**学校代码：10200　　　　　　　　 学号：2016012071**

****



本科毕业论文

**中医诊所管理系统的设计与实现**

**Design and implementation of management system of TCM clinic**

作者：**韦佳妮**

指导教师：**邹向华 讲师**

学科专业：**软件工程**

学 院：**信息科学与技术学院**

东北师范大学学位评定委员会

2020年 5月

摘 要

党的十九大报告提出，要坚持中医和西医并重，推动中医以及中医药发展。借助“互联网＋中医”，进一步推进中医以及中医药的事业发展成为了一大趋势。中医诊所作为中医事业的重要发展阵地，更是要积极地进行创新变革和与时俱进。

传统的中医诊所管理需要消耗大量的人力资源和物力资源，门诊中大量的信息资料是由人工操作的，这样不仅会导致错误率较高，而且管理效率低下，同时还发现，由于就诊后的客户管理混乱，导致了患者流失的问题。在信息高速发展的时代，传统的管理方法已经不符合现在的中医事业发展需求。采用中医诊所信息化管理，可以极大地提高门诊各个岗位的工作效率，该系统有预约挂号、就诊、收费、病历管理等功能，除此之外，还有中医的问候提醒功能，这样能够建起立良好的医患关系，体现了中医的特色关怀。

本论文将会从项目需求分析、设计、以及具体实现等各个方面进行详细的介绍。项目整体结构采用b/s体系结构，客户端开发采用Vue.js并结合Css、Html、Elemenu-ui进行开发，服务端采用Node.js进行开发，数据库采用MongoDB进行开发。系统功能主要有就诊管理、医生工作站、药房管理、客户管理、运营数据分析等功能。该项目的前期调研阶段，包括与中医诊所工作人员进行交流、网上查询相关资料、整理数据，项目中期的设计阶段，包括总体系统设计、模块设计、数据库设计，以及项目后期阶段，包括客户端开发、服务端开发以及数据库开发等工作，均是由本人独自完成。目前，该系统已经开发完成。

**关键词：**互联网+中医；中医诊所；信息化管理；b/s

**Abstract**

According to the report of the 19th National Congress of the Communist Party of China, we should attach equal importance to traditional Chinese medicine and Western medicine, and promote the development of traditional Chinese medicine and traditional Chinese medicine. With the help of "Internet + traditional Chinese medicine", further promoting the development of traditional Chinese medicine and traditional Chinese medicine has become a major trend. As an important position for the development of traditional Chinese medicine, the clinic of traditional Chinese medicine should actively carry out innovation and reform and keep pace with the times.

Traditional Chinese medicine clinic management needs to consume a lot of human and material resources. A large number of information materials in the clinic are manually operated, which will not only lead to high error rate, but also low management efficiency. At the same time, it is found that due to the confusion of customer management after treatment, the loss of patients is caused. In the era of rapid development of information, the traditional management methods have not met the needs of the development of traditional Chinese medicine. The information management of TCM clinic can greatly improve the work efficiency of each post in the outpatient department. The system has the functions of appointment registration, medical treatment, charging, medicine management, medical record management, etc. in addition, it also has the function of greeting and reminding of TCM, which can establish a good doctor-patient relationship and reflect the characteristic care of TCM.

This paper will be from the project requirements analysis, design, as well as specific implementation and other aspects of a detailed introduction. The overall structure of the project adopts B / S architecture. Vue.js is used for client development, combined with CSS, HTML and elemen UI. Node.js is used for server development and mongodb is used for database development. The main functions of the system are: medical treatment management, doctor workstation, pharmacy management, customer management, operation data analysis and other functions. The early research stage of the project, including communication with TCM clinic staff, online query of relevant information, data sorting, the middle design stage of the project, including the overall system design, module design, database design, and the later stage of the project, including client development, server development and database development, are all completed by myself. At present, the system has been developed.

**Key words:**Internet plus Chinese medicine; Chinese medicine clinics;information management;B/S

# 

# 目 录

[摘 要 I](#_Toc323898592)

[Abstract II](#_Toc323898593)

[目 录 I](#_Toc323898594)V

[1 概述 1](#_Toc323898596)

[1.1 项目研究的背景和意义 1](#_Toc323898597)

[1.2 国内外研究现状分析 2](#_Toc323898598)

[1.3 项目目标 2](#_Toc323898598)

[1.4 主要工作 2](#_Toc323898598)

[2 相关技术综述 3](#_Toc323898602)

[2.1 Vue框架 3](#_Toc323898603)

[2.1.1 Vue框架概述 2](#_Toc323898599)

[2.1.2 Vue框架特点 2](#_Toc323898599)

[2.2 Node.js 4](#_Toc323898604)

[2.2.1 Node.js概述 2](#_Toc323898599)

[2.2.2 Node.js特点 2](#_Toc323898599)

[2.3 MongoDB数据库 4](#_Toc323898604)

[2.3.1 MongoDB数据库概述 2](#_Toc323898599)

[3 系统分析 5](#_Toc323898605)

[3.1 项目概述 5](#_Toc323898606)

[3.1.1 问题分析 2](#_Toc323898599)

[3.1.2 用户范围 2](#_Toc323898599)

[3.2 需求分析 5](#_Toc323898606)

[3.2.1 可行性分析 2](#_Toc323898599)

[3.2.2 功能性需求 2](#_Toc323898599)

[4 系统设计概要 6](#_Toc323898609)

[4.1 系统架构及原理 6](#_Toc323898610)

[4.2 UML建模 6](#_Toc323898611)

[4.2.1 就诊模块 2](#_Toc323898599)

[4.2.2 医生工作站模块 2](#_Toc323898599)

[4.2.3 客户管理模块 2](#_Toc323898599)

[4.2.4 运营数据分析模块 2](#_Toc323898599)

[4.3 数据库设计 6](#_Toc323898612)

[4.3.1 患者详细信息集合 2](#_Toc323898599)

[4.3.2 预约挂号记录集合 2](#_Toc323898599)

[4.3.3 缴费记录集合 2](#_Toc323898599)

[4.3.4 诊间缴费集合 2](#_Toc323898599)

[4.3.5 医生详细信息集合 2](#_Toc323898599)

[4.3.6 科室信息集合 2](#_Toc323898599)

[4.3.7 医生药方集合 2](#_Toc323898599)

[4.3.8 病历集合 2](#_Toc323898599)

[5 项目详细设计与实现 7](#_Toc323898614)

[5.1 就诊模块设计与实现 7](#_Toc323898615)

[5.1.1 就诊模块设计 2](#_Toc323898599)

[5.1.2 就诊模块代码实现 2](#_Toc323898599)

[5.2 医生工作站模块设计与实现 7](#_Toc323898616)

[5.2.1 医生工作站模块设计 2](#_Toc323898599)

[5.2.2 医生工作站模块代码实现 2](#_Toc323898599)

[5.3 客户管理模块的设计与实现 7](#_Toc323898616)

[5.3.1 客户管理模块设计 2](#_Toc323898599)

[5.3.2 客户管理模块代码实现 2](#_Toc323898599)

[5.4 运营数据分析模块的设计与实现 7](#_Toc323898616)

[5.4.1 运营数据分析模块设计 2](#_Toc323898599)

[5.4.2 运营数据分析模块代码实现 2](#_Toc323898599)

[6 软件测试 7](#_Toc323898614)

[6.1 就诊模块测试 7](#_Toc323898615)

[6.1.1 预约挂号功能测试 2](#_Toc323898599)

[6.1.2 费用记录功能测试 2](#_Toc323898599)

[6.1.3 诊间缴费功能测试 2](#_Toc323898599)

[6.1.4 患者个人信息功能测试 2](#_Toc323898599)

[6.1.5 医生加号病人功能测试 2](#_Toc323898599)

[6.1.6 医生对患者进行诊断功能测试 2](#_Toc323898599)

[6.1.7 医生药方管理功能测试 2](#_Toc323898599)

[6.2 医生工作站模块测试 7](#_Toc323898616)

[6.2.1 查看病历功能测试 2](#_Toc323898599)

[6.3 客户管理模块测试 7](#_Toc323898616)

[6.3.1 查看客户列表功能测试 2](#_Toc323898599)

[6.4 运营数据分析模块测试 7](#_Toc323898616)

[6.4.1 查看经营数据分析功能测试 2](#_Toc323898599)

[6.4.2 查看医生的业绩分析功能测试 2](#_Toc323898599)

[6.4.3 查看患者情况分析功能测试 2](#_Toc323898599)

7 工作[总结与展望 8](#_Toc323898617)

[7.1 总结 7](#_Toc323898616)

[7.2 展望 7](#_Toc323898616)

[参考文献 9](#_Toc323898618)

[致 谢 12](#_Toc323898619)

1 概述

1.1 项目研究的背景和意义

我国的中医文化历史悠久。中医是我国几千年继承下来的珍贵宝物，它拥有着属于自身的优势和特点，为治疗许多的疑难杂症提供了帮助。 近年来，互联网在不断地发展壮大，众多行业都在积极与互联网联合融入。我国正在推动各个行业与互联网的联结，鼓励行业实施“互联网+”。随着“互联网+”理念的出现，作为中华文化瑰宝的中医，自然也不能置身事外，我国也在不断加大对“互联网+中医”的投入，让更多的人民百姓得益于中医。

根据调查了解，很多医院都采购医疗信息化管理系统，利用互联网技术来对整个医院实施信息化管理。例如佛山市顺德区大良医院就运用了“互联网+”理念，制定了新的医院管理政策，采用医疗信息化管理系统，重心在于自动生成电子病历，目的是降低误诊率，提高医疗质量。医院采用信息化的管理方法，让看病就诊更加合理化和智能化，提升了医院的服务质量，也便于医院管理者制定合理的决策。

根据国家卫健委数据显示，2018年中医诊所达到4万多家，比2017年增加将近5千家。其他资料显示，2020年中医治疗量与总治疗量的比值达到了大约30%。由此可见，大量的中医诊所正在建立，中医事业正在逐步发展壮大。然而中医事业的发展也在面临着一些问题，中医是起步较晚的行业，在中医事业的管理上还缺乏经验。中医在信息化管理意识薄弱，现在的大部分中医诊所还是使用传统的管理办法，很多信息记录还是基于手工操作的，信息管理混乱，以至于经常发生文件丢失、查找资料效率低下、误诊等情况。当前我国的中医事业发展情况与我国的中医发展目标还存在着距离，所以有必要开发出一个符合我国中医事业发展情况的中医诊所管理系统。

1.2 国内外研究现状分析

1.2.1 国内研究现状分析

中医是我国古代人们经过长期医学实践研究而总结出来的医学理论体系，是我国的国粹。我国正在积极发展中医以及中医药事业。根据国家卫健委数据显示，从2014年至2018年，我国的中医诊所数量呈增长趋势，从2014年的3万多家增长到了2018年的4万多家，可见我国的中医事业的巨大的发展潜力。针对中医事业的热潮的掀起，我国制定了相关的中医药发展战略，正在不断加大对中医药事业的投入，推动中医药事业的发展。根据《2019年中国中医诊所市场前景研究报告》的数据显示， 从2014年至2019年，国家就推动中医以及中医药事业的发展颁布了诸多政策，包括《中华人民共和国中医药法》 、 《中医诊所备案管理暂行方法》 、 《关于加强中医药健康服务科技创新的指导意见》等。由此可见国家对中医事业的高度重视。近年来，我国互联网、大数据以及人工智能技术蓬勃发展，中医事业也因此得到这些新技术的支持。然而，作为推动中医事业的发展发展的中医诊所目前还面临着许多困难，例如中医诊所还没有一个规范化的管理模式、中医诊所缺乏中医的特色、没有适合的营销手段等。针对中医诊所所存在的问题，我国借助互联网的高速发展，正在推动中医事业与互联网的融合发展，致力于中医事业的健康与蓬勃发展。

1.2.2 国外研究现状分析

中医不仅服务于中华民族，国外的许多地区也得益于中医。中医正在被越来越多的国家和地区认可，在有的国家已经获得了合法地位，还被纳入医疗保险系统。中医诊所也在160多个国际被建立起来，设立诊所的国家分布在东南亚和欧洲较多，分布在非洲地区相对较少。和我国类似，许多国家也支持本国的中医事业的发展，但是大部分还是处于一个初级发展阶段，中医诊疗对国外地区来说更多的是作为一个辅助治疗方案，所以国外对中医的研究和还不够透彻，以及制定的中医诊所的管理机制还不够成熟。国外的许多地区投入大量资金，致力于创建完善的医疗信息化管理系统，采用数字化的运作方式，然而这些依托互联网的信息化管理还未渗透到中医诊所的管理当中。因此，建立一个完善的、符合中医事业发展的中医诊所管理系统是非常有必要的，不仅有利于国内的中医事业发展，国外的中医事业发展也能得益于此。

1.3 项目目标

运用所学知识和技术，结合中医诊所的实际特点，设计实现一个能够简便操作的、可对中医诊所信息进行高效管理的中医诊所管理系统。

该系统主要角色分为病人、门诊医生、门诊管理者。系统功能主要有就诊管理、医生工作站、客户管理、运营数据分析等功能。

（1）就诊管理实现患者预约缴费、查看预约、就诊期间缴费等功能以及实现医生加号病人、线上配制处方、查看病历等功能。

（2）医生工作站模块实现存储患者病历的功能，提供医生查看病人病历、选用智能处方模版，也可提供医生查看相似病历，使得医生就诊更加高效化。

（3）客户管理实现对病人就诊后的问候功能，能够建起立良好的医患关系，也能够获体现中医的关怀特色。

（4）运行数据分析实现经营数据分析、医生的业绩分析、患者情况分析等功能，通过echarts实现数据的可视化。

1.4 主要工作

（1）通过百度搜索在网上查找现在市场上的各个中医诊所管理系统平台，对比各个平台产品，并分析总结出各个平台产品的特色和优点及缺点，再根据各个平台产品的信息提出创新改进方案。

（2）在网上查找关于中医发展以及中医诊所的相关资料，包括相关文献和新闻资讯，同时也阅读相关书籍，参考这些资料所阐述的研究方法和内容，再确定好研究计划。

（3）确定项目所采用的技术，确定整体结构采用B/S体系结构，该结构具有高效性、可靠性的特点。项目采用MVC前后端分离式开发中，客户端开发使用HTML、CSS结合Vue.js技术进行开发，服务端开发采用Node.js进行开发，数据库的开发采用MongoDB。

（4）对项目进行整体设计，设计中医诊所管理系统每个功能模块，对项目数据进行设计，设计项目的页面视觉样式。

（5）学习项目所使用到的技术，包括Vue.js、Node.js、MongoDB。

（6）开始编写项目代码，进行实际开发。

（8）测试项目，确定项目质量。

2 相关技术综述

2.1 Vue框架

2.1.1 Vue框架概述

Vue是一套用于构建用户界面的**渐进式框架**。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与[现代化的工具链](https://cn.vuejs.org/v2/guide/single-file-components.html)以及各种[支持类库](https://github.com/vuejs/awesome-vue" \l "libraries--plugins" \t "https://cn.vuejs.org/v2/guide/_blank)结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。[1]

MVVM框架如图2.1.1.1所示。

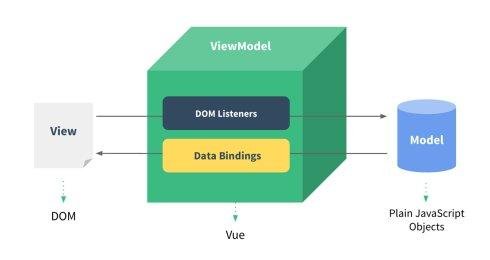


图2.1.1.1 MVVM框架

2.1.2 Vue特点

（1）双向数据绑定

在创建Vue实例时，传入对象中的data属性是数据中心。对于data中的每个数据，Vue通过Object.defineProperty来定义数据的特性，在数据的getter中定义方法来添加订阅者，在数据的setter中定义方法来通知观察者数据的变化，从而让观察者触发相应的回调事件，更新视图。另一方面，添加观察者是在模板解析的过程中进行的，添加观察者的同时定义响应回调函数从而更新视图。

Vue双向绑定如图2.1.1.1 MVVM框架所示。

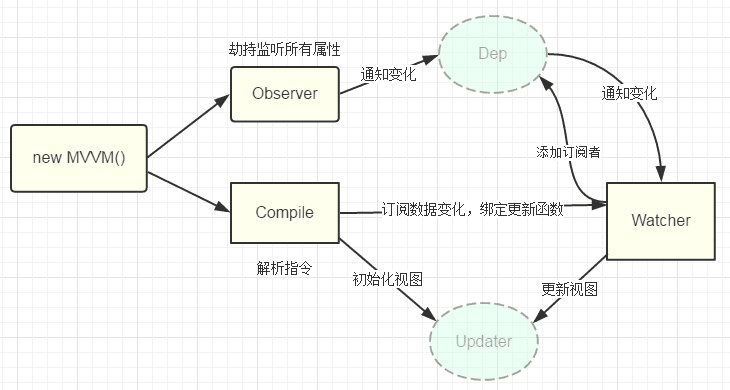


图2.1.2.1 Vue双向绑定原理

（2）组件化

使用Vue来开发单页面应用，可以通过组件化配合上Vue-Router的方式来分别展示页面的指定部分。使用组件化开发，能够提高代码的复用率和开发效率，同时也让代码更加整洁。组件化的开发是通过在父组件中注册子组件的方式完成的，每个组件拥有属于自身的视图和方法，各个组件互不干扰，可以通过特定的方法实现组件间的通信，包括父组件与子组件、子组件与父组件、平行组件的通信。在子组件中也可以注册组件，进而形成了Vue组件树。

Vue组件树如图2.1.2.2所示。



图2.1.2.2 Vue组件树

（3）虚拟DOM

当数据发生变化时，页面要进行更新。虚拟DOM可以对DOM元素进行批量操作，减少浏览器的回流次数，提升页面性能。

2.2 Node.js

2.2.1 Node.js概述

简单的说Node.js 就是运行在服务端的 JavaScript。Node.js 是一个基于Chrome JavaScript 运行时建立的一个平台。Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎，V8引擎执行Javascript的速度非常快，性能非常好。[2]

2.2.2 Node.js特点

（1）单线程

操作系统只创建一个线程来执行Node.js代码。实现宏观上的并行执行是通过非阻塞I/O、事件循环完成的。单线程的优点在于减少操作系统由于创建线程和销毁线程所花费的时间和资源。

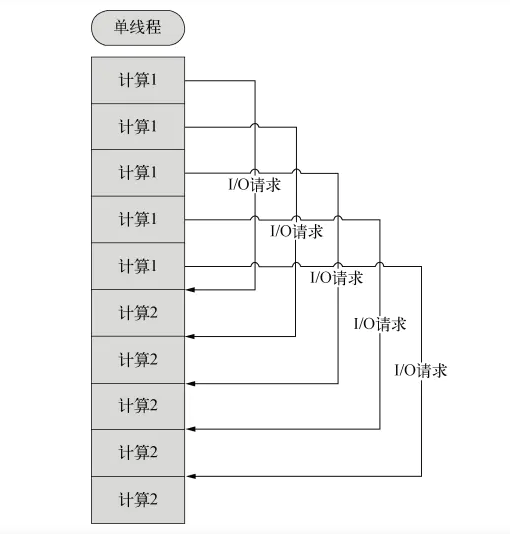


图2.2.2.1 Node单线程

（2）非阻塞I/O

由于node.js是单线程的，当遇到异步任务时，比如读取数据库，那么就会先等待结果返回才可以继续执行，等待的这一段时间什么事情都不做，代码执行效率大大降低。Node.js的非阻塞I/O机制就是用来解决node.js的单线程特点带来的问题的，当遇到异步任务时，会先将其抛入任务队列里，接着继续执行同步代码，这样就不必等到异步任务有结果了才继续执行。当执行完同步代码后，查看异步任务的返回的结果，再执行相应的回调函数。

（3）事件驱动

Node.js处理异步任务是通过事件循环机制进行的。当进入到事件循环后，会分为几个不同的阶段，分别是timer阶段、IO callbacks阶段、idle，prepare阶段、poll阶段、check阶段、closing阶段。在每一个阶段都会先检查是否有当前阶段的回调函数，然后检查是否有process.nextTick任务，再检查是否有微任务，最后退出该阶段进入下一阶段。

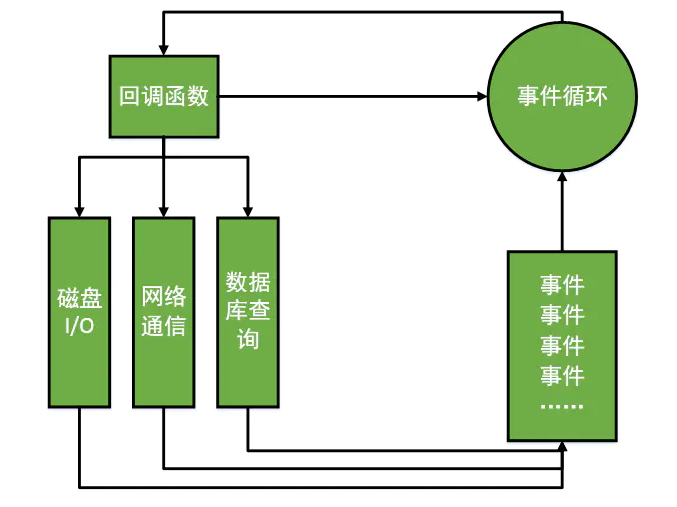


图2.2.2.2 Node事件循环

2.3 MongoDB数据库

2.3.1 MongoDB数据库概述

MongoDB 是由C++语言编写的，是一个基于分布式文件存储的开源数据库系统。在高负载的情况下，添加更多的节点，可以保证服务器性能。MongoDB 旨在为WEB应用提供可扩展的高性能数据存储解决方案。MongoDB 将数据存储为一个文档，数据结构由键值(key=>value)对组成。MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档，数组及文档数组。[3]

3 系统分析

3.1 项目概述

3.1.1 问题分析

我国高度重视中医以及中医药事业的发展。随着我国国民消费水平的提高，人们越来越注重养生，中医市场需求不断扩大。目前，中医诊所的规模在不断增长，对中医诊所的服务要求也越来越高。然而，我国中医事业还处于初级发展阶段，中医诊所的发展存在着以下问题：

（1）管理效率低下：我国的大部分中医诊所还在使用传统的管理手段，诊所就医流程不规范，患者排队等待时间过长，并且诊所医生还是使用手写纸质药方和病历，这样在来回传递过程中可能出现处方错乱的情况，导致误诊。纸质病历带来的另一个问题是不利于保存和查找且已丢失易，当医生想要查找类似病历时，查找效率低下，既费时又费力。

（2）人才缺乏：中医事业的从业人员年龄受到限制，虽然中医诊所规模在不断地扩大，然而我国中医方面的专业人才却不多，中医事业的发展出现了矛盾。人才的缺乏不利于中医诊所的长远发展，再加大人员培养的同时，也应该在中医诊所的管理上下功夫。

（3）缺乏营销手段：大部分的中医诊所的营销方式依靠的是口碑，这样的营销手段效果不明显，难以维持诊所的发展。制定正确合理的营销手段还需要依靠数据统计分析，而现在的中医诊所还没有适合的统计数据工具。

3.1.2 用户范围

该项目的角色分为三种，包括病人、门诊工作人员（医生、护士、药剂师等）、门诊管理者。每个角色的功能如下：

（1）病人：病人可以通过该系统进行挂号和预约挂号，在挂号和预约挂号的同时选择科室、医生、时间，可以在就诊期间进行线上支付，通过在线支付领取药方。

（2）门诊工作人员：门诊管理人员包括医生、护士和药剂师，医生可以加号病人、编辑药方、选择智能药方以及查看电子病历。药剂师根据医生的药单提前准备好药材等待患者取药。

（3）门诊管理者：门诊管理者可以对诊所分析的数据进行分析统计，以便于制定合理的运营手段。通过分析规划诊所的未来发展。

3.2 需求分析

3.2.1 可行性分析

（1）国家支持：中医作为我国的特色文化，我国正在不断加大对中医事业的投入。在国家的支持下，我国的中医诊所规模也不断扩大，国家制定了相关政策来规范中医诊所的运营，使中医诊所的运营更加高效、合理化，使百姓能获得更加优质的中医服务。

（2）技术发展：我国正处于技术热潮的冲击中，新的技术不断涌现，例如大数据、5G、人工智能等，这些新技术的发展也在不断推动中医诊所的发展。在技术不断沉淀过程中，中医诊所也正在与时俱进，不断变革。

（3）消费水平提高：根据调查显示，改革开放以来，我国经济快速增长。我国全体居民人均年度可支配收入不断提高，从2013年的1.8万元增至2018年的2.8万元。国民的消费水平不断提高，使国民对健康的需求也越来越高，而中医具有的治未病、安全的特点，国民更青睐于将消费投向于中医治疗。

（4）人口老龄化：根据调查显示，我国是老龄化人口最多的国家，老人人口的占比从2013年的7%到2018年增至高达17.5%。中医能够改善老年人体质，老龄化程度的加大使国民更加注重中医养生，因此中医受到国民追捧。

3.2.2 功能性需求

中医诊所管理系统按照功能划分可分为以四下个模块：

（1）就诊管理模块：实现患者预约缴费、查看预约、就诊期间缴费等功能以及实现医生加号病人、线上配制处方、查看病历等功能。

（2）医生工作站模块：实现存储患者病历的功能，提供医生查看病人病历、 选用智能处方模版，也可提供医生查看相似病历，使得医生就诊更加高效化。

（3）客户管理模块：实现对病人就诊后的问候功能，能够建起立良好的医患关系，也能够获体现中医的关怀特色。

（4）运行数据分析模块：实现经营数据分析、医生的业绩分析、门诊情况以及患者情况分析等功能，通过echarts实现数据的可视化。

4 系统设计概要

4.1系统架构及原理

本系统采用的B/S的三层架构体系架构。三层架构包含：表现层，业务逻辑层，数据访问层。三层架构结构非常清晰，其“高内聚，低耦合”的思想降低了各层之间的依赖，开发者只需要关注自己开发的那一层即可，提高了开发效率的同时还降低了维护的成本。[4]

web技术的兴起，进而促进了B/S系统架构的产生和发展。B/S系统架构最大的特点在于不需要下载安装专门的应用软件就可以使用，通过web浏览器就可以访问服务器上的各种资源文件，简易方便。减少了使用成本。其次，B/S系统架构它促进了ajax技术的发展，可以通过使用ajax技术实现页面的局部刷新，更新数据不需要重新加载整个页面，从而降低了服务器的负载。在B/S系统架构之前，实现一个功能需要复杂的专用软件才能进行开发，而B/S系统架构利用web浏览器技术，就可以实现强大的需求功能，这样大大节约了系统开发成本。

B/S系统架构是建立在C/S系统架构基础之上，它是C/S系统架构的一个特例。B/S系统架构相对于B/S系统架构的一个区别在于B/S系统架构采用的是三层架构，包括表现层、业务逻辑层和数据访问层。而C/S系统架构采用的是两层结构，包括表示层和数据访问层。C/S系统架构的表示层集合了所有的业务逻辑，在客户机上请求获得数据库服务器上返回的数据。B/S系统架构采用的三层架构分工更加明确，表现层负责展示界面，展示界面是通过web浏览器实现的；业务逻辑层负责接收表现层发送的请求，并通过访问数据访问层返回对应数据给表示层；数据访问层负责返回业务数据给业务逻辑层。B/S系统架构的各个层分工明确，各其所职，从而减轻了开发者效率，提高了开发效率。B/S和C/S架构图分别如图4.1.1、图4.1.2所示。

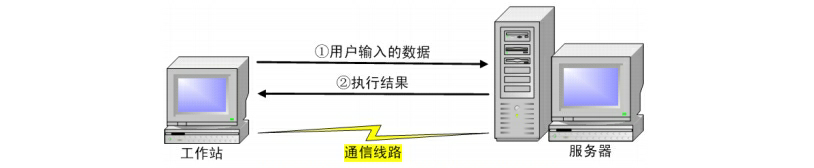


图4.1.1 C/S架构

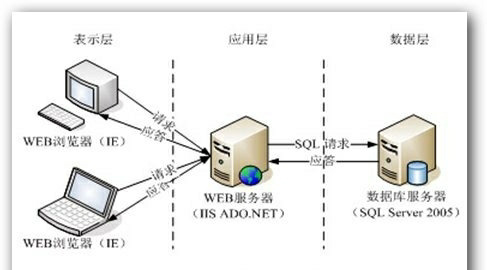


图4.1.2 B/S架构

B/S系统架构的工作原理是采用客户端发送请求服务器响应请求并返回数据的工作模式。B/S系统架构的工作流程大致分为四个过程，首先，由客户端通过ajax技术发送http请求，提交表单；第二，web服务器接收请求，并分析分析客户端请求；第三，服务器访问数据库服务器，将响应的数据返回给客户端；最后，客户端接收到服务器的数据，再通过浏览器解析展示页面。B/S架构工作原理如图4.1.3所示。

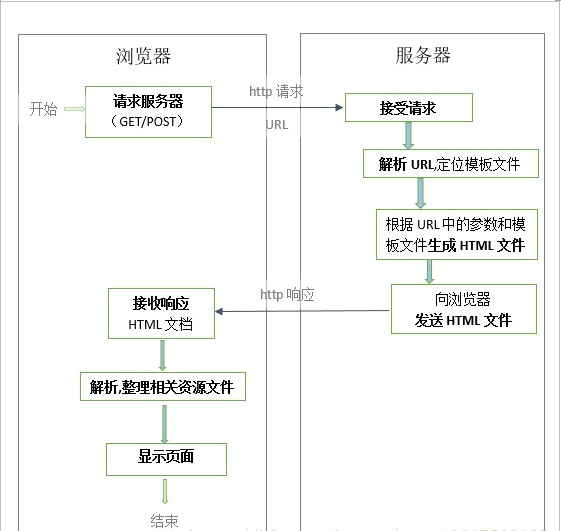


图4.1.3 B/S架构工作原理

4.2 UML建模

本节将通过用例图和顺序图来阐述各个个用例的实现。

4.2.1 就诊模块

中医诊所管理系统的主要业务都包含在了就诊模块。就诊模块的使用者包含了患者和医生。

患者部分：

患者通过登录进入到系统，点击进入预约挂号，根据个人需求选择科室、科室医生和就诊时段，最后通过支付挂号费用完成预约。患者等待叫号就诊，就诊后进入系统进行诊间缴费，诊间费用包括医药费用和中医治疗费用。另外，患者在等待就诊时间内可通过点击查看预约进入查看个人预约信息页面，可以选择取消预约。

医生部分：

医生通过登录进入到系统，在病人管理部分显示了病人的预约列表，医生对预约列表依次加号，在处理每一个患者的挂号时进入到配制处方的页面，医生根据患者实际情况填写主诉、现病史、既往史、体格检查、诊断、处方，在配制处方的时候可选择处方模板。完成处方配制后会生成该患者的一个病历单存储到数据库，同时生成一个患者此次就诊的需要缴纳的费用。在就诊模块，医生还可以进入药方管理，药方管理是提供医生添加和编辑药方模板的，以便于医生在患者就诊时选用药方模板，提高就诊效率。患者和医生就诊用例图分别如图4.2.2.1、图4.2.2.2所示。



图4.2.2.1 患者就诊用例图

图4.2.2.2 医生就诊用例图

4.2.2 医生工作站模块

医生工作站模块实现存储患者病历的功能，医生在就诊完成后提交患者此次就诊数据，生成一个患者病历存储到数据库中。电子病历提供提供医生根据实际情况查看患者过往情况，也可提供医生查看相似病历，使得医生就诊更加高效化，减少了误诊率。

医生工作站模块的用例图如图4.2.2.2所示。

4.2.3 客户管理模块

在就诊完成后系统生成该医生的病人的详细信息，医生根据患者提供的联系信息在一定的时间间隔问候患者的身体健康状况，这样能够建起立良好的医患关系，也能够获体现中医关怀特色。

客户管理模块的用例图如图4.2.3.1所示。



图4.2.3.1 客户管理模块的用例图

4.2.4 运行数据分析模块

运行数据分析模块的使用者为中医门诊的管理者，系统采用echarts来展示门诊的各个数据的可视化，使得数据数据更加直观。门诊运行数据包括经营、医生的业绩、患者情况等数据。运行数据分析有利于门诊管理者制定符合门诊发展的正确决策。

运行数据分析模块的用例图如图4.2.4.1所示。



图4.2.4.1 运行数据分析模块的用例图

4.3数据库设计

该系统采用的是MongoDB数据库，有关MongoDB的详细介绍在第二章节。系统共有13个集合，如表4-3-1所示。

表4-3-1 数据库集合描述表

|  |  |  |
| --- | --- | --- |
| 序号 | 集合名称 | 描述 |
| 01 | users | 用户信息集合 |
| 02 | userDetail | 患者详细信息集合 |
| 03 | appointmentRecord | 预约挂号记录集合 |
| 04 | payRecord | 缴费记录集合 |
| 05 | medicalPay | 诊间缴费集合 |
| 06 | depOfDoctor | 科室信息集合 |
| 07 | doctorDrug | 医生药方模板集合 |
| 08 | doctorDetail | 医生详细信息集合 |
| 09 | medicalRecord | 病历集合 |

接下来的章节是系统主要集合进行详细描述。

4.3.1患者详细信息集合

用户信息集合主要记录患者的姓名、年龄、出生日期、性别、联系电话、居住地址，

患者详细信息集合文档描述如表4.3.1.2所示。

该集合的文档设计：

{

                    "\_id" : ObjectId("5eaae2308e06674d547ac1d2"),

                    "name" : "user2",

                    "age" : 22,

                    "birthData" : "1996-04-15T16:00:00.000Z",

                    "sex" : "男",

                    "telNumber" : NumberLong("18477296770"),

                    "region" : "广西壮族自治区柳州市鹿寨县",

                 }

表4.3.1.2 患者详细信息集合文档描述

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 描述 |
| name | String | 患者姓名 |
| age | Int32 | 患者年龄 |
| birthData | String | 患者出生日期 |
| sex | String | 患者性别 |
| telNumber | String | 患者联系电话 |
| region | String | 患者居住地址 |

4.3.2预约挂号记录集合

预约挂号记录息集合主要记录预约的患者姓名、预约科室、预约医生、就诊时间、预约挂号支付、预约创建时间、预约状态，集合文档描述如表4.3.2.1所示。

该集合的文档设计：

  {

                  "\_id" : ObjectId("5eaadc3d8e06674d547ac1ce"),

                  "username" : "user2",

                  "depName" : "内科",

                  "doctorName" : "内科医生1",

                  "treatTime" : "2020-5-2 9:00~11:20",

                  "pay" : 80,

                  "createTime" : "2020/4/30 下午10:10:04",

                  "overFlag" : true

                }

表4.3.2.1 预约挂号集合文档描述

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 描述 |
| username | String | 患者姓名 |
| depName | String | 预约科室 |
| doctorName | String | 预约医生 |
| treatTime | String | 就诊时间 |
| pay | Int32 | 预约挂号支付 |
| createTime | String | 预约创建时间 |
| overFlag | Boolean | 预约状态 |

预约状态：true代表预约过期已经无效，false表示预约是有效的

4.3.3 缴费记录集合

缴费记录集合主要记录患者费用的详细信息，包括姓名、缴费时间、支付金额、支付方式、支付描述，科室名称，集合文档描述如表4.3.3.1所示。

该集合的文档设计：

 {

      "\_id" : ObjectId("5eba3bfa3adbb7579493a64f"),

      "username" : "user2",

      "createTime" : "2020/5/12 下午2:02:08",

      "pay" : 210,

      "payWay" : "微信支付",

      "payDes" : "诊间缴费",

      "depName" : "内科"

    }

表4.3.3.1 缴费记录集合文档描述

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 描述 |
| username | String | 患者姓名 |
| createTime | String | 缴费时间 |
| pay | Int32 | 支付金额 |
| payWay | String | 支付方式 |
| payDes | String | 支付描述 |
| depName | String | 科室名称 |

4.3.4 诊间缴费集合

诊间缴费集合主要记录患者在就诊后的费用信息，包括患者姓名、费用创建时间、中医治疗费用、药物费用，集合文档描述如表4.3.4.1所示。

该集合的文档设计：

  {

                "\_id" : ObjectId("5eb4d27a5e2bf31cecdb52da"),

                "payUsername" : "user2",

                "createTime" : "2020/5/8 上午11:31:06",

                "TCMTreatmentPay" : 10,

                "drugPay" : 200

              }

表4.3.4.1 诊间缴费集合文档描述

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 描述 |
| payUsername | String | 患者姓名 |
| createTime | String | 费用创建时间 |
| TCMTreatmentPay | Int32 | 中医治疗费用 |
| drugPay | Int32 | 药物费用 |

4.3.5 医生详细信息集合

医生详细信息集合主要记录医生的姓名、费用创建时间、中医治疗费用、药物费用，

集合文档描述如表4.3.5.1所示。

该集合的文档设计：

 {

                "\_id" : ObjectId("5eaa6cfb38a383e371ab1faa"),

                "name" : "外科医生1",

                "pay" : 80,

                "grade" : "专家"

              }

表4.3.5.1 医生详细信息集合文档描述

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 描述 |
| name | String | 患者姓名 |
| pay | Int32 | 费用创建时间 |
| grade | String | 中医治疗费用 |

4.3.6 科室信息集合

科室信息主要记录科室的名称，以及科室所包含的医生。医生的详细信息描述在4.3.5章节，集合文档描述如表4.3.6.1所示。

该集合的文档设计：

{

                "\_id" : ObjectId("5eaa653338a383e371ab1fa8"),

                "name" : "外科",

                "doctors" : [{

                    "name" : "外科医生1",

                    "pay" : 80,

                    "grade" : "专家"

                  }, {

                    "name" : "外科医生2",

                    "pay" : 60,

                    "grade" : "主治医生"

                  }, {

                    "name" : "外科医生3",

                    "pay" : 40,

                    "grade" : "普通医生"

                  }, {

                    "name" : "外科医生4",

                    "pay" : 40,

                    "grade" : "普通医生"

                  }]

              }

表4.3.6.1 科室信息集合文档描述

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 描述 |
| name | String | 科室名称 |
| doctors | Object | 科室所包含的医生 |

4.3.7 医生药方模板集合

医生药方模板集合主要记录医生创建的药方模板的名称、药方模板的ID、药物成分、药物用法、创建该药方的医生姓名，集合文档描述表4.3.7.1所示。

该集合的文档设计：

 {

                "\_id" : ObjectId("5eb4e33d5e2bf31cecdb52db"),

                "dragname" : "风热感冒",

                "ingredient" : "桔梗12克，土牛膝30克，浙贝母12克",

                "usage" : "每日一剂，共三剂",

                "drugId" : "k9xpr0ba",

                "username" : "内科医生1"

              }

表4.3.7.1 医生药方模板集合文档描述

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 描述 |
| dragname | String | 药方模板的名称 |
| ingredient | String | 药物成分 |
| usage | String | 药物用法 |
| drugId | String | 药方模板的ID |
| username | String | 创建该药方的医生姓名 |

4.3.8 病历集合

病历集合主要记录患者信息、诊断信息、病历创建时间，集合文档描述如表4.3.8.1所示。

该集合的文档设计：

      {

"\_id" : ObjectId("5eb4d27a5e2bf31cecdb52d9"),

              "patientMsg" : {

                "\_id" : "5eaae2308e06674d547ac1d2",

                "name" : "user2",

                "age" : 22,

                "birthData" : "1996-04-15T16:00:00.000Z",

                "sex" : "男",

                "telNumber" : NumberLong("18477296770"),

                "region" : "广西壮族自治区柳州市鹿寨县",

                "username" : "user2",

                "role" : "0"

              },

              "doctorDiagnosis" : {

                "mainSuit" : "咽痛四天，流涕，喷嚏。",

                "currentIllHis" : "仍咳嗽",

                "oldIllHis" : "无特殊",

                "bodyCheck" : "咳清充血、双侧扁体未见分泌物、苔薄白脉滑",

                "diagnosis" : "感冒（风热）",

                "drugs" : {

                  "ingredient" : "桔梗12克，土牛膝30克",

                  "usage" : "每日一剂，共三剂"

                },

                "doctorName" : "内科医生1",

                "depName" : "内科"

              },

              "createTime" : "2020/5/8 上午11:31:06",

           }

表4.3.8.1 病历集合文档描述

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 描述 |
| patientMsg | Object | 患者信息 |
| doctorDiagnosis | Object | 诊断信息 |
| createTime | String | 病历创建时间 |

5 项目详细设计与实现

5.1 就诊模块设计与实现

中医诊所管理系统的主要业务都包含在了就诊模块。就诊模块实现患者预约缴费、查看预约、就诊期间缴费等功能以及实现医生加号病人、线上配制处方、查看病历等功能。

5.1.1 就诊模块设计

就诊模块流程图与界面设计图如下列图片所示。

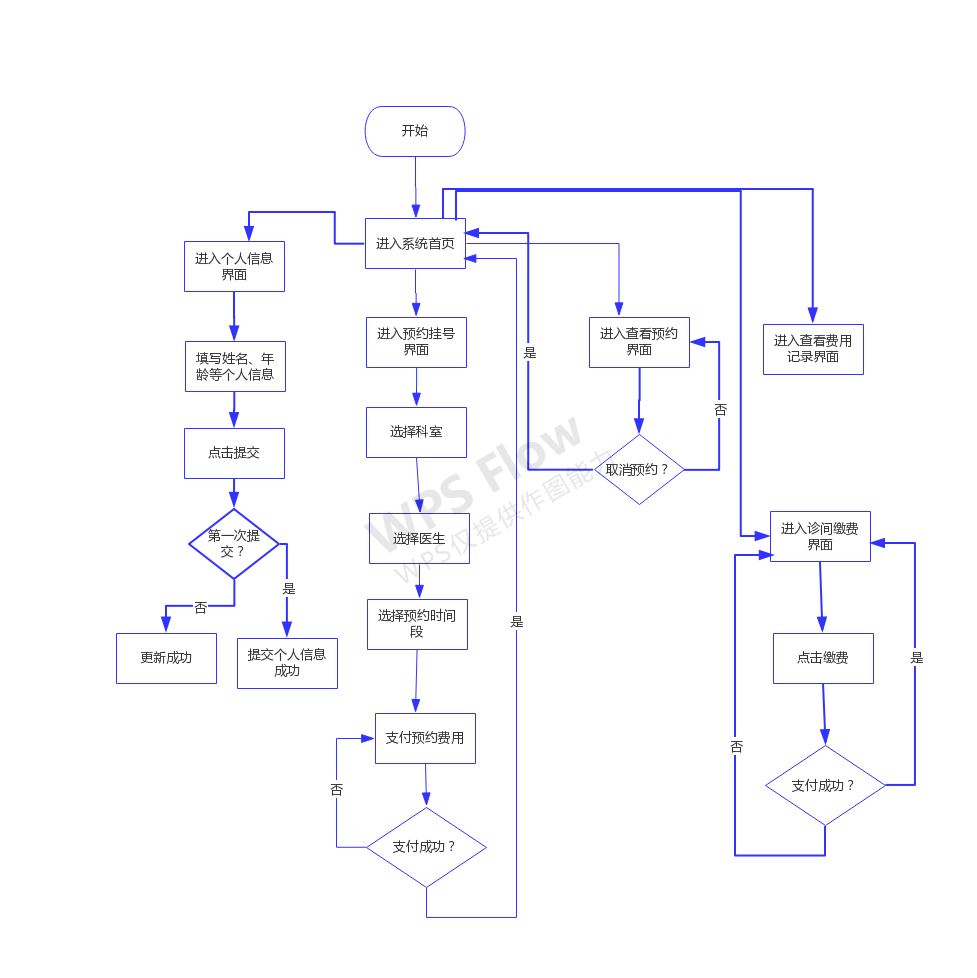


图5.1.1.1 就诊模块（患者）流程图



图5.1.1.2 就诊模块（患者）首页设计



图5.1.1.3 就诊模块（患者）预约挂号选择科室界面设计



图5.1.1.4 就诊模块（患者）预约挂号选择医生界面设计



图5.1.1.5 就诊模块（患者）预约挂号选择就诊时间界面设计



图5.1.1.6 就诊模块（患者）查看预约界面设计



图5.1.1.7 就诊模块（患者）查看诊间缴费界面设计



图5.1.1.8 就诊模块（患者）查看缴费记录界面设计



图5.1.1.9 就诊模块（患者）编辑个人信息界面设计

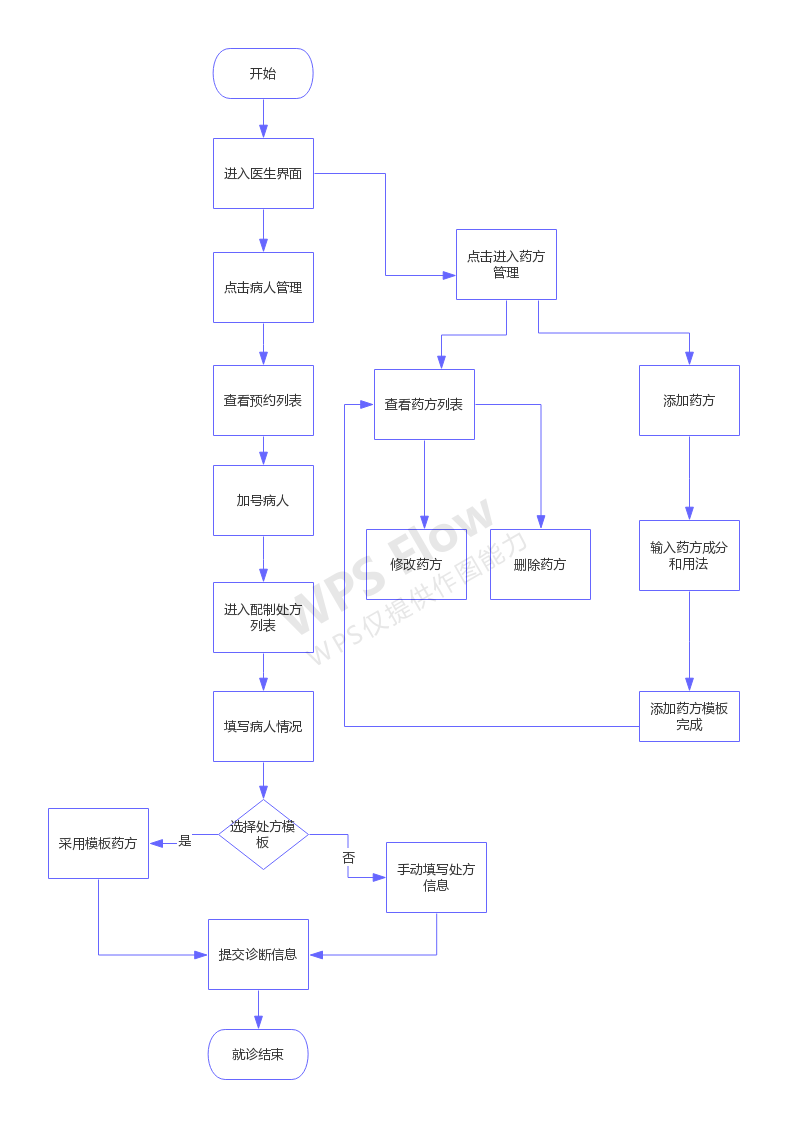


图5.1.1.10 就诊模块（医生）流程图

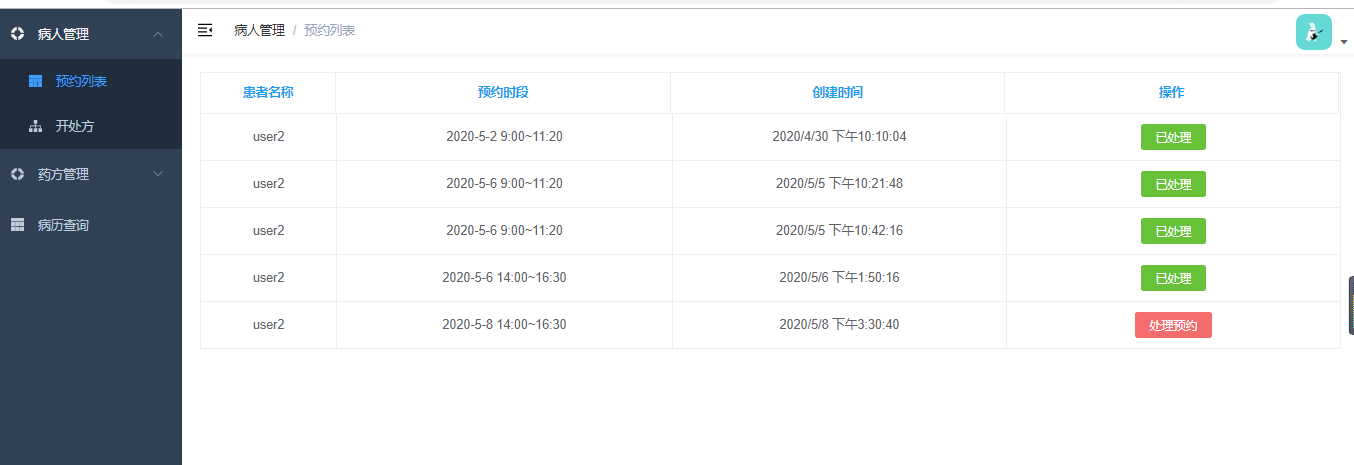


图5.1.1.11 就诊模块（医生）查看病人列表界面设计



图5.1.1.12 就诊模块（医生）查看诊断界面设计



图5.1.1.13 就诊模块（医生）选用药方模板界面设计

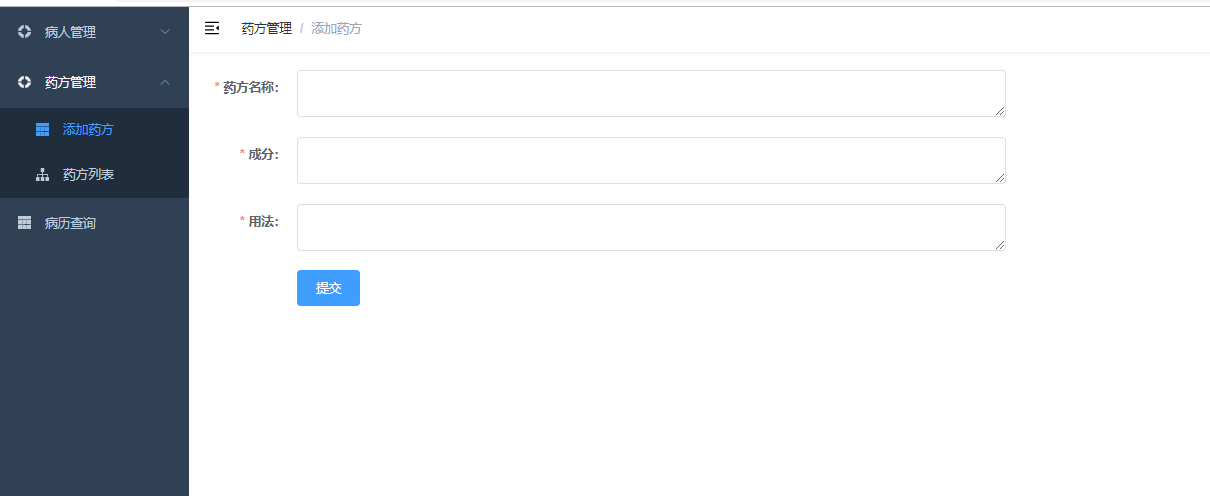


图5.1.1.14 就诊模块（医生）添加药方界面设计

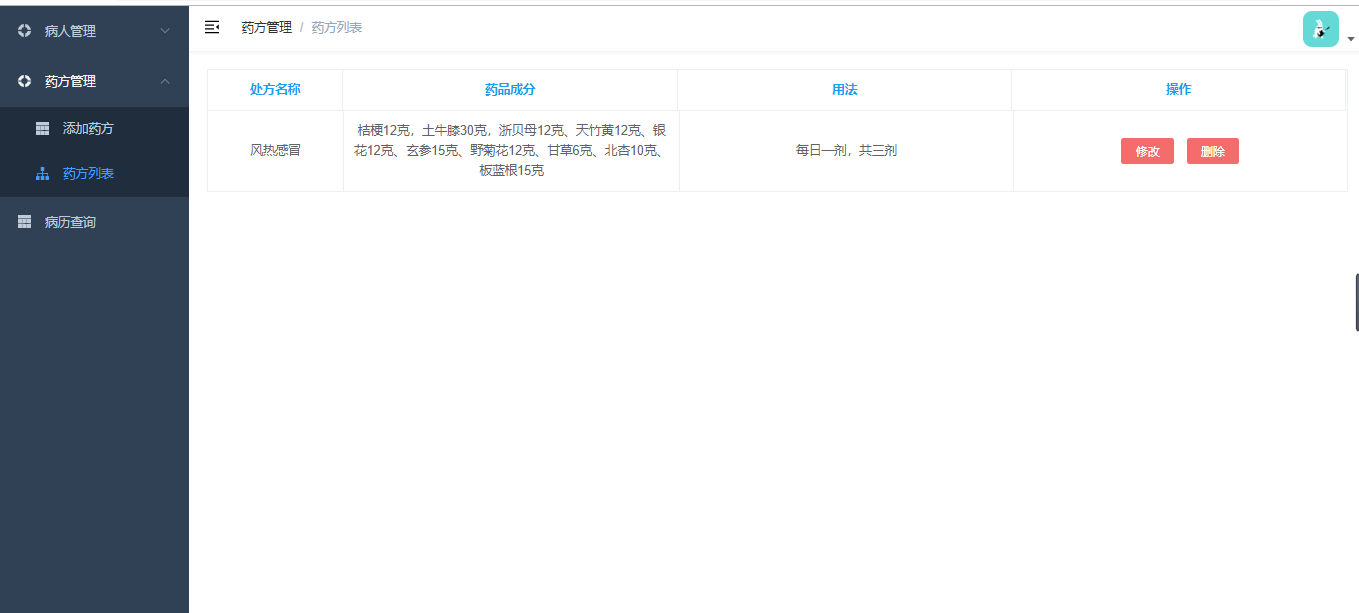


图5.1.1.15 就诊模块（医生）药方列表界面设计



图5.1.1.16 就诊模块（医生）添加药方模板界面设计

5.1.2 就诊模块代码实现

就诊模块（患者）提交预约信息代码：

// 提交预约信息

    submitMsg() {

      if (!this.payWay) {

        this.showAlert = true;

        return false;

      } else {

        let submitMsg = {

          username: getUserName(),

          depName: this.$store.getters.depName,

          doctorName: this.$store.getters.doctorName,

          treatTime: this.$store.getters.treatTime,

          pay: this.$store.getters.doctorPay,

          payWay: this.payWay,

          createTime: new Date().toLocaleString(),

          payDes: "预约",

          overFlag: false

        };

        this.$store

          .dispatch("patient/makeAnAppointment", submitMsg)

          .then(data => {

            if (data.resCode == 1) {

              let payData = {

                username: getUserName(),

                createTime: new Date().toLocaleString(),

                pay: this.$store.getters.doctorPay,

                payWay: submitMsg.payWay,

                payDes: "预约"

              };

              postPayRecord(payData)

                .then(response => {

                  if (response.data.resCode == 1) {

                    Message({

                      message: "支付成功,预约成功",

                      type: "success",

                      duration: 3 \* 1000

                    });

                  }

                  this.$router.push({ path: "/patient/home" });

                })

                .catch(error => {

                  console.log("postPayRecord", error);

                  reject(error);

                });

            }

          });

        this.cancelSubmit();

      }

    },

就诊模块（患者）查看预约记录代码：

//   获取预约列表

    getAppointRecordList() {

      let username = getUserName();

      this.$store

        .dispatch("patient/getAppointRecordList", username)

        .then(data => {

          if (data.resCode == 1) {

            this.appointRecordList = data.appointRecordList.filter(

              (item, index) => {

                return item.overFlag == false;

              }

            );

          }

        });

    },

就诊模块（患者）诊间缴费代码：

 //   获取诊间缴费列表

    getMedicinePayList() {

      let username = getUserName();

      this.$store

        .dispatch("patient/getMedicinePayList", username)

        .then(data => {

          if (data.resCode == 1) {

            this.medicinePayList = data.medicinePayList;

          }

        });

    },

// 诊间缴费，提交缴费信息

    postPayRecord() {

      if (!this.payWay) {

        this.showAlert = true;

        return false;

      } else {

        let that = this;

        let postData = {

          username: getUserName(),

          createTime: getCurrentTime(),

          pay: this.rowData.allPay,

          payWay: this.payWay,

          payDes: "诊间缴费"

        };

        postPayRecord(postData)

          .then(response => {

            Message({

              message: "支付成功",

              type: "success",

              duration: 3 \* 1000

            });

            that.cancelSubmit();

            deleMedicalPay(that.row)

              .then(response => {

                that.getMedicinePayList();

              })

              .catch(error => {

                console.log("deleMedicalPay", error);

                reject(error);

              });

          })

          .catch(error => {

            console.log("postPayRecord", error);

            reject(error);

          });

      }

    },

就诊模块（患者）查看费用记录代码：

  //   获取费用记录列表

    getPayRecordList() {

      let username = getUserName();

      this.$store.dispatch("patient/getPayRecordList", username)

.then(data => {

        if (data.resCode == 1) {

          this.payRecordList = data.payRecordList;

        }

      });

    }

就诊模块（患者）提交个人信息代码：

 // 提交个人信息

    submitHandle(flag) {

      let otherData = {

        username: getUserName(),

        role: getRole(),

        newSubmit: flag

      };

      let postData = Object.assign(this.ruleForm, otherData);

      postUserDetail(postData)

        .then(res => {

          this.getUserDetail();

          if (res.data.resCode) {

            Message({

              message: res.data.des,

              type: "success",

              duration: 3 \* 1000

            });

          } else {

            Message({

              message: res.data.des,

              type: "error",

              duration: 3 \* 1000

            });

          }

        })

        .catch(error => {

          console.log("userDetail", error);

          reject(error);

        });

    },

就诊模块（医生）提交诊断信息代码：

// 医生提交诊断信息

    submitHandle() {

      let medicalData = {

        patientMsg: this.patientDetail,

        doctorDiagnosis: this.ruleForm,

        createTime: new Date().toLocaleString(),

        patientName: this.patientName

      };

      postMedicalRecord(medicalData)

        .then(res => {

          if (res.data.resCode === 1) {

            // console.log("提交病历成功")

            makeAnAppointment({

              patientName: this.patientName,

              doctorName: getUserName(),

              updateFlag: true

            })

              .then(res => {

                if (res.data.resCode === 1) {

                  // console.log("修改预约成功")

                  postMedicalPay({

                    payUsername: this.patientName,

                    createTime: new Date().toLocaleString(),

                    TCMTreatmentPay: 10,

                    drugPay: 200

                  })

                    .then(res => {

                      if (res.data.resCode === 1) {

                        // console.log("提交诊间缴费成功")

                        this.$router.push(

{ path: "/doctor/appointmentList" });

                      }

                    })

                    .catch(error => {

                      console.log("postMedicalPay", error);

                      reject(error);

                    });

                }

              })

              .catch(error => {

                console.log("makeAnAppointment", error);

                reject(error);

              });

          }

        })

        .catch(error => {

          console.log("postMedicalRecord", error);

          reject(error);

        });

    }

就诊模块（医生）查看药方模板信息代码：

// 获取药方模板

    getDragList() {

      getDragList({ username: getUserName() })

        .then(res => {

          this.dragList = res.data;

        })

        .catch(error => {

          console.log("getDragList", error);

          reject(error);

        });

    },

就诊模块（医生）查看修改和删除药方模板信息代码：

 // 修改药方模板

    modifiedSure() {

      uptateDrug(this.form)

        .then(res => {

          if (res.data.resCode == 1) {

            Message({

              message: res.data.des,

              type: "success",

              duration: 3 \* 1000

            });

            this.drugDialogVisible = false;

            this.getDragList();

          }

        })

        .catch(error => {

          console.log("uptateDrug", error);

          reject(error);

        });

    },

 // 删除药方模板

    deleteDrug(data) {

      deleteDrug(data)

        .then(res => {

          if (res.data.resCode == 1) {

            Message({

              message: res.data.des,

              type: "success",

              duration: 3 \* 1000

            });

            this.getDragList();

          }

        })

        .catch(error => {

          console.log("deleteDrug", error);

          reject(error);

        });

    },

5.2 医生工作站模块设计与实现

医生工作站模块实现存储患者病历的功能，提供医生查看病人病历和查看相似病历，使得医生就诊更加高效化。

5.2.1 医生工作站模块设计

医生工作站模块流程图与界面设计图如下列图片所示。

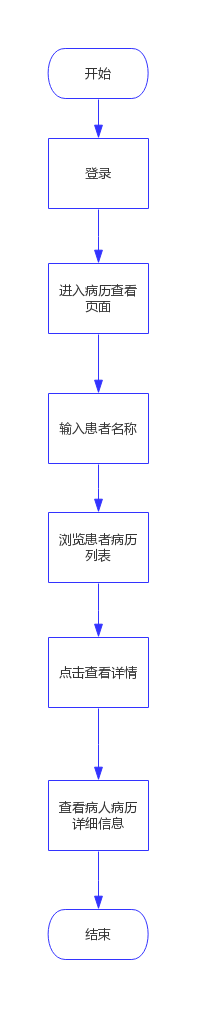


图5.2.1.1 医生工作站查看病历流程图



图5.2.1.2 医生工作站查看患者病历列表界面设计



图5.2.1.3 医生工作站查看患者病历详情界面设计

5.2.2 医生工作站模块代码实现

医生工作站模块（患者）获取患者病历代码：

 // 获取患者病历信息

    getMedicalRecordList() {

      getMedicalRecordList({ patientName: this.patientNameSearch })

        .then(res => {

          this.medicalRecordList = res.data.medicalRecordList;

          this.loading = false;

        })

        .catch(error => {

          console.log("getMedicalRecordList", error);

          reject(error);

        });

    },

5.3 客户管理模块设计与实现

客户管理实现对病人就诊后的问候功能，能够建起立良好的医患关系，也能够获体现中医的关怀特色。

5.3.1 客户管理模块设计

客户管理模块流程图与界面设计图如下列图片所示。

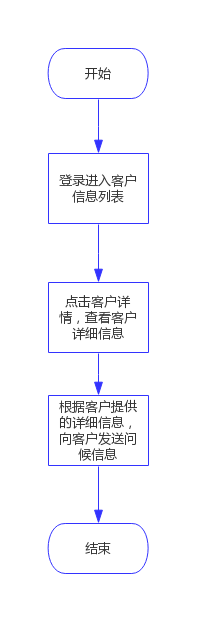


图5.3.1.1 客户管理模块流程图

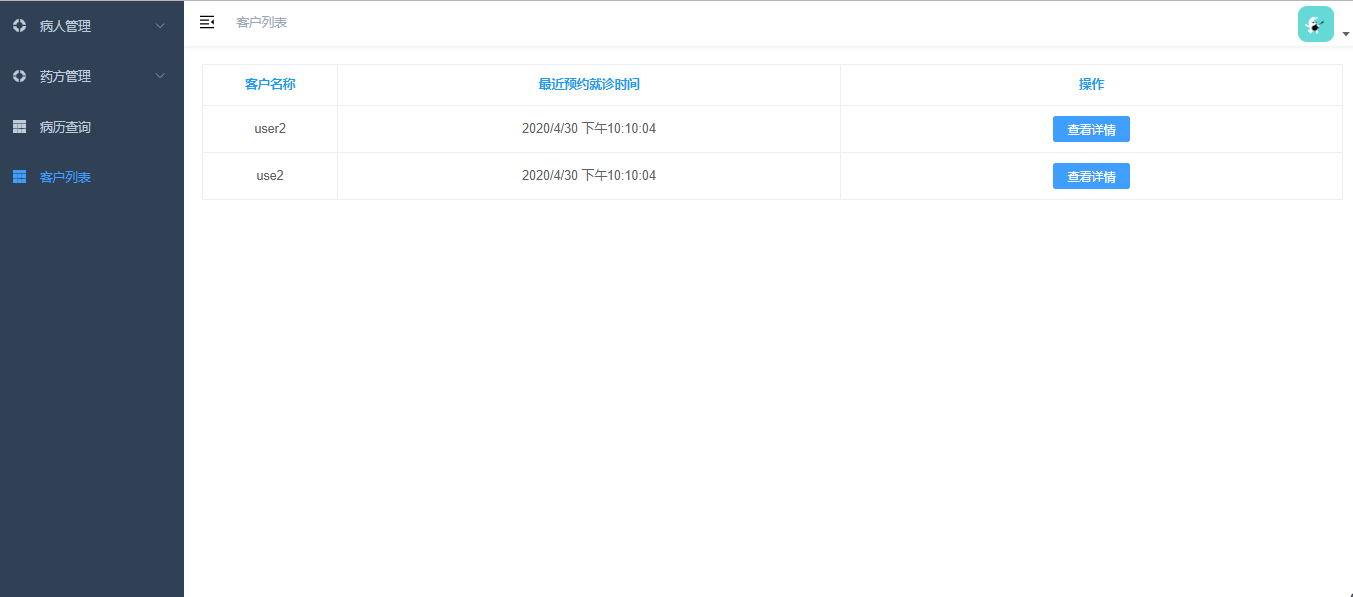


图5.3.1.2 客户管理查看客户信息界面设计

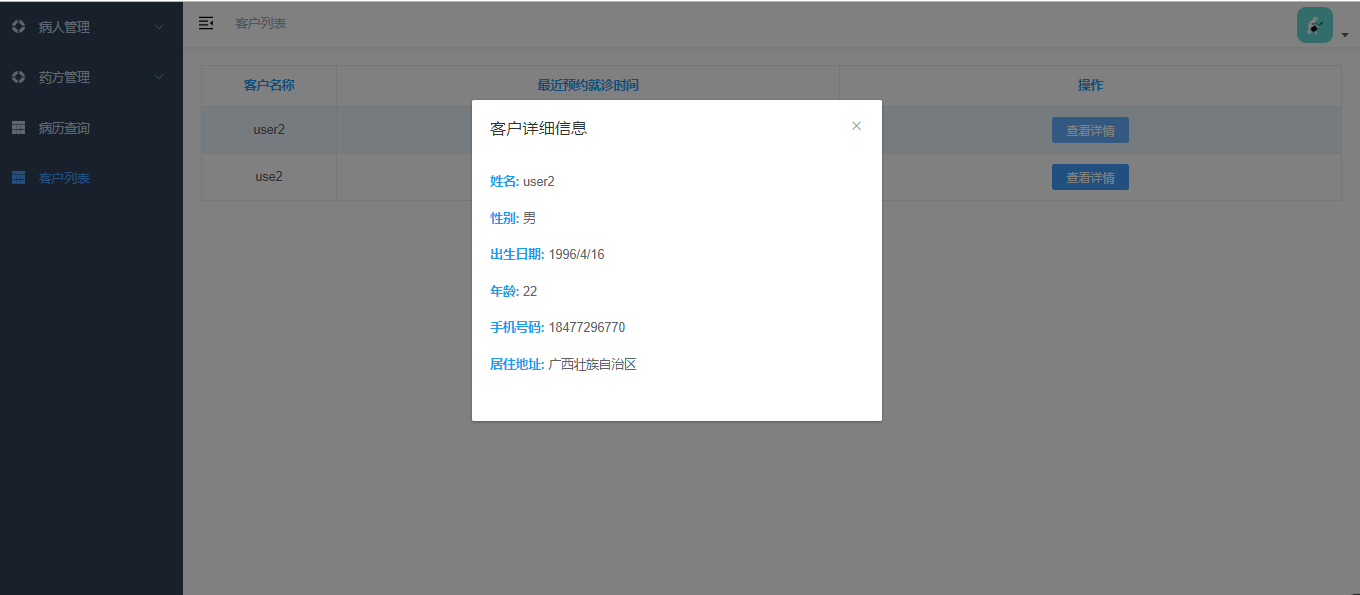


图5.3.1.3 客户管理查看客户详情信息界面设计

5.3.2 客户管理模块代码实现

客户管理模块查看客户信息代码：

// 获取客户信息列表

    getCustomerList() {

      getAppointRecordList({ doctorName: getUserName() })

        .then(res => {

          this.customerList = res.data.appointRecordList.filter(

              (item, index) => {

                return item.overFlag == true;

              }

            );

          this.customerList = this.deDuplication(this.customerList)

          this.loading = false;

        })

        .catch(error => {

          console.log("getAppointRecordList", error);

          reject(error);

        });

    },

// 获取客户详情信息

     getUserDetail(data) {

      let postData = {

        username: data.username,

        role: "0"

      };

      getUserDetail(postData)

        .then(res => {

          let userDetail = res.data.userDetail[0];

          this.userDetail = userDetail

        })

        .catch(error => {

          console.log("getUserDetail", error);

          reject(error);

        });

    },

5.4 运营数据分析模块设计与实现

运行数据分析实现经营数据分析、医生的业绩分析以及患者情况分析等功能，通过echarts实现数据的可视化。

5.4.1 运营数据分析模块设计

运营数据分析模块流程图与界面设计图如下列图片所示。

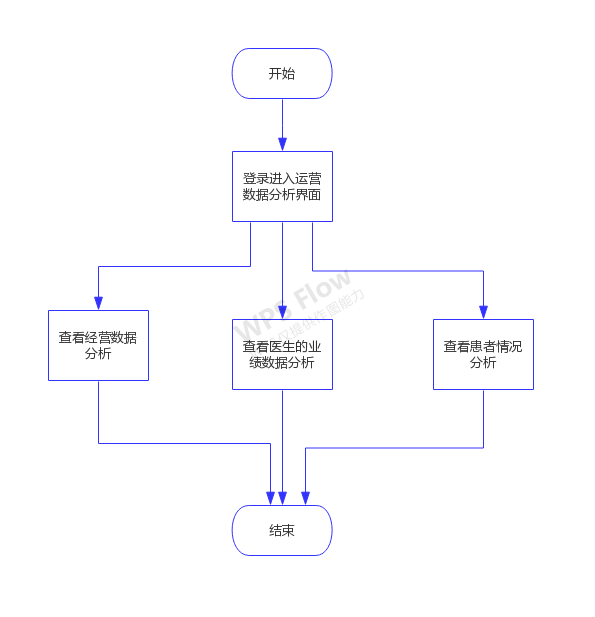


图5.4.1.1 运营数据分析模块流程图

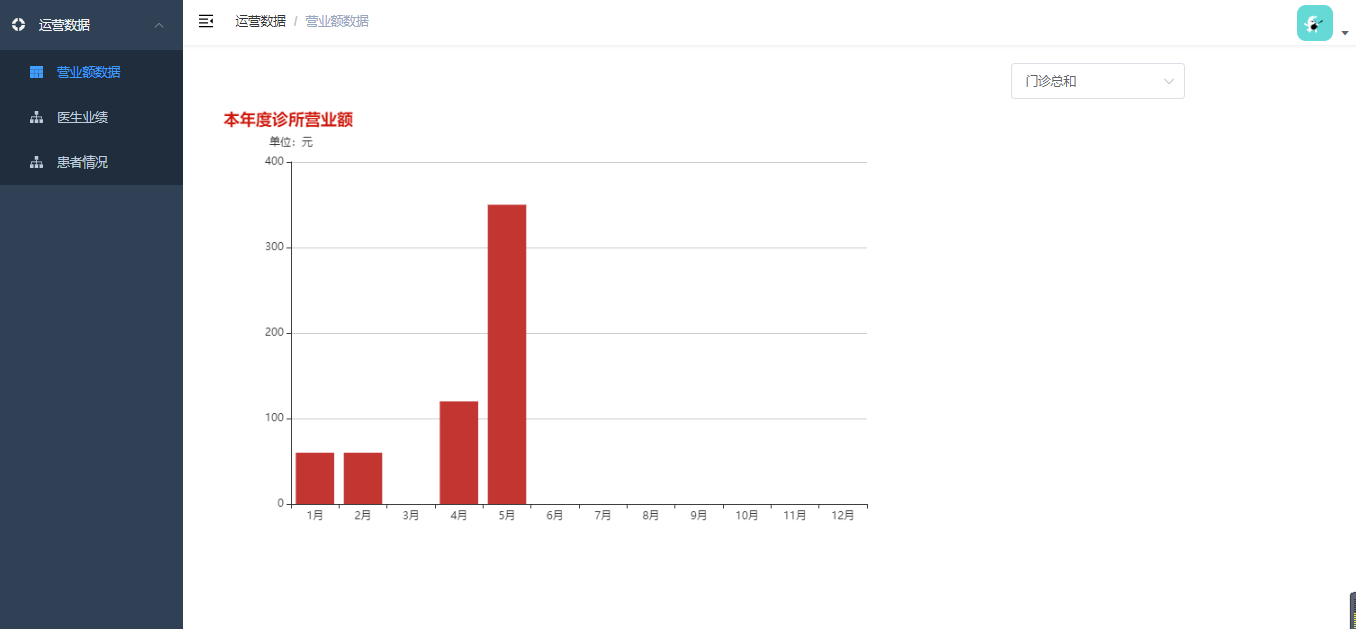


图5.4.1.2 运营数据分析模块经营数据界面设计

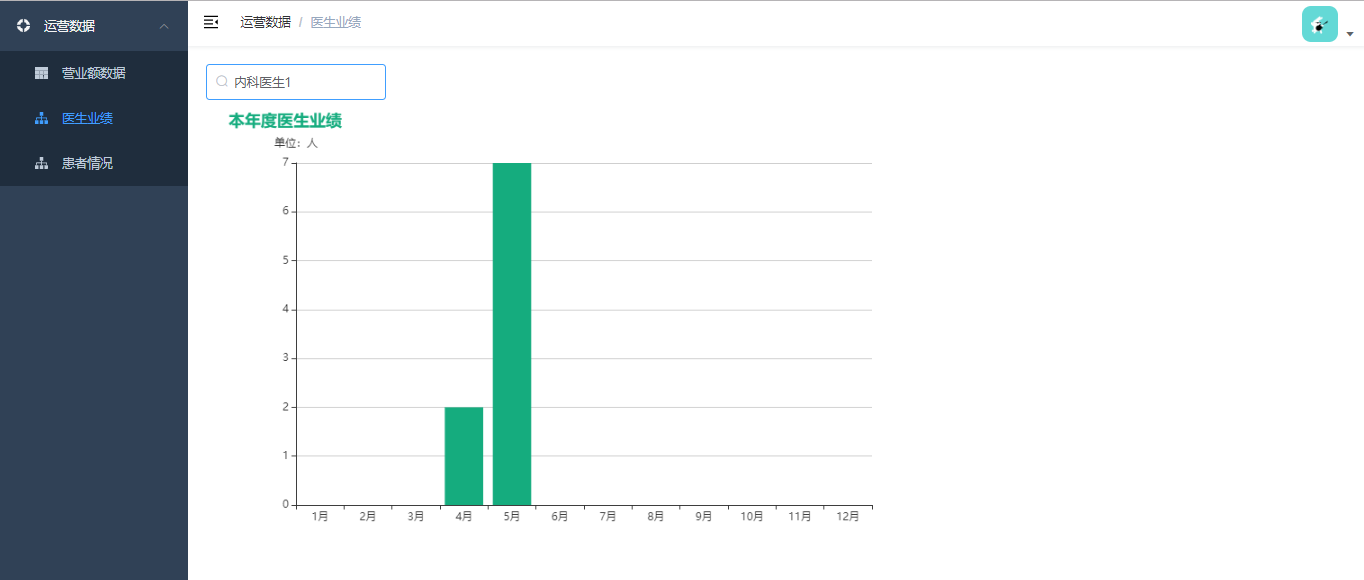


图5.4.1.3 运营数据分析模块医生业绩分析界面设计

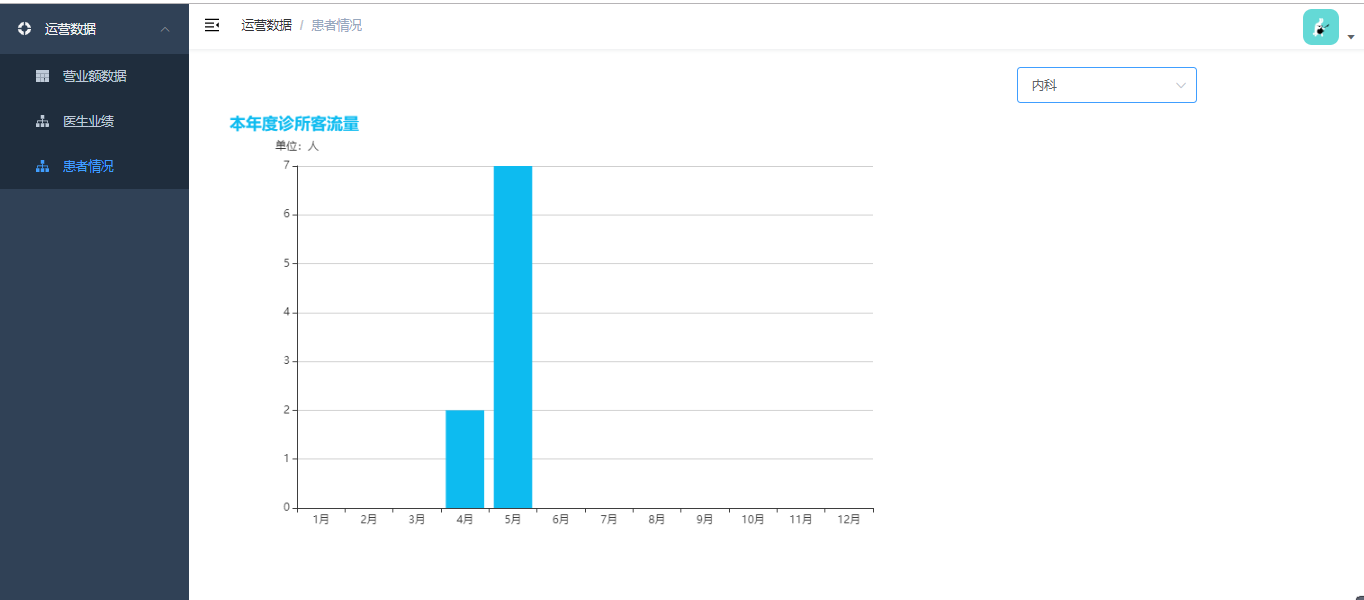


图5.4.1.4 运营数据分析模块患者情况界面设计

5.4.2 运营数据分析模块代码实现

运营数据分析模块查看经营数据代码：

//   获取经营数据

    getPayRecordList() {

      getPayRecordList()

        .then(res => {

          if (res.data.resCode == 1) {

            this.payRecordList = res.data.payRecordList;

            this.initChart();

          }

        })

        .catch(error => {

          console.log("getPayRecordList", error);

          reject(error);

        });

    },

     // 初始化图表

    initChart() {

      let chart = echarts.getInstanceByDom(

document.querySelector("#chart"));

      if (chart === undefined) {

        chart = echarts.init(document.querySelector("#chart"));

      }

      this.chart = chart;

      let seriesData = getChartData(this.payRecordList)

      chart.setOption(getOption(seriesData));

    },

运营数据分析模块查看医生业绩代码：

//   获取医生业绩数据

    getDoctorPerformanceList() {

      getAppointRecordList()

        .then(res => {

          if (res.data.resCode == 1) {

            this.appointRecordList = res.data.appointRecordList;

            this.initChart();

          }

        })

        .catch(error => {

          console.log("getPayRecordList", error);

          reject(error);

        });

    },

    // 初始化图表

    initChart() {

      let chart = echarts.getInstanceByDom(

document.querySelector("#chart"));

      if (chart === undefined) {

        chart = echarts.init(document.querySelector("#chart"));

      }

      this.chart = chart;

      let seriesData = []

      chart.setOption(getOption(seriesData));

    },

运营数据分析模块查看患者情况代码：

 //   获取患者情况数据

    getPatientSituationList() {

      getAppointRecordList()

        .then(res => {

          if (res.data.resCode == 1) {

            this.appointRecordList = res.data.appointRecordList;

            // console.log(this.appointRecordList)

            this.initChart();

          }

        })

        .catch(error => {

          console.log("getPayRecordList", error);

          reject(error);

        });

    },

     // 初始化图表

    initChart() {

      let chart = echarts.getInstanceByDom(

document.querySelector("#chart"));

      if (chart === undefined) {

        chart = echarts.init(document.querySelector("#chart"));

      }

      this.chart = chart;

      let seriesData = getChartData(this.appointRecordList)

      chart.setOption(getOption(seriesData));

    },

6 测试

6.1 就诊模块测试

6.1.1 预约挂号功能测试

表6.1.1 预约挂号功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-1-1 | 功能特性 | 测试用户系统预约功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以患者身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | 1. 选择科室、医生、就诊时间 2. 点击支付该预约费用 3. 点击确定 | | 科室、医生、就诊时间 | 预约成功 | 预约成功 |
| 02 | （1）点击查看预约 | | 通过cookie获取用户名 | 成功查看患者预约信息 | 成功查看患者预约信息 |
| 03 | 1. 选中需要取消的预约 2. 点击取消预约 | | 要删除的预约信息 | 取消预约成功 | 取消预约成功 |
| 04 | 点击退出 | | 无 | 退出预约挂号页面 | 退出成功 |

6.1.2 费用记录功能测试

表6.1.2 费用记录功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-1-2 | 功能特性 | 测试查看费用记录功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以患者身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | （1）点击费用记录 | | 通过cookie获取用户名 | 查看费用列表成功 | 查看费用列表成功 |
| 02 | 点击退出 | | 无 | 退出费用记录页面 | 退出成功 |

6.1.3 诊间缴费功能测试

表6.1.3 诊间缴费功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-1-3 | 功能特性 | 测试诊间缴费功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以患者身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | 1. 点击诊间缴费 2. 点击查看诊间费用 | | 通过cookie获取用户名 | 查看诊间费用信息成功 | 查看诊间费用信息成功 |
| 02 | （1）选中要查看的费用  （2）点击查看费用明细 | | 选中要查看的诊间费用 | 查看诊间费用详细信息成功 | 查看诊间费用详细信息成功 |
| 03 | 1. 选中要查看的费用 2. 选择支付方式 3. 点击缴费 | | 选中要查看的费用以及支付方式 | 诊间缴费成功 | 诊间缴费成功 |
| 04 | 点击退出 | | 无 | 退出诊间缴费页面 | 退出成功 |

6.1.4 患者个人信息功能测试

表6.1.4 患者个人信息功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-1-4 | 功能特性 | 测试个人信息功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以患者身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | （1）点击个人信息  （2）输入个人信息   1. 点击提交 | | 输入个人信息 | 添加个人信息成功 | 添加个人信息成功 |
| 02 | （1）点击个人信息 | | 通过cookie获取用户名 | 查看个人信息成功 | 查看个人信息成功 |
| 03 | （1）点击个人信息  （2）输入要修改的信息  （3）点击修改 | | 输入要修改的信息 | 修改个人信息成功 | 修改个人信息成功 |
| 04 | 点击退出 | | 无 | 退出个人信息页面 | 退出成功 |

6.1.5 医生加号病人功能测试

表6.1.5 医生加号病人功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-1-5 | 功能特性 | 测试医生加号病人功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以医生身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | （1）点击病人管理  （2）点击预约列表 | | 通过cookie获取医生名 | 查看预约患者列表成功 | 查看预约患者列表成功 |
| 02 | 1. 选中预约患者 2. 进入患者诊断页面 | | 选中的预约患者 | 加号病人成功 | 加号病人成功 |

6.1.6 医生对患者进行诊断功能测试

表6.1.6 医生对患者进行诊断功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-1-6 | 功能特性 | 测试医生对患者进行诊断功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以医生身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | （1）输入诊断信息  （2）点击提交 | | 输入诊断信息 | 诊断信息提交成功 | 诊断信息提交成功 |
| 02 | 1. 点击选择药方模板 2. 浏览模板信息 3. 选中模板 | | 通过cookie获取用户名 | 选择模板药方成功 | 选择模板药方成功 |
| 03 | （1）点击提交诊断信息  （2）弹出信息 | | 诊断信息 | 弹出生成患者诊间费用成功信息 | 弹出生成患者诊间费用成功信息 |
| 04 | 点击退出 | | 无 | 退出患者诊断页面 | 退出成功 |

6.1.7 医生药方管理功能测试

表6.1.7 医生药方管理功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-1-7 | 功能特性 | 测试医生药方管理功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以医生身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | （1）选择药方管理  （2）点击添加药方  （3）输入药方成分、用法  （4）点击提交 | | 药方成分、用法 | 添加药方成功 | 添加药方成功 |
| 02 | 1. 选择药方管理 2. 选择药方列表 3. 选择要删除的药方 4. 点击删除 | | 要删除的药方 | 删除药方成功 | 删除药方成功 |
| 03 | （1）进入药方管理  （2）选择药方列表  （3）选择要修改的药方  （4）点击修改  （5）输入修改信息  （6）点击确定 | | 要修改的药方 | 修改药方成功 | 修改药方成功 |
| 04 | 点击退出 | | 无 | 退出药方管理页面 | 退出成功 |

6.2 医生工作站模块测试

6.2.1 查看病历功能测试

表6.2.1 查看病历功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-2-1 | 功能特性 | 测试查看病历功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以医生身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | （1）点击病历查询  （2）输入患者名称  （3）点击回车 | | 患者名称 | 查看病历成功 | 查看病历成功 |
| 02 | 1. 选择要查看的病历 2. 点击查看 | | 无 | 查看病历详情成功 | 查看病历详情成功 |
| 03 | 点击退出 | | 无 | 退出病历查询页面 | 退出成功 |

6.3 客户管理模块测试

6.3.1 查看客户列表功能测试

表6.3.1 查看客户列表功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-3-1 | 功能特性 | 测试查看客户列表功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以医生身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | （1）点击客户管理  （2）查看客户列表 | | 通过cookie获取医生名称 | 查看客户列表成功 | 查看客户列表成功 |
| 02 | （1）选择要查看的客户  （2）点击查看客户详情信息 | | 无 | 查看客户详情成功 | 查看客户详情成功 |
| 03 | 点击退出 | | 无 | 退出客户列表页面 | 退出成功 |

6.4 运营数据分析模块测试

6.4.1 查看经营数据分析功能测试

表6.4.1 查看经营数据分析功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-4-1 | 功能特性 | 测试查看经营数据分析功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以管理者身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | 点击经营数据分析 | | 无 | 查看经营数据分析成功 | 查看经营数据分析成功 |
| 03 | 点击退出 | | 无 | 退出经营数据分析页面 | 退出成功 |

6.4.2 查看医生的业绩数据分析功能测试

表6.4.2 查看医生的业绩数据分析功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-4-2 | 功能特性 | 测试查看医生的业绩数据分析功能功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以管理者身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | 点击查看医生的业绩数据分析 | | 无 | 查看医生的业绩数据分析成功 | 查看医生的业绩数据分析成功 |
| 03 | 点击退出 | | 无 | 退出医生的业绩数据分析功能页面 | 退出成功 |

6.4.3 查看经营数据分析功能测试

表6.4.3 查看经营数据分析功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用例编号 | | 6-4-3 | 功能特性 | 测试查看患者情况分析功能是否可以正常运行 | |
| 测试目的 | | | 验证数据的正确性 | | |
| 前置条件 | | | 用户以管理者身份正常登陆成功 | | |
| 用例分支 | 操作描述 | | 数据 | 预期结果 | 实际结果 |
| 01 | 点击查看患者情况分析 | | 无 | 查看患者情况分析成功 | 查看患者情况分析成功 |
| 03 | 点击退出 | | 无 | 退出患者情况分析页面 | 退出成功 |

7 工作总结与展望

7.1 工作总结

中医诊所管理系统是在对我国中医诊所现状的调查研究下后，再结合我国中医诊所存在的问题的情况下进行开发的。目前我国中医诊所的发展存在以下问题：

（1）管理效率低下；

（2）人才缺乏；

（3）缺乏营销手段。

因此，中医诊所管理系统针对以上问题实现下列功能：

（1）就诊管理实现患者预约缴费、查看预约、就诊期间缴费等功能以及实现医生加号病人、线上配制处方、查看病历等功能。

（2）医生工作站模块实现存储患者病历的功能，提供医生查看病人病历、选用智能处方模版，也可提供医生查看相似病历，使得医生就诊更加高效化。

（3）客户管理实现对病人就诊后的问候功能，能够建起立良好的医患关系，也能够获体现中医的关怀特色。

（4）运行数据分析实现经营数据分析、医生的业绩分析、患者情况分析等功能，通过echarts实现数据的可视化。

目前，中医诊所管理系统已经实现了计划的功能。

7.2 工作展望

中医诊所信息化管理实现了预约挂号、就诊、自助缴费、病历管理、问候客户等功能，这些功能的实现可以极大地提高门诊各个岗位的工作效率，还能够建起立良好的医患关系，体现了中医的特色关怀。但是由于客观因素的限制，该系统还有一些需要完善的地方，包括下面几点：

1. 系统页面的优化：由于系统开发时间有限，该中医诊所管理系统的页面还需要进一步优化，需要更加切近中医特色。
2. 细节功能的完善：该系统的主要功能已经开发完成，但是一些细节方面还需要进一步完善，比如可以提供患者查看门诊的相关信息，包括发展历程、所获荣誉、门诊优势、医生详细信息等。
3. 手机端的支持：目前开发的中医诊所管理系统是支持PC端的，考虑到系统的便捷性，也应该让该系统适应手机端，促进中医诊所管理系统的多样化和实用性。

参考文献

[1]vue官网[DB/OL].<https://cn.vuejs.org/v2/guide/>, 2014.

[2]菜鸟教程[DB/OL].<https://www.runoob.com/nodejs/nodejs-tutorial.html>,2013.

[3]菜鸟教程[DB/OL].<https://www.runoob.com/mongodb/mongodb-intro.html>,2013.

[4]陈海鸿. 面向小型诊所的医疗管理系统设计与实现[D].中山大学,2015.

致 谢

本课题在选题及研究过程中得到邹向华老师的悉心指导。老师的严谨细致、一丝不苟的作风一直是我工作、学习中的榜样，导师们的高深精湛的造诣与严谨求实的治学精神，将永远激励着我。这四年中还得到众多老师的关心支持和帮助。在此，谨向老师们致以衷心的感谢和崇高的敬意！

最后，我还要感谢我的家人们和朋友们，是家人的良苦用心和朋友们的鼓励支持，才让我有所成长，忘却伤痛，继续坚强前行。谢谢你们，我会继续努力！