所要解决的主要问题目标，根据相关的书籍资料与网上搜索的信息对现有的几个组态软件产品进行调研，提出本课题所实现的系统所解决的具体问题，得出将会采用的主要技术，之后概述了本项目的国内外相关的发展趋势以及对未来的展望。

其次，本论文将从需求出发，对本课题进行详尽的需求分析，并总结需求分析的结果，之后，将会详细的描述本课题所采用的技术与原理，在此之上提出系统的设计方法，按模块介绍系统的具体实现，之后简对该系统的部署与上云进行介绍。

最后，本课题对该系统进行了总结和展望。

第二章 需求分析

2.1 工程背景需求

本课题以实现一个风电监控系统的上层部分为具体的需求背景，在该工程中信息的流转图如图 2.1 所示。



图 2.1 信息流转图

在此工程背景下，数据采集需要对原始数据值进行模拟，下层监控需要对工程中的数据点进行具体的配置，上层展示需要完成浏览器界面的编写和数据展示。

2.2 功能需求

功能需求包含此系统所至少需要完成的功能，要实现一个风电监控系统的上层部分，我们最关注的是数据在展示汇总层的相对应的功能，在上一节的工程背景需求下，风机现场风机存在着多种状态的数据，首先这些数据能够被正确的获取到，其次对这些数据进行汇总分析，生成可视化的图标给管理者进行分析，最后就是一些报警与登陆登出信息的记录，按照以上逻辑进行梳理，最后归纳的功能需求列表如表 2.1 所示。

表 2.1 功能需求列表

功能序号	功能描述	优先级
1	双层架构（服务器端）	3
2	运行日志	3
3	数据报警	3
4	远程监控	2
5	数据实时采集	3
6	公网发布	2
7	动态 IP 主动上报	2

优先级是指该需求条目在设计时被考虑实现的权重，优先级越高表示需求越重要，更应该首要保证。

功能序号 1 中，双层架构指本课题完成的系统有上层和下层两个不同的架构，上层和下层各启动一个 WebAccess 程序，上层负责可能多个下层服务器的数据汇总，下层则处理数据的采集。

功能序号 2 表示对 WebAccess 运行的日志进行记录和展示，运行日志包括有 WebAccess 的登入登出信息，数据以及相关的运行环境信息等。

功能序号 3 数据报警表示对 WebAccess 中点的值在超过量程时，对 WebAccess 发出的报警信息进行记录并展示。

功能序号 4 远程监控是指通过浏览器设备可以直接在上层控制下层的监控数据。

功能序号 5 表示数据需要进行实时的采集，并且在界面中合理的展示出来。

功能序号 6 表示整个系统应该能够进行公网发布。

功能序号 7 的含义是上层和下层之间通过 IP 上报的方式进行数据的同步。

2.3 非功能需求

非功能需求主要指不影响系统运行但是有助于提高系统的表现的需求，包括有系统本身性能、安全性、易用性、使用体验等等。首先，本系统作为上层数据汇总部分，需要在浏览器中对数据进行展示，网页的加载时间是一个性能方面所需要考虑的因素。其次，上层部分本身不能够对底层硬件进行直接的控制，但是如果要求登陆校验，系统的数据信息的安全性将会大大提高，最后，如果能够完整的说明文档，可以对二次开发或者后续维护有较好的支持。综合以上几点，该系统的非功能性需求表如表 2.2 所示。

表 2.2 非功能性需求

需求分类	需求描述
性能	页面响应时间在 2.0 秒内，渲染结束时间在 7.0 秒内
安全	要求用户登录后才可访问
可维护性	提供文档与说明书

性能需求，上层系统的浏览器响应时间应该尽可能的快速，这里的 2s 和 7s 参考了部分网页系统的时间指标。

安全需求，由于 web service 提供的接口是可以进行公开访问的，所以安全性也是一个重要的需求，但由于这里只是对工程的一种模拟，并不会牵涉到实际的设备，因此安全性需求被放在了非功能性需求的列表中。

可维护性，由于本课题实现的系统完成了一个完整的 web 系统，并在实际实现的过程中编写了大量的代码，因此，需要保证代码具有良好的可维护性。

2.4 本章小结

本章主要介绍和分析了本课题研究的系统所存在的具体需求，包括有工程背景需求，功能需求和非功能需求，在进行需求分析时，由于本课题的研究对象旨在提供一种解决方案，并不是适用于某一个具体的工业环境下的系统，因此许多可能较为特定的需求没有考虑到，在和企业导师沟通过程中也与导师沟通过这个问题，我和导师认为，在本章节中提出的需求都满足并且提供良好的说明文档的情况下，本系统能够进行很好的扩展与应用，这也正是双层架构上层和下层分离所独有的优势所在。

第三章 相关原理与技术

在开发本系统之前，对整个流程的重要原理和技术进行概括时十分有必要的，一方面，在整个流程上的把控可以让自己对系统的需求有更深入的了解，同时也能够的技术选型上进行原始的设计，避免选择了整体架构却难以实现的情况。本课题中，采用 B / S 的架构来实现双层架构，前端采用 Vue.js 的相关方案，后端用 Node.js 开发的中间服务器，WebAccess 的 web service 相当于数据库服务。

3.1 系统架构选型

在开发本系统之前，基于 WebAccess 软件实现上层服务器数据汇总有两种架构选型：

1. 采用 WebAccess 本身的 HMI 绘制功能。WebAccess 中提供了一个绘图的子程序中，在这个程序中可以按照窗口拖拽的方式进行界面的编写，这种方式被研华公司的目前大多数工程现场所采用，作为下层部分，使用 WebAccess 绘图功能完成的工程能够很好的进行双向的展示与控制，但是却也面领着图形界面过于陈旧，脚本编写（宏）难度较大等问题，由于本系统旨在上层的数据采集与汇总，这种方式显得过于笨重，并且不是完整意义上的 B/S 架构。
2. 采用 WebAccess 的 web service 功能。web service 是一项资源获取服务，能够以 http 请求的方式获取相应格式的回应数据。在这种架构中，WebAccess 被抽象成了服务端，任何人采用指定的方式（Http Method）都可以进行数据的获取，获取到数据之后，采用相关的前端技术，就可以获得更好的用户体验与系统维护状态。

采用方案 2 的关键在与对 B / S 架构的设计与理解。B/S 架构是随着互联网技术的发展越来越流行的一种软件开发模型，B/S 模式下软件程序运行在浏览器环境下，通过 Http 请求与服务器进行通信，B / S 架构的示意图如图 3.1 所示。

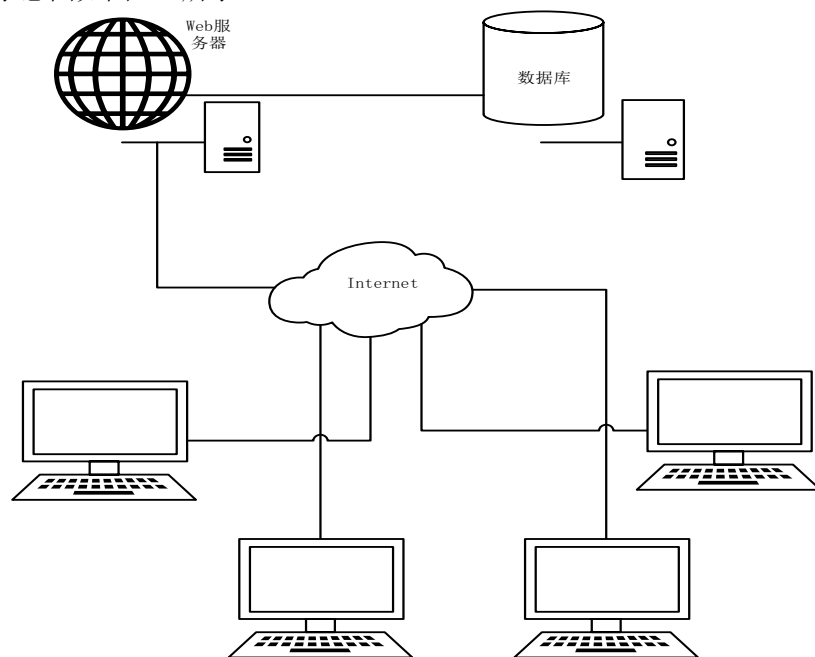


图 3.1 B/S 架构示意图

在 B/S 架构下，web 服务器从数据库服务器获取数据，通过 web 应用程序发布网页，用户在访问 web 服务器时获得即时的信息服务，服务端和浏览器之间通过 http 协议来进行通行，http 协议主要定义了 Http 报文的格式以及发送的方式，图 3.2 展示了 Http 协议的报文组成。

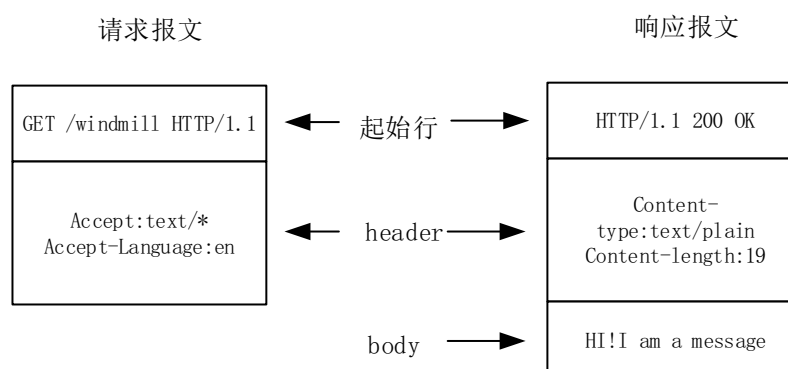


图 3.2 Http 协议报文组成

Http 方法是指发起 http 请求的方式，根据发起请求的关键字服务端程序可以进行丰富的编程扩展，Http 请求方式如表 3.1 所示。

表 3.1 Http 请求方式

方法	意义
OPTION	请求一些选项的信息
GET	请求读取由 URL 所标志的信息
HEAD	请求读取由 URL 所标志的信息的首部
POST	给服务器添加信息
PUT	在制定的 URL 下存储一个文档
DELETE	删除指定 URL 下存储的一个文档
TRACE	用来进行环回测试的请求报文
CONNECT	用于代理服务器

按照 Http 协议的规范，服务器返回的 header 中包含有当前信息的类型 Content-type，浏览器根据此类型对文档进行解析和渲染。比如 content-type 是 html 类型时，浏览器按照 html 标准对文档中的文本、图片等进行解析，渲染成一个存在于浏览器内存中的树状对象模型（document object mode，DOM），当 html 中引用了外部样式表或者脚本文件时，浏览器发送新的 http 请求把这些资源请求到并加载到页面中，样式表（CSS）负责处理网页的布局与样式颜色字体等，而脚本文件负责处理浏览器的使用者与当前网页的交互，如移动鼠标，点击，键盘输入等等事件。

3.2 前端架构

前端。通常指网站用户看到的界面的代码实现，前端的基本语言包括 HTML，CSS，JavaScript。

HTML（Hypertext Markup Language，超级文本标记语言），超文本，就是指这种语言可以包含图片，链接甚至音乐视频等内容，HTML 是 web 编程的基础，它本身也是一种 W3C 和浏览器厂商协调下的一个标准和规范，HTML 语言具有简易性，可扩展性，平台无关性，通用性等特点，从具体的程序角度看，它只是一个用成对的尖括号所组成的文档，如<head></head>表示网页的头部区域，<body></body>表示了网页的主题部分。

CSS (Cascading Style Sheets, 层叠样式表), 是用来定义网站样式结构一种语言, CSS 中可以定义 HTML 标签的具体外观, 比如段落<p></p>可以定义段落中的字体大小, 字体颜色, 字体的位置等等, CSS 能够对网页的 html 结构和内容进行丰富的样式自定义, 基本上支持目前所有的 GUI 表现的效果。

JavaScript 是一种被设计在浏览器环境下运行的脚本语言, 它被设计成弱类型, 面向对象和基于原型的特点, JavaScript 原来只是在浏览器环境下的一种脚本, 只能被浏览器的 JavaScript 解释引擎解释执行, 后来独立成为一门编程语言, 随着 node.js 等框架的兴起, JavaScript 也被应用在了服务端, 在浏览器环境下的 JavaScript 主要是和 DOM(Document Object Mode) 和 BOM(Browser Object Mode)进行打交道, 比如从 DOM 中获取到某个节点, 对这个节点的内容进行删改, 或者是和 BOM 打交道。

JavaScript 目前是 web 编程应用中使用最普遍的编程语言, 目前所有的 web 平台上都能够支持 JavaScript 语言, 而且目前所有稍大型的网页组件库都是用 JavaScript 作为主要语言来进行实现, 包括有各种各样的上传组件, 图表分析组件等等, 随着服务器端的 node.js 的兴起, JavaScript 逐渐应用到了服务器端的开发中, 甚至已经出现了使用 JavaScript 来做增删改查的数据库, 所以本课题的代码实现中使用了大量的 JavaScript 编程, 掌握必要的 JavaScript 编程会对本课题的代码实现有更多的了解。

在进行前端架构的选择时, 市场上已经有很多比较高效的解决方案, 很多前端类库都有各自的优势与不足, 比较有名的前端框架有 AngularJS, React.js, Vue.js 等, 在经过仔细对比书籍和技术社区对三种框架的讨论与评价之后, 本系统最终采用了 Vue.js 的作为前端的基本框架, 理由有以下几点:

1. Angular.js 虽然有逻辑清晰的模块化机制, 但是对于本系统而言, 它的技术成本显得过于笨重, 并且框架提供的许多功能学习成本比较高, 较短时间内很难开发出一个性能良好的网站。
2. React.js 是国外引进的一个前端框架, 它本身工程设计良好, 但是相比于传统的前端开发, 引进了许多较难掌握的特性和使用方法, 并且 React.js 一般应用于较大型的工程项目中。
3. Vue.js 框架小而灵活, 上手也十分容易, 社区文档都有中文版, 学习成本较低。

Vue.js 是前端的一个提高开发效率的一个前端框架, 在 Vue 框架中, 它定义了一个概念清晰, 功能明确, 并且逻辑流程统一的开发模式, 传统的开发方式中, HTML, CSS, JavaScript 这三种语言相互引用资源, 往往浪费了大量的时间去排除引用资源路径出错的问题, 另一方面, 由于 JavaScript 语言单线程和异步调用的特性, 很多的回调函数让页面的 JavaScript 脚本看起来非常难以维护, 而在 Vue 的开发框架中, 开发人员在单文件组件中可以方便的书写与当前页面相关的 HTML, CSS, JavaScript 代码, 同时由于 Vue 框架中定义了生命周期函数, 因此开发者只需要在对应生命周期函数钩子上对页面逻辑的代码进行编写调用就可以了, 由于在代码的编写中需要配合 vue.js 的框架特性来写单文件组件, 因此这里对 vue.js 框架的生命周期做简要的介绍, 图 3.3 展示了 vue.js 的完整生命周期。

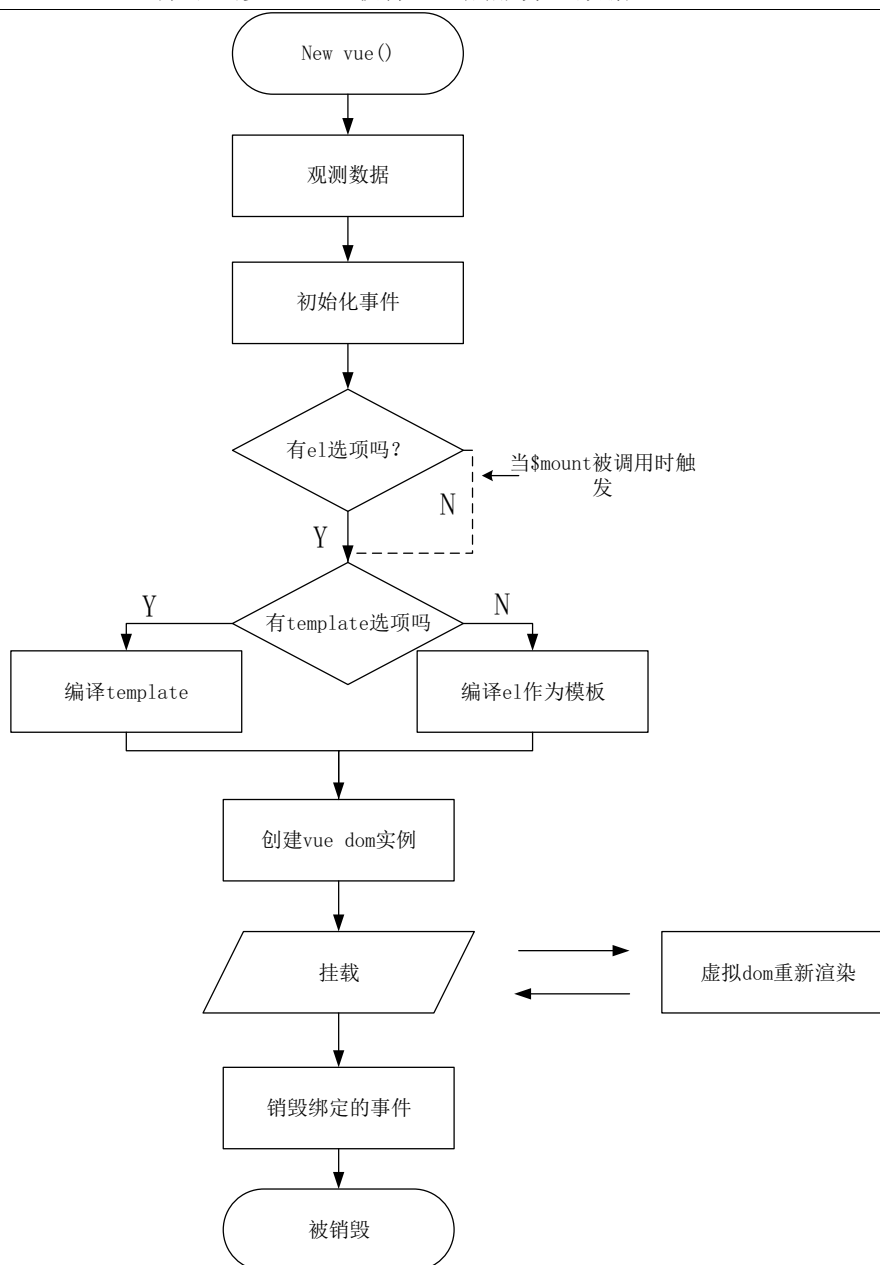


图 3.3 vue.js 的生命周期

从图 3.3 中可以看出，vue.js 的实现是基于 JavaScript 语言的原型机制，在实际的代码编写中，我们只需要关注一些事件被触发的阶段，然后按照 vue.js 的模块组织起来就可以了。

在构建工具的选择上，采用 webpack 作为打包工具，webpack 是一个前端的打包工具，前端所使用的资源比如图片，样式表文件(css)，脚本文件，字体等，这些资源在 webpack 中被抽象成一个个独立的模块，webpack 使用相应的 loader 来进行加载，最终打包成一个 JavaScript 文件，这样减少了请求，开发者也无需关注这些资源之间相互的依赖关系。

3.3 后端架构

在进行后端架构的选择时，参考了几个比较流行方案，综合比较下来，采用 node.js 是最合适的方案，Node.js 是一个服务端 JavaScript 语言的运行平台。它是对谷歌浏览器的 JavaScript 引擎进行了封装。V8 引擎执行 JavaScript 的速度非常快，性能非常好。Node.js 对一些特殊的用法进行了优化，提供了在操作系统上访问的替代 API，使得 V8 在非浏览器环境下也能够运行并且运行的性能很高。。

Node.js 的内部实现采用模块机制，它本身提供的 Api 也都是模块化的，比如 http 模块封装实现了 HTTP 协议，net 模块封装了 TCP 协议，在 node.js 中引用一个模块需要遵守 CommonJs 规范—非常简单，引入一个模块使用 require 函数就能够方便的引入，node.js 的架构图如图 3.4 所示：

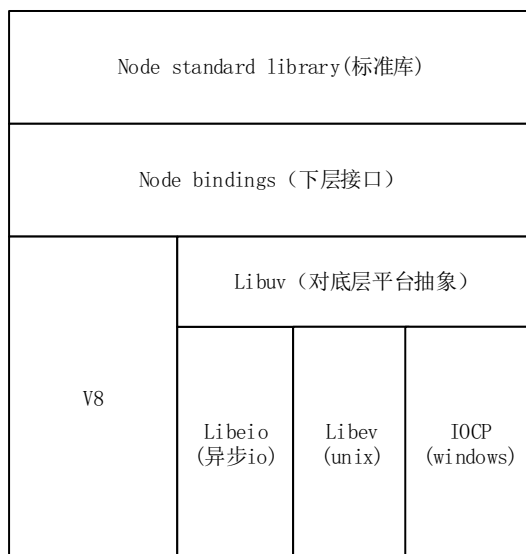


图 3.4 Node.js 架构图

在实际的工程项目中，用到的更多的是 node 的 npm 包管理器，npm 是 node 实现的一个核心管理器，在 node 中，需要下载一个模块，使用 npm 命令相关命令即可。

值得一提的是，本课题除了实现一个 Node.js 服务器之外，还完成了数据获取的实时更新。

在浏览器上的实时通讯长期以来都是困扰开发人员的一个难题，浏览器在考虑用户体验时，在开始设计时就对实时推送做了许多的限制，而且市场上也没有一个统一的解决方案，但是在长期的开发实践中，开发人员总结出了四种实现方法：固定时间轮询、长轮询技术、WebSocket、SSE (Server-sent Events)。

固定时间轮询，是指按照一定的时间发起请求，将每次得到的数据进行替换，适用于架构较小的中小型系统解决方案，它的技术成本低，性能中等，并且便于维护，是小型应用场景下的最优选择，缺点是系统用户数量较多时，性能成本将会上升很多，所以对于大流量的软件系统，应该考虑其他方式。

长轮询是一种非正式的解决方案，使用长轮询技术可以在浏览器标准或者服务器不进行大改造的基础上完成数据的近乎实时的推送，目前已经在一些场景下被广泛使用，但是长轮询技术复杂度比较高，并且很难做到很好的优化，

WebSocket 是一种浏览器标准支持的方式，WebSocket 是一种双工通信—浏览器可以通知服务器，服务器也可以通知浏览器，在本系统的情景下，只需要实现服务端通知客户端，因此功能有些多余，

SSE (Server-sent Events, 服务器推送事件)，服务器推送事件是 HTML 5 规范中的一个组成部分，可以用来从服务端实时推送数据到浏览器端。相比较于长轮询和 WebSocket 技术而言，服务器推送事件的代码实现更简单，对服务器端的额外开发也比较小，但是 SSE 传输数据时是按照文本的格式进行传输，当数据量比较大时，对数据的解析将会十分困难，因此很适用于传输较小的数据量，数据量偏大的情景将不是很适合。

综合以上几点，本系统采用了最简单的固定时间轮询的方案来完成，固定时间轮询开发成本较低，并且能够足够满足本系统将要实现的数据更新的要求，使用固定轮询的方案，只需要对获取数

据的异步请求设置一个定时器，每隔一定时间重新发起请求就可以了。

3.4 下层架构

3.4.1 WebAccess 工程搭建与配置

本课题选用的组态软件是研华的 WebAccess 组态软件它整合了从工程搭建、人机界面制作、数据库管理和软件配置等组态软件常用的功能^[9]，在本课题的研究范围内，web access 以相当于数据库服务的角色出现，因此有必要对 WebAccess 的工程搭建流程做简要的概述，供读者参考，WebAccess 搭建流程图如图 3.5 所示：

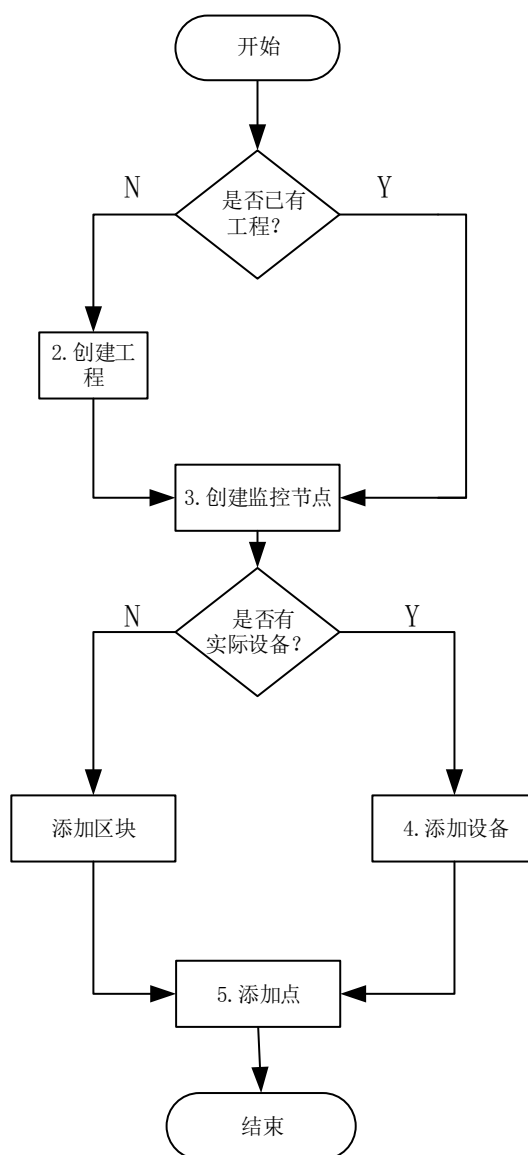


图 3.5 WebAccess 工程搭建流程图

1. 进行软件的安装。在下载完软件并进行一系列安装步骤之后，WebAccess 会自动启动 IIS 服务器，监听本机(IP: 127.0.0.1)的请求，打开浏览器并输入 <http://127.0.0.1>。可以看到工程首页，点击工程管理并输入用户名和密码校验后，会进入到管理员主页，我们可以新建一个工程或者时管理已有的功能。

2. 工程的创建。点击工程管理并输入用户名和密码校验后，会进入到管理员主页，我们可以新建一个工程或者管理已有的功能，新建一个工程时，按照输入框的提示填入相应的内容即可。
3. 监控节点的创建。新建完工程后，进入工程管理界面，在界面的上方选择“添加监控节点”选项，然后点击“建立新的监控节点”，之后在输入框中填写监控节点的名称，监控节点的 IP 地址、主要 TCP 端口号、次要 TCP 端口号都使用默认值就可以了（IP 地址不填时默认为本机的 IP），节点超时指的是监控节点与客户端、工程节点和 ASP 服务器正常通讯的时间，远程密码访问指的是系统显示默认的远程访问密码，这个密码在软件安装时设定。
4. 添加通讯端口。通讯端口是监控节点与硬件设备的一种物理连接，在监控节点的顶部选项栏中，选择“添加通讯端口”进入添加通讯端口的界面，此时应该考虑具体的硬件设备是采用何种方式进行通信的，对应选择不同的接口名称，WebAccess 支持的通讯接口名称由 API, Bacnet, LNS, OPC, RSLINX, Serial, TCP/IP
5. 添加设备。配置好通讯端口后，还需要设置对应端口的硬件设备，每个设备都归属于一个特定的通讯端口，因此添加完通讯端口后，只有符合该通讯端口的设备才会被添加。在添加的其他选项中，单元号是指，对某些驱动入 WebAccess，需要指定总线的地址栈号，设备类型同通讯端口的接口名称，是指该设备的硬件支持的通讯方式。
6. 添加点。在 WebAccess 中工程采集的数据都以点的值的形式被采集进来，添加点在这个工程中是非常重要的步骤，她是软件与硬件之间最底层通信的桥梁，在一般的 WebAccess 工程中，通常需要根据现场设备的情况选择不同的添加点方案。添加点的方法是，在设备属性界面上选择添加点，在点的类型上选择对应的数据类型（模拟量，数字量，相位量），点名称按照惯例来说通常按照点的类型与编号进行命名，如数字量用字母 O 表示，模拟量用字母 A 表示，所以命名一般为 A1, A2, A3...。地址填框第一个设置 WebAccess 会根据设备的首地址进行自动填入，但是后面的点需要依次递增。其他默认值不用变，在底下的记录到 ODBC 的频率改为 10 秒，这样每 10s 工程就可以接收到一次数据，5 分钟内工程就可以采集到大量的数据来分析。这样就添加完了第一个点。

值得一提的是，在 WebAccess 中，有一种特殊类型的设备，它是 WebAccess 之间通信的一种方式，它被称为虚拟 SCADA 节点，虚拟 SCADA 节点可以通过驱动程序代替 Modbus 驱动，实现与另一个 WebAccess 的通信，它经常被用在双层架构中上层用来将下层的工程添加成虚拟的设备节点^[10]。

在 WebAccess 中还有一种比较特殊的点类型，它被称作为“区块（block）”，区块是采集的参数集合，并且每个参数都可能有不同的类型，区块的出现弥补了建立点时大量的重复操作，而使用偏移地址进行点的自动生成，通常还应用于详情显示和区块报警等^[11]。

IP 主动上报是指，在 WebAccess 双层架构中，下层服务器的 ip 地址在有可能发生变化的情况下，主动将自己的地址上报给上层服务器，上层服务器根据此地址拉取下层的代码，并将下层的工程虚拟化成为一个区块添加在上层工程的监控节点之下，之后，上层服务器通过轮询机制来定期请求下层的数据更新，同步更新到上层对应区块的对应点中。IP 主动上报是 WebAccess 软件自带的一个软件功能，可以在工程目录下的监控节点选项中，进行添加。

3.4.2 下层数据的模拟

在工业现场中，原始数据来源于底层控制网络和现场总线或者设备的直接上报，但是往往会出现无法及时联通硬件设备的情形出现，因此在 WebAccess 中经常需要对假数据进行模拟，模拟的方式有两种：第一种采用 Modsim 等串口模拟软件^[12]，第二种，采用 WebAccess ViewDAQ 面板提供的宏指令进行模拟。在本课题中，上层的设计不关心数据采集层的方式，因此采用 ViewDAQ 面板的宏指令来进行数据的模拟。

在进行简单的绘图操作并且进行下载之后，启动监控程序，打开主控界面，然后在监控面板中的上方工具栏中，点击点管理图标，在点管理的右侧有一个调用宏指令的选项卡，点击展开后找到宏指令“%DKRLMODE%”，双击并且登陆之后选择 SIMU，此时所有的点的值都会被模拟成该点被设置的量程的一半^[13]。

3.4.3 Web Service 服务介绍

Web Service 技术，实际上就是通过 http 请求能够对资源进行访问和修改，而不需要借助特定的软件或者编程语言来进行实现，在依据 web service 规范开发的应用中，无论服务端和客户端采用的技术、语言或者平台是什么，都能够进行资源的访问和集成。

在使用 web service 技术时，资源通过 http 请求的方式被服务端发送给客户端，根据前两个小结的相关内容，我们在 node 平台上实现访问 web service 资源的流程如图 3.6 所示：

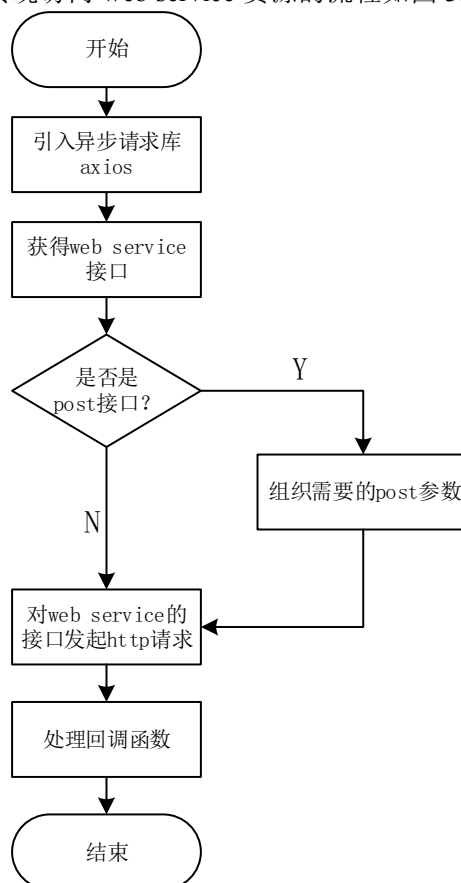


图 3.6 Web Service 访问程序流程图

首先，我们需要创建一个 http 请求，这里我们使用了 node 平台下一个非常知名的异步请求库 axios，使用 npm install 安装完异步请求库 axios 之后，按照 axios 的官方文档，可以创建一个 http 请求。它绑定了一个回调函数，回调函数的参数就是 web service 所返回的数据。

在 WebAccess 中，web service 提供了两种不同的数据格式，XML 和 JSON，XML 格式需要相应的解析器来进行解析，而 json 相比较而言有着数据量小，易于处理，解析方便的特点，因此，在本系统中，采用 json 的接口来进行数据的获取。

在本系统中，web service 是 WebAccess 组态软件自带的一项服务，附录 C 中列出 WebAccess 的 Web Service 常用接口的原型以及含义。

3.5 本章小结

本章主要对实现上层服务器数据采集所需要的相关技术原理进行了叙述，包括系统上层的前端架构采用 Vue.js 作为主要框架、后端使用 Node.js 搭建 web 服务器，下层 WebAccess 工程搭建的相关过程以及 Web Service 的相关介绍等。

在技术方案的选择过程中，一方面考虑了技术的成熟型和是否易于上手，同时也对技术在本课题中被应用的主要方式进行了重点介绍，包括有前端 vue.js 的生命周期，后端 node.js 的核心实现，以及数据实时获取方案的介绍，由于本课题的主要研究重点在于上层服务器的数据采集汇总，因此，上层的系统知识原理相对而言就为丰富，组态软件 WebAccess 的安装和下层的配置只做了简要的介绍。

第四章 系统总体设计

在介绍完相关原理与技术后，接下来进行系统的总体设计概括。其中，本课题所实现的系统包含系统的分层设计与系统功能模块的设计。

4.1 系统分层设计

系统分层设计，由于本系统所要实现的功能需求之一就是双层架构，这里的层就是指上层与下层，在上一章节中，已经就上层和下层的主要技术与原理进行了介绍，因此，本系统所采用的系统分层设计架构如图 4.1 所示。

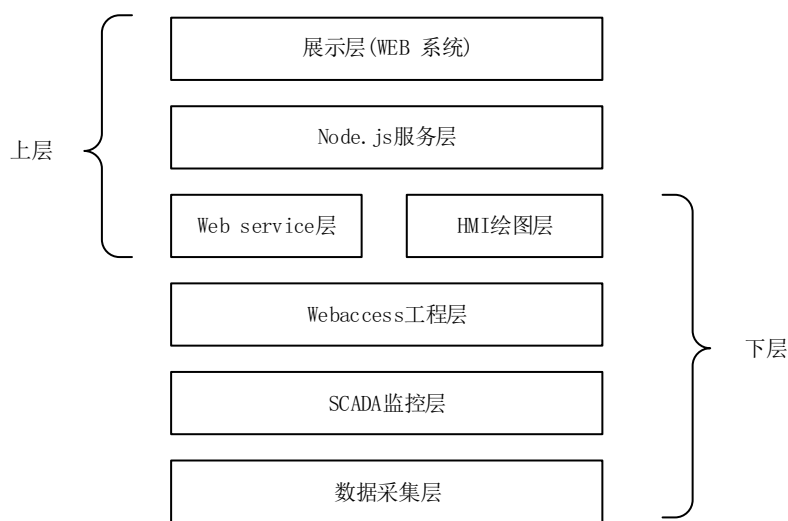


图 4.1 系统分层设计架构图

最上层为展示层，它的主要内容就是 web 界面，这一层是用户所接触与交互的一层，在这一层需要实现将从接口获得的数据进行处理，并使用合理的方式展示出来。

Node.js 服务层是 web 服务器所工作的地方，在这里进行数据的请求与处理，并且以 HTTP 相应的方式返回给前端，以此实现界面的展示，同时还会接受从显示层发过来的请求，并给予响应。

Web service 层，主要提供一个 RESTful 的接口机制，把组态软件的实时数据提供到这一个层面上，这样可以让第三方应用进行数据请求和处理，这一层同时实现了认证机制。

WebAccess 工程层，这一层是工程所建立的地方，在双层架构中，这一层实际上包含了上下层两个 WebAccess 软件运行的程序，但是他们的功能是统一的，都是用来对工程进行管理，因此在这里把它们放在了一个层级。

WebAccess SCADA 监控层，这一层实现数据的监控与收集，是 WebAccess 数据库原始数据的来源点。

WebAccess 数据采集层，这一层提供现场的数据采集，数字量或者模拟量以标准的形式被采集到监控节点中。

在进行分层设计时可以清晰地看到不同层次之间信息传递的方向，同时对整个系统的架构理解也更加清晰、明了。

4.2 系统模块设计

对系统进行按模块的设计有助于提高系统的可维护性和代码的可读性，同时也可以实现比较清晰的把握系统实现的思路，本系统实现了功能需求中的六大模块功能，这六大模块之间没有相互耦合，并且各自功能独特。从数据层面上有 Web Service 模块、Node.js 服务端模块，数据实时获取模块。从内容层面上，包含有前端各个不同功能内容的模块，其具体的模块结构图如图 4.2 所示。

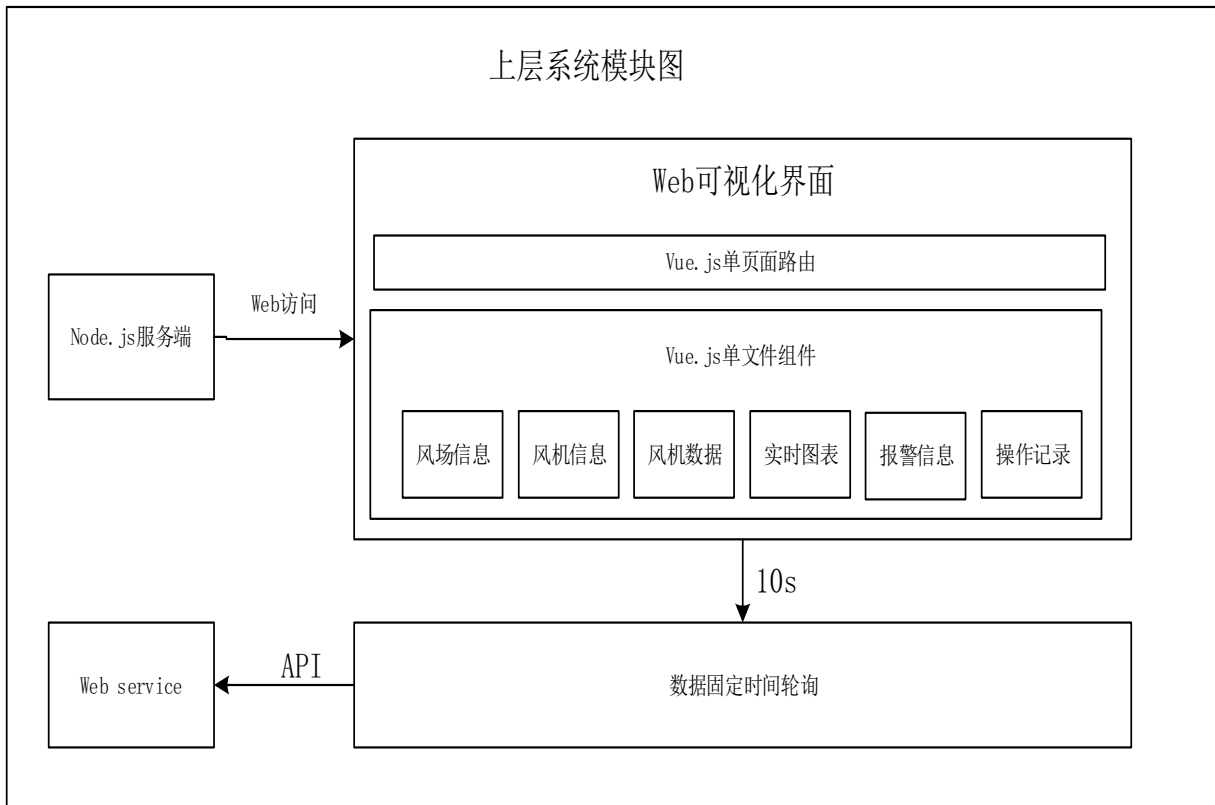


图 4.2 模块结构图

1. 风场信息模块。主要内容为展示风场的运行状态信息，包括下层服务器版本，服务器时间，当前用户，服务器最大连接数等。
2. 风机信息模块。主要内容为系统中所有风机状态的简要展示，如运行功率，运行状态，风速等
3. 风机数据模块。主要内容为系统中所有风机的各项数据指标的表格展示，包含有风机的各项参数信息。
4. 实时图表模块。此模块的主要内容对风机数据关键数据项进行汇总和图表化的展示
5. 报警信息模块。此模块记录风机状态报警信息。
6. 操作记录模块。此模块记录系统登入登出信息。

从图中可以看出本系统设计的模块之间耦合性较低，并且大多数都是数据的展示，没有非常复杂的交互功能，非常便于代码的书写以及测试。由于在前端已经采用 Vue.js 作为基本框架，因此在设计这些模块时自然地采用了 VUE 的单文件组件来表示这些不同的模块，按照上一小节的分层设计的思路，在每个模块中实现对固定时间的轮询来获取刷新数据，这些数据的具体内容按照 web service 提供的接口列成表 4.1 所示。

表 4.1 不同模块中内容所对应的接口名称

模块	接口名称
风场信息模块	Server time, client max num, userinfo, version
风机信息模块	Tag name, tag value , node detail,
风机数据模块	Tag name , tag value , node detail
实时图表模块	Tag name , tag value , node detail
报警信息模块	Alarm count, alarm detail
操作记录模块	Log count, log detail

4.3 本章小节

本章主要介绍了系统的总体设计，从分层和模块的角度对系统的架构进行了概括，并对每一层或者每个模块的功能进行了叙述。

从分层的角度，本系统可以分成六层，分别为展示层，Node.js 服务层，web service 层，WebAccess 工程层，监控层，数据采集层，并且对每一层的功能进行了详细的解读。

从模块的角度，对不同模块的功能进行了介绍，同时概括了不同模块的主要内容，之后给出了模块的不同内容的信息接口。

第五章 系统实现

在进行完总体设计之后，接下来介绍具体的系统实现，在第三章相关原理与技术中，已经介绍了下层 WebAccess 软件的安装配置，由于本课题的研究重点在于上层服务器的数据汇总，因此这里的系统设计也指上层前端和后端，系统实现的重点包含有三个部分：服务器实现，数据实时获取实现，系统主要功能实现。

5.1 服务器实现

一般来说，实现一个比较简易的服务器可以使用 Node.js 的 Http 模块，但是在本系统中，为了使用社区一些较好的开发工具，我们使用 Node.js 下比较出名的 Express 服务器，在进行安装后，我们可以自行编写服务器的启动脚本，服务器脚本的程序流程图如图 5.1 所示。

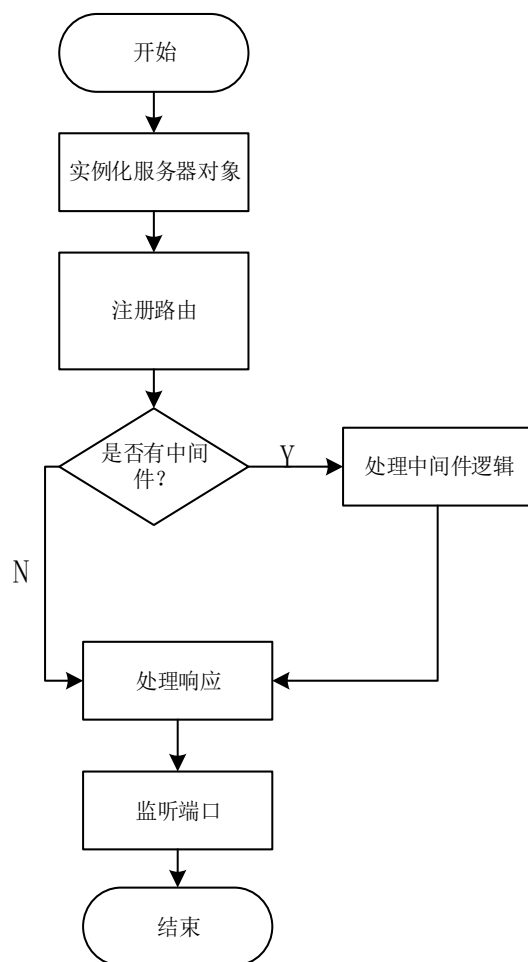


图 5.1 创建服务器程序流程图

从代码流程图可以看出，Node.js 实现一个网页服务器的工作十分简单，首先创建一个 Express 实例，然后根据注册路由，即判断 http 请求的路径与参数信息，之后判断是否有中间件，在目前流行的前端技术中，通常会有一些热更新，服务端渲染等等技术，这些技术统一被视为服务器中间件，由于中间件的种类和功能繁多，并且已经和本课题的研究内容相差较远，因此不做展开。中间件逻

辑处理完成后进行响应处理，最后在 express 实例上监听端口，完成服务器的创建。

在附录 A 中，列出了 Node.js 实现 web 服务器的完整代码，首先实例化一个 Express 服务，即代码清单中的变量 app，app 函数带有两个参数，一个是 req，一个是 res，分别代表 http 的请求对象和相应对象，对象之上添加的许多方法，请求和响应对象的方法列表如表 5.1 所示：

表 5.1 请求与响应的部分属性和方法列表

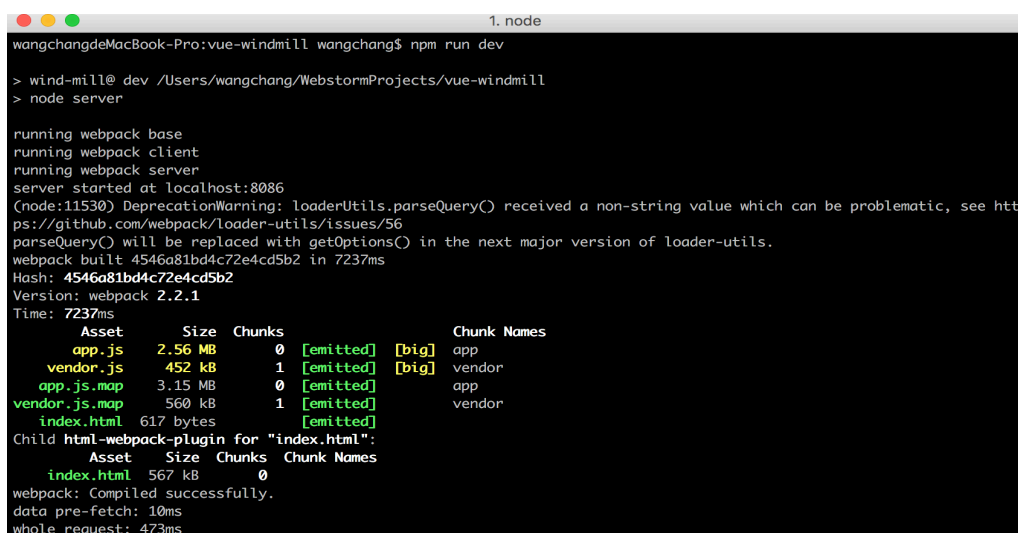
请求(req)/响应 (res)	方法/属性	方法/属性描述
req	path	请求的路径
req	accept()	检查 content-type 是否可以接受
req	get()	获得 http 协议的字段值
res	headersSent	检查是否已经返回了 http header
res	send()	发送响应
res	end()	结束响应

在代码的前 4 行，使用 app.use()函数实现了对静态文件的服务，这里的四个函数的参数含义分别是网站的标签栏图标，网站缓存清单，网站的根目录下的产出文件，网站的公开静态资源。

在代码的第 5 行到倒数 5 行，实现了一个 http 请求的监听，其中 app.get()函数接受了两个参数，第一个是路由的字符串，第二个是当请求到匹配的路由时执行的回调函数，代码中请求任意路径“*”时，将会执行第二个参数的回调函数。

在代码的第 6 行到倒数第 6 行，是回调函数的完整代码，在回调函数中关键的实现都是依托于 req 和 res 这两个参数所提供的方法，在附录代码中，它依次处理的逻辑是：处理未渲染异常、设置 http 请求的头部，发送 html，处理发送 html 出现的异常。

图 5.2 显示了本系统开启服务器过程中，控制台输出的打印信息。



```
wangchangdeMacBook-Pro:vue-windmill wangchang$ npm run dev
> wind-mill@ dev /Users/wangchang/WebstormProjects/vue-windmill
> node server

running webpack base
running webpack client
running webpack server
server started at localhost:8086
(node:11530) DeprecationWarning: loaderUtils.parseQuery() received a non-string value which can be problematic, see https://github.com/webpack/loader-utils/issues/56
parseQuery() will be replaced with getOptions() in the next major version of loader-utils.
webpack built 4546a81bd4c72e4cd5b2 in 7237ms
Hash: 4546a81bd4c72e4cd5b2
Version: webpack 2.2.1
Time: 7237ms
   Asset      Size  Chunks             Chunk Names
  app.js    2.56 MB          0  [emitted]  [big]  app
 vendor.js   452 kB          1  [emitted]  [big]  vendor
  app.js.map  3.15 MB          0  [emitted]
 vendor.js.map  560 kB          1  [emitted]
  index.html  617 bytes          0  [emitted]
Child html-webpack-plugin for "index.html":
   Asset      Size  Chunks             Chunk Names
  index.html  567 kB          0
webpack: Compiled successfully.
data pre-fetch: 10ms
whole request: 473ms
```

图 5.2 服务器控制台打印信息

在图 5.2 中可以看到主要在开启服务器过程中主要是 webpack 进行了编译和打包的工作，最后生成了五个文件，并且把 index.html 作为主要的服务器返回的主页面。

5.2 数据实时更新实现

数据实时更新方案采用第三章中介绍的固定时间轮询的方案。

固定时间轮询包含两方面的内容，一方面需要轮询获取数据，另一方面将数据显示在 web 界面上，在代码层面上，轮询的数据接口采用统一的一个接口来进行处理，通过计时器每隔固定时间进行一次请求，请求到的数据返回各前端 VUE 组件的数据接口中，由于采用的是固定时间轮询的方案，因此对固定轮询的时间指标需要有一个合适的判断和衡量，在开发测试中，由于自己的个人电脑性能可能不是很好，在实际开发和测试中采用了 10s 中作为轮询的固定时间。

在前端中，固定时间轮询采用的函数是 `setInterval()`，这个函数第一个参数接收一个匿名函数，第二个参数为时间，由于 JavaScript 单线程的特性，`setInterval` 函数将会每隔 10s 调用一下第一个参数的匿名函数，所以在代码层面上，只需要在第一个函数中进行异步请求获取相应接口数据，并且按照 Web Service 的返回结构进行数据解析和处理，最后添加到 DOM 上，进行视图的更新就可以了，整个数据实时更新的程序流程图如图 5.2 所示。

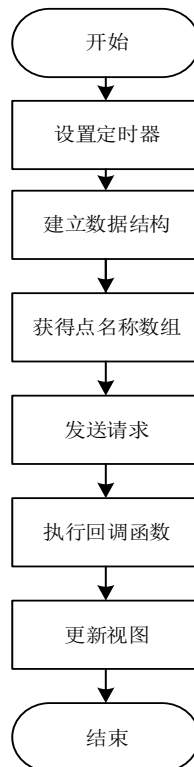


图 5.3 数据实时更新程序流程图

从流程图中可以看出，数据的实时更新是在客户端上进行的一次请求，在浏览器上每隔 10s 将会调用接口去从 web service 去请求数据，接口请求的数据之后通过回调函数的参数返回出来给外部处理，如此定时地循环往复实现了数据近乎实时的更新。附录 B 第一部分给出了从 Web Service 接口获得数据更新的 JavaScript 代码，第二部分给出了前端回调函数中对获得的数据进行处理的完整代码。

在第一部分的代码中，首先第 4 行到第 42 行，实现了所有风机的数据结构，这里使用一个 for 循环，出于性能的考虑，这里总共建立了 9 个风机数据接口，每个风机都是 JavaScript 语言的一个对象，9 个风机被存放在一个数组中，在 for 循环中我们可以看到，每个风机都有一个 10 个属性，他们所

代表的含义如表 5.2 所示

表 5.2 风机属性的数据类型与含义

属性	数据类型	含义
id	Number	该风机的 id
name	String	风机的名称, 以 A0 开头
status	Object	包含有对应的点名称的点的值
windspeed	Object	包含有对应的点名称的点的值
power1	Object	包含有对应的点名称的点的值
power2	Object	包含有对应的点名称的点的值
speed1	Object	包含有对应的点名称的点的值
speed2	Object	包含有对应的点名称的点的值
angle	Object	包含有对应的点名称的点的值
frequency	Object	包含有对应的点名称的点的值

第 44-60 行实现了点名称的集中获取, 按照 WebAccess 的 Web Service 接口规定, 获取点的值必须使用 HTTP post 方法, 并且需要传递点名称的数组作为参数(字符串化, JSON.stringify()), 在这一段代码中, 我们将上一步数据结构中的点名称收集起来进行集中的存储, 便于发送请求。

第 62-88 行实现了请求和回调函数的处理, 在请求之前需要设置 content-type 为 json 格式, 并且字符集为 UTF-8, 在请求中需要传递上一步获得的点名称列表, 请求成功的回调函数中, 由于 WebAccess 返回的数据结构是按照点名称来返回的, 因此这里需要将点名称和已有的数据结构进行匹配, 匹配之后存放在之前建立的数据结构的 value 属性中, 最后把整个的数据结构作为参数返回给外部回调函数调用

在附录 B 的第二部分代码中, 给出了外部回调函数的调用方式, api.getvalue()就是第一部分我们编写的对 Web Service 发起请求的函数, 由于发起请求采用的是异步的方式, 因此在这里被调用并且传递一个回调函数, 回调函数在请求成功后被执行, 在保证前后端的数据格式一致的情况下, 回调函数的主要内容很简单——将最终的数据直接复制给 Vue 的 data 对象的 items 属性, vue.js 会自动更新界面的数据更新。

5.3 系统主要功能实现

系统主要功能包含有实现双层架构, 上层实现监控, 上层内各个模块的功能。

双层架构, 是指包括下层的数据采集和上层的数据汇总, 本课题研究的是双层架构的上层部分, 作为双层架构的一个组成部分而存在, 因此实现双层架构只需要在上层搭建好 WebAccess 工程并且和下层联调成功就可以了。如第三章所介绍, 这里采用动态 IP 主动上报的方式来进行实现^[14]。为便于叙述, 我们在局域网环境下进行了上下层的联调, 功能实现的具体步骤为:

1. 在下层 WebAccess 工程的工程节点选项中, 选择“上移远程监控节点作为 WA SCADA 的点”, 之后在填写的工程节点 IP 地址中输入下层工程的 IP 地址, 如 192.168.1.123, Modbus 监听端口选择 504, 完成初始化。
2. 在下层的监控节点上选择上移代码到远程监控节点, 之后在 IP 地址输入框中输入上层的 IP

地址，程序提示完成后结束。

上层实现监控。由于使用 **Web Service** 只能进行数据的读取无法对数据进行控制，因此，对数据进行控制需要用到 **WebAccess** 本身的监控功能，所以一个比较合理的方式就是在上层的 **web** 监控系统中的网页里嵌入一个 **DRW** 绘制的网页^[15]，这样在网页中选择点击就可以进行对数据的控制了

上层模块，这里模块的含义是在前端架构中的不同功能模块，包含有风场信息模块，风机信息模块，实时图表模块，报警信息模块，操作记录信息模块。由于前端采用了 **vue.js** 作为主要框架，因此自然的采用 **vue.js** 的单文件组件来进行模块的组织。

vue.js 的单文件组件是一种更优的代码书写方式，它将内容相关的 **html**，**css**，**JavaScript** 的书写放在了一个文件中，并且有相应的工具插件来对代码进行转换，在编写代码的过程中，我们只需要将相关的 **HTML**，**CSS**，**JavaScript** 代码编写在一个 **.vue** 文件中，使用自动的打包工具，就可以完成一个模块的编写。

图 5.3 给出了编写一个 **VUE** 单文件组件的程序流程图。

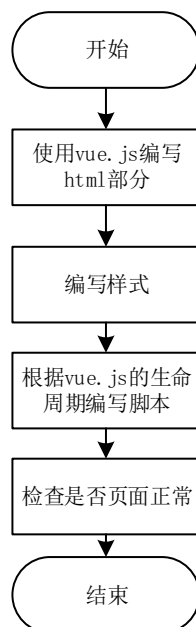


图 5.4 编写 **VUE** 单文件组件流程图

从流程图可以看出，编写单文件组件的主要难点在于脚本部分，根据第三章相关原理，需要在 **vue.js** 的相关生命周期注册不同的事件，同时在不同的模块中，需要完成定时从接口获取数据并且完成数据信息的解析和转换。附录 C 给出了风机信息模块的完整代码。

在完成了不同模块的单文件组件编写之后，使用 **npm run dev** 命令开启服务器的过程中，**webpack** 会自动的对文件进行编译和解析，最终生成了一个存在与内存当中包含有所有资源与代码的单文件的页面，在工程目录下，我们可以把具体的问价内容输出到文件当中去，包含有一个 **index.html**，一个样式表文件 **style.css**，一个脚本压缩混淆文件 **app.js** 以及脚本库打包文件 **vendor.js**，这些文件的后缀名中都添加了 **md5** 戳用户缓存控制，打开 **index.html** 发现这个网页文件对上面的四个文件都引用了进来。在生产模式下，使用 **npm start**，服务器会以 **dist** 目录作为服务器的根目录，同样监听本机的 8086 端口，在浏览器中输入 <http://localhost:8086>，系统的主界面如图 5.5 所示。

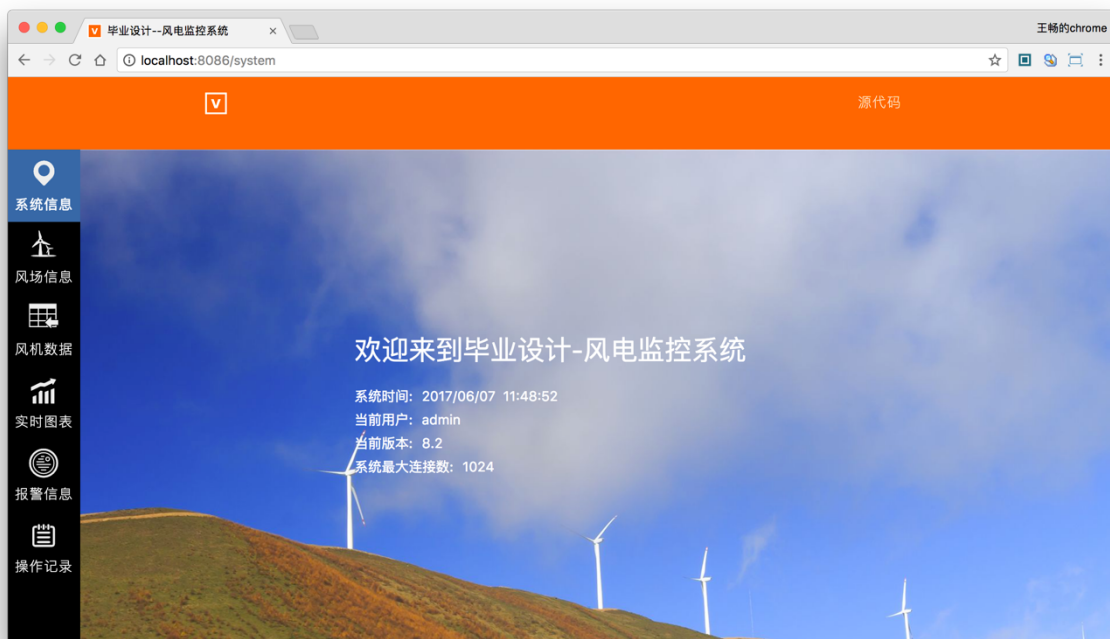


图 5.5 系统主界面

左侧的 6 个模块就是我们编写的单文件组件的视图，点击可以进行跳转。图 5.6 展示了风机信息界面的视图。

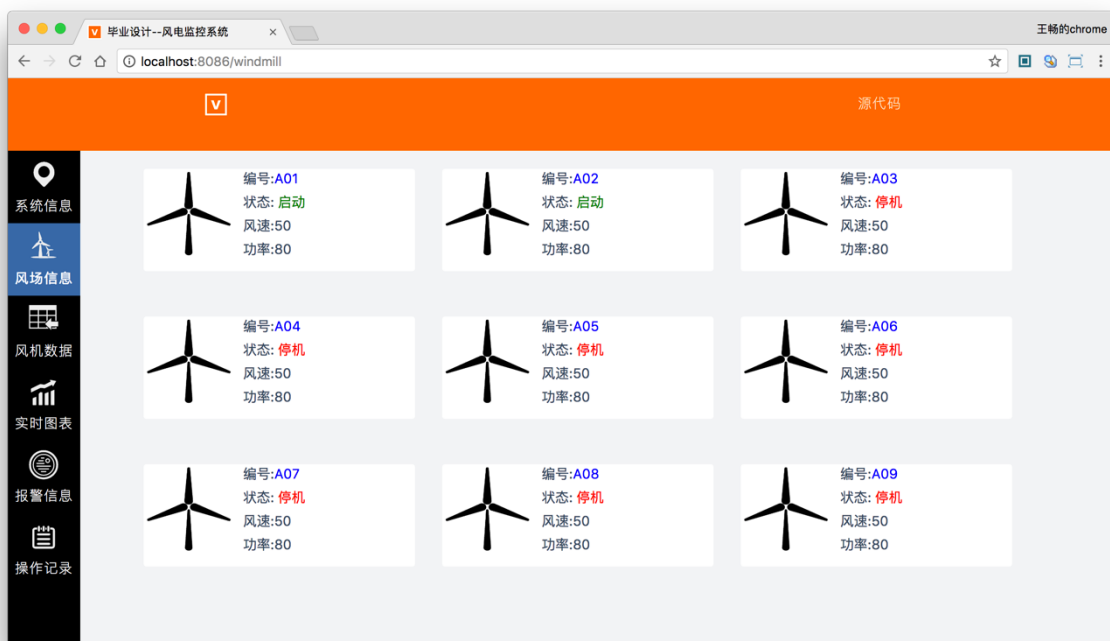


图 5.6 系统风场信息界面

风机数据、实时图表、报警信息和操作记录视图与此类似，只是页面中信息汇总的内容不一样，这里就不在一一展开赘述了。

5.4 软件部署与上云

在完成上层服务器数据展示系统的开发并进行可靠性测试后，我们可以将它部署到云服务器上，

云服务器是可以进行外网访问的服务器，实际上任何一个产品最终要被用户访问到都需要提供一个 Internet 的入口，这个入口就是云服务器的 IP 地址。云服务器提供了域名解析和弹性存储等功能，目前国内较为知名的云服务器厂商有阿里云，腾讯云等。在购买完合适配置的服务器之后，下面简要介绍本软件的上云流程。

由于 WebAccess 软件需要安装在 windows 系统上，因此，首先需要连接到远程的 windows 服务器上，为避免卡顿可以选择一个性能比较好的 windows 电脑，在程序或者开始中搜索打开一个 windows 远程桌面连接，然后在里面输入云服务器的 IP 地址(前提是云服务器已经启动并且在运行，这一部分不同的服务商管理方式不同，具体可以查看帮助或者寻求服务商的技术支持)，然后像登陆自己的电脑一样输入购买时填写的用户名和密码就可以了。

进入到远程 windows 下之后，完成 WebAccess 的安装，并按照第三章的相关流程完成下层 WebAccess 的配置，这个过程和在本地完成 WebAccess 工程搭建是一样的，安装好 WebAccess 之后，需要下载 node.js 完成上层服务器的建立。上层服务器的建立流程如下：

1. 下载并安装 Node.js，注意 Node.js 版本需要大于 6,安装完成之后能够在 CMD 控制台使用“node -version”命令能够打印出 node 版本，说明安装成功。
2. 下载上层系统的源代码，目前所有的源代码都已经托管在网上的 GitHub 社区，代码的地址为 <https://github.com/feaswcy/windmill.git>
3. 下载完成代码之后，使用命令行进入工程目录，首先执行 npm install 命令下载所有需要的依赖包，等待下载网域后执行 npm build 命令，将会生成一个 dist 文件夹，文件夹内有打包好的 html 文件和 JavaScript 文件。
4. 使用 npm start 命令启动服务器，稍等一段时间等待命令行界面提示服务器开启成功之后，上层服务器的启动内容就完成了
5. 使用浏览器访问云服务器的 IP 地址的对应端口号，将会看到上层服务器的系统展示。

需要注意的是，如果是自己搭建的适用于另外工程场景下的 WebAccess 工程，那么可以对当前的代码进行改造和扩展。在搭建公网服务的时候，如果在购买云服务器的时候同时购买了域名服务，那么也可以通过域名来访问该系统，此时需要对域名进行解析。

5.5 本章小节

本章主要介绍了系统的总体设计，从分层和模块的角度对系统的架构进行了概括，并对每一层或者每个模块的功能进行了叙述。

从分层的角度，本系统可以分成六层，分别为展示层，Node.js 服务层，Web Service 层，WebAccess 工程层，监控层，数据采集层，并且对每一层的功能进行了详细的解读。

从模块的角度，对不同模块的功能进行了介绍，同时概括了不同模块的主要内容，之后给出了模块的不同内容的信息接口。

结束语

2017 年开学伊始，在学校老师的指导安排下，我就开始着手了毕业论文的相关工作，刚开始时由于之前在研华接触过 WebAccess 组态软件，心中觉得应该不是很困难，因此只是对论文的内容进行了大概的构想。一直到准备学校组织的开题报告答辩时，我才意识到虽然之前使用过 WebAccess 这个组态软件，但是对这个软件的理解非常浅薄，而且由于平时很少看一些论文著作，对论文的写作也一时感觉难以下手，不过随着和研华的企业指导老师的沟通以及对 WebAccess 这个软件的实际使用，我对课题的工程背景以及整个系统应该是什么样子的越来越清楚了。

于是整个毕业设计就有条不紊的展开了，在一次去研华进行毕业设计答疑的过程中，研华的指导老师说我希望你能用 Web Service 功能完成这次毕业设计时，我马上就答应了下来——我在大二时就加入了学校的编程社团，并且相关的经历还算丰富 Web Service 技术我以前就有所研究和应用，我自己对将传统软件和互联网结合起来的这个工程项目也感到十分有兴趣，我查阅了相关的技术书籍，在开题答辩前对整个工程的技术架构已经非常清楚了，因此一边在完成代码的编写，一边忙着进行毕业论文相关文献的阅读和学习，随着时间的积累，我的浏览器组态软件的上层服务器数据汇总，在中期答辩前已经基本成型了，在这之后，我和学校的督导老师周亚丽老师进行了多次答疑，主要是对毕业论文的组织思路和论文的书写要求进行了详细的了解，在一次次的修改之后，我的论文在五月下旬基本完成了。

在整个毕业设计的过程中，对于我来说比较困难的还是技术的实现方面，虽然之前会一些网页编程的技术，但是在做这个工程的过程中，还是有很多技术在编程方面感到十分困难，有些功能我觉得非常好想要加上去，却发现独立一人工作量十分巨大，最终只能无奈舍弃，但是总体来说我对这次的工程项目是非常满意的，在工程方面，研华的 WebAccess 组态软件在工业工控领域有着大量的应用，在研究组态软件的过程中我对工业控制行业也有了更多的理解，同时在上层架构的搭建过程中，我使用了许多目前非常流行的互联网编程技术，比如 Web 数据实时通信方案、前端构建打包技术，服务器轮询技术等。这些技术已经在很多地方都有所应用，因此对我来说技术上的收获是很大的。在论文的写作方面，除了学习了一些论文的写作技巧之外，我更懂得了踏实做事的道理：当你把一件问题分解成一个个的步骤之后，一步步的完成，最终一定能够有所收获。

参考文献

- [1]刘忠超, 张燕, 尉乔南. 组态软件实用技术教程[M].西安: 西安电子科技大学出版社, 2016
- [2]周力尤, 罗隆, 谢雪芳. 组态软件技术与应用[M].北京: 电子工业出版社, 2012
- [3]姜建芳. 西门子 WinCC 组态软件工程技术[M].北京: 机械工业出版社, 2015
- [4]张仁杰. 网际组态软件 Advantech Web Access 应用技术[M].北京: 机械工业出版社, 2011
- [5]何坚强, 薛迎成, 徐顺清. 工业组态软件及应用[M].北京: 北京大学出版社, 2014
- [6]Darren Cook. HTML5 数据推送应用开发[M].北京: 人民邮电出版社, 2014
- [7]陈寅生. 基于 WebAccess 的燃气表远程监控管理系统的设计与实现[D].北京: 北京邮电大学, 2013
- [8]邵子惠. 基于网际组态软件 WebAccess 的 PLC 综合远程控制系统研究[D].成都: 西华大学, 2008
- [9]田凡顺. 网际组态软件 WebAccess 在配电网监控系统中的应用[D].西安: 西安电子科技大学, 2014
- [10]宋志崇. 监控组态软件的研究与设计[D].大连: 大连理工大学, 2008
- [11]左佳儒. 基于 WebAccess 的水位监控系统设计与实现[D], 北京: 北方工业大学, 2008
- [12]张佳文. 基于 WebAccess 的楼宇自动化系统的设计与实现[D].北京: 北方工业大学, 2010
- [13]任振东. 基于 WebAccess 的热力系统远程监控的实现与优化控制的研究[D].北京: 北方工业大学, 2009
- [14] Jinping LIU, Lei LIU, Wei ZOU. Energy Monitoring and Management System for Solar Sanitary Hot Water Heater Based on Advantech WebAccess[J]. Measurement & Control Technology, 2012, 31(1):34-37
- [15] Jian Liu, Xiaoqin Lian, Xiaoli Zhang, Chongchong Yu. Automatic Control System of Intelligent Building Based on WebAccess[J]. IEEE Proceedings, Intelligent Control and Automation, 2008, WCICA 2008:7079-7084

致谢

持续三个多月的毕业设计和论文写作工作终于接近尾声了，我的校园生活也快要到达终点，大学四年的生活如白驹过隙，一晃而过，马上我们将要进入社会开始新的征程，回想起大学四年的生活，有时会反问自己大学究竟意味着什么，是学习到了专业的技能能够在社会上立足，还是认识了一帮志同道合的朋友？我想大学这个阶段最重要的是教会了我独立思考和融入社会的能力，无论是学习，工作还是生活，我都从身边比我优秀的人身上获取到了很多帮助，也看到了一些学长学姐在毕业之后融入社会的坎坎坷坷，四年的生活这些身边人都给我很多的启发和感悟，也让我从一名学生的思考方式慢慢地转变成成年人的思考方式，在此，首先要感谢的就是在大学里认识到的这些朋友、同学和老师，谢谢你们在这四年中带给我的帮助，让我学会了在遇到挫折时独立思考与成长。

在毕业设计的过程中，我首先想要感谢的就是我的两位老师，在研华的企业指导老师卢晶晶老师和学校的督导老师周亚丽老师。卢晶晶老师在工程实践方面对我和另外一名同学进行了多次的答疑，每次去见卢老师之后都感觉收获很多，卢老师为人真诚友善，每次都对我们的问题进行详细的解答。周老师在工作上认真负责，在我进行毕业设计的过程中给予了我很多的帮助，并且对我的论文内容投入了很多的时间进行了仔细的指导。在毕业设计完成的过程中离不开两位老师对我的帮助，在此表示衷心的感谢。

其次我想要感谢的就是我在滴滴进行实习时的同事和领导，因为这次课题相对而言编程工作量比较大，而我恰好在滴滴做相关的实习工作，因此在工程代码的编写过程中向他们咨询过，他们都是经验非常丰富的工程师，在代码的编写和架构上给了我很多非常实用的建议和帮助，如果技术上没有他们的指导，我独自完成这样一个软件系统十分困难，在此感谢他们对我全力支持以及无私的帮助。

在这三个多月的论文撰写和修改中，我的室友和同学也给了我许多的帮助，特别是同组的同学汪楷迪，我们分别完成上下层课题的过程中进行了多次的联调合作，大家相互共享书籍资料，一起讨论论文的组织思路，在生活上相约一起学习，让我在这几个月的时光中能够积极踏实地完成毕业设计的工作内容，在此也对他们表示感谢，感谢他们作为同学和朋友在这段时间的陪伴和付出。

最后，还要感谢一下撰写相关资料的各位前辈学者，在写论文的过程中他们的一些书籍对我有莫大的帮助和启发，感谢他们所完成的研究，让我站在了巨人的肩膀上，最终顺利地完成了论文。