中文摘要

随着 Web2.0 观念的深入人心与 HTML5 标准的确立，近年来 Web 技术发展迅猛，浏览器前端能实现的功能日益强大，计算效率显著提升。在这样的发展背景下 WebGIS 迎来了良好的发展时期，使得地理信息系统(GIS)从一门大众闻所未闻的学科而融入到了普通百姓的日常生活中。

整个 Web App 使用了前端模块化开发的模式，借助 Require.js 按需加载相应的 js 文件，使得单一文件功能更独立，业务逻辑与通用组件实现了分离，文件间耦合性更低内聚性更高，实现了引入相关 js 即可实现相关功能的目的；借助 Backbone.js 这一前端 MV*框架实现了视图与数据分离的目的，同时作为单页面应用也让 WebApp 更像原生应用，没有了页面间切换的间隔，用户体验更好；使用 Backbone.js——Node.js——MongoDB 这一整套技术来构建整个应用，使得前后端与数据库都使用 JavaScript 这一种语言来开发，降低了开发语言间切换所导致的低效率与学习不同语言导致的高学习成本。

论文以“老有所医 WebApp”为依托，思考了如何通过现有地理信息基础服务平台(高德地图 JavaScript API)来实现定制化应用，向真正需要提供位置查询的人群给予地理信息相关服务的帮助，与此同时获取相关地理信息数据进行时空分析来做预见性决策。

关键词：WebApp　WebGIS　前端开发　Backbone.js　Require.js

Abstract

With the establishment of the concept of Web2.0 and HTML5 standard, Web technology has developed rapidly in recent years, the browser front end can achieve more and more powerful, and the computational efficiency is significantly improved.Under such a background of the development of WebGIS ushered in a good period of development, the geographic information system (GIS) from a mass unheard of disciplines and into the common people's daily life.

The web app using the modular front-end development mode, with Require.js on-demand loading the corresponding JS file, the single file function more single, business logic and common components to achieve the separation, file between the coupling of lower cohesion of more high,realizing the introduction of relevant JS can achieve the purpose of related functions;With the help of Backbone.js ,a MV* framework of front-end,achieving the purpose of the separation of view and data,while the single page applications also make WebApp more like a native application, there is no page switching between the interval, the user experience better;Using Backbone.js Node.js MongoDB to build the entire contents of this set of applications, so that the front end back end and databases only use one type of language- JavaScript to develop, reducing the low efficiency of the development of the inter language switch and the high learning costs caused by different languages.

The paper was relied on "LaoYouSuoYi WebApp",thinking about how to use the existing geographic information based service platform (Gaode map API of JavaScript) to achieve customized applications.To the people really need to provide location queries with the help of geographic information services,meanwhile obtain related geographic information data in time and space to do predictive decision analysis.

Keywords: WebApp WebGIS Front-end development Backbone.js Require.js

目　　录

第一章 绪论

1.1 研究背景

1.1.1 人口老龄化

按照世界卫生组织规定来看，如果总人口数的 10%为 60 岁以上老年人或者 7%及以上占比为 65 岁以上老年人，这样的社会现象就称为人口老龄化[1]。依照这个标准，我国在 2000 年底就已经进入了老龄社会。截止到 2015 年我国老年人口总数突破 2 亿人，2027 年超过 3 亿人。虽然依据人口发布情况预测来看在 2040 年前后印度总人口将超过中国，但是在一段时间内中国仍将占据世界上老年人口最多国家的榜首。

然而，由于诸多因素的限制，我国老年医疗服务需要量却并未顺利转化为医疗有效需求。由于人体机能的衰退，老年人更易患传染性疾病和各类慢性疾病，两者的患病率都是正常人群的 2~3 倍，但由于治病难，治病贵怕花钱等观念的影响，老年人的医疗服务率却低于总人口医疗水平。而且，仍有 2/3 左右的老人患病而未能就诊，是总人口未就诊率的近 2 倍;至少半数的患病老人需住院而未能住院治疗[2]。医疗有效需求不足阻碍了老年人健康状况的进一步改善，而老龄化又对医疗服务提出了更严峻的挑战。

1.1.2 就医推荐机制混乱，莆田系医院扰乱市场

经过对市场上的几款主流医疗类应用进行分析后发现，每款应用都有着很强的主观性，包括百度搜索得出的医院排名也是存在竞价机制的，这导致用户很难选择到真正的好医院，莆田系医院现在大行其道，其疗法很多都不具备科学依据，这样很可能会耽误了患者的救治时间从而引发更严重的并发症。

1.1.3 地理信息服务蓬勃发展

测绘地理信息技术与相关高新技术加速融合的趋势逐步加强，相关应用加速出现，产业活动更加繁荣，“大众化”趋势更为明显。从国际上看，以美国为首的发达国家继续在测绘地理信息技术创新及应用中处于领先地位，并以其技术、资本等方面的优势继续加紧抢占我国市场，并对我国国家安全构成威胁。从国内来看，以企业为主体的创新体系加速构建。北斗卫星系统，卫星对地观测体系，现代化测绘基准体系，测绘地理信

息处理、管理及服务等基础设施加速完善[3]。不同领域地理信息数据共享、集成应用以及产业化开发日趋活跃。

1.2 研究意义

1.2.1 促进社会医疗体系的完善

最近一段时间空巢老人的现象不断涌现，子女外出打工就业造成了这样的现象。在这样的情况下老人进行医疗救治变得尤为困难，与此同时还伴随着病急乱投医的出现，使得老人的医疗健康得不到保证。“老有所医”WebApp 的提出就是为了能够让老人在需要医疗救治的时候能够多一份选择，能够找到最适合自己病情需要的医疗机构来进行相关的诊疗。

1.2.2 提高社会资源利用率，使得资源合理分配

现如今的医疗保障制度其实是有很明确的层级划分的，但是低层次的医疗机构往往不能得到很好的利用。我认为这样情况出现的原因在于老人对于自身病情并不能很好的把控，出于大医院医疗环境更好的心态往往会选择等级更高的医疗机构，这其实往往使得医疗资源无法物尽其用。在“老有所医”WebApp 中提供了根据病情搜索功能，可以按照老人描述的病情合理的推荐相关医疗机构，具有针对性的提供诊疗。

1.2.3 地理信息服务与医疗密切结合

在“老有所医”WebApp 中充分地将地理信息服务与老年医疗密切结合，针对老人所在的地理位置进行范围搜索，使得周边的医疗机构一览无余，在同一条件下老人能有更多的选择；智能手机的普及使得 GPS 定位更加容易，与此同时我们采取精度更高耗电更少安全性更高的 LBS 定位服务，只需要基于基站网络定位，室内室外均可进行，可行性更高，更易用。

1.2.4 利用领先技术解决实际问题

在选题的实现上采取了最先进的前后端与数据库技术，Node.js+Backbone.js+Mongodb，使得先进技术能够应用于生活中实际问题的解决；利用 HTML5 新提供的 Geolocation 技术简化定位流程，使得整个过程更简便更易用，缩短了基础原理的研究时间而更多的着眼于功能的实现上，将毕业设计真正的作为一份产品去对待而非只是空谈。

1.3 国内外研究现状

1.3.1 老年医疗保障体系研究现状

1）医疗保险体系

国外老年医疗保障系统主要由政府主导，企业参与合作的形式进行开展，这其中政府决策起到了很大的作用，主要是建立了医疗照顾制度，医疗补助制度，补充医疗保险及长期护理保险这几项，更多的是从经济上给予补助政策上给予支持，在完善疾病发生后赔偿上下了很大功夫，个人要缴纳的费用很低，但是有着十分优质的医疗保障服务[4]。

我国医疗保障体系存在问题较多，主要集中在以下几个方面：1.覆盖范围狭窄，在城镇受到医疗保障的群体主要是国家企事业单位，贫困家庭是最需要保护的反而因为无力支付固定费用便无法享受到相关的制度保障；2.政府投入不足，政府在老年医疗上的投入资金明显不足，老年人相较其他年龄层次人群来说患病几率大种类繁杂，个人医疗支出比例大，使得许多老年人无力支付高额的医疗费用从而耽误了救治的最好时机，加速了病情的发展。

2）医疗服务提供体系

国外主要是建立了一些老年医疗设施，例如老人医院，老人病房，养老院，老年收容所等针对不同社会阶级的老年群体提供服务，与积极治疗相比更佳侧重于护理与病前预防，尤其倡导家庭康复治疗，与专门的健康康复医疗机构进行密切的合作。服务体系的建立多是以区域为基础，以社区为依托紧密结合的多层次医疗服务体系，在老人有护理需求时进行相关评估，使得各级资源得以合理利用[5]。

而我国目前有部分医院盲目追求经济效益，药品定价虚高，普遍存在以药养医的补偿机制，使得医疗保障体制存在严重问题，生活水平较高的老人在就医过程中往往存在大医院能治病的心理，无论病况大小都刻意去大医院看的情况使得各级医疗资源得不到很好的利用，而条件相对较差的家庭则存在治病花费太高负担不起的心态，也没有认识到社区医疗机构的存在意义和福利便民的特性。

1.3.2 WebApp 开发研究现状

从名称上可以看出，Web App 即互联网与应用的结合，是将各种硬件设备都具备的浏览器作为客户端，实现传统应用所能够完成的任务的一类应用程序。Web App 是 Google 在设计 Chrome 时提出的概念，是 Google 推广云计算的其中一步。

Web App 发展不乏经典。亚马逊、谷歌、Airbnb、微软等著名的互联网企业都运用了 WebApp 相关技术来构建简单易用的相关 app 来实现浏览即用得应用程序。Gmail 更是 Web App 的经典，Gmail 集合了 Gtalk、Google Voice、Google Plus 等 App，方便用户通过电脑以及智能移动终端直接访问。用户只要登录 Gmail 即可获取相应产品的推送服务。国内，像百度、360、UC 等互联网公司均已开发 Web App[6]。

1.3.3 WebGIS 发展状况

近年来地理信息系统相关技术发展势头迅猛，与地理信息服务相关的应用层出不穷，为我们的日常生活带来了许多便利。与此同时由于互联网的普及和发展，促使在新的环境下产生了对 GIS 新的认识，即地理信息系统应用于互联网，就是我们通常称的 WebGIS[7]。顾名思义，借助互联网这一平台实现 GIS 相关功能，将 GIS 的互联网特性发扬光大就成为了 WebGIS，这使得束之高阁的 GIS 软件变得平易近人，真正的能够走入到大家的生活。

1.4 主要研究内容

思考并了解中老年就医需求，结合 WebGIS 与 LBS 地理定位服务开发老有所医 WebApp，借助应用达到优化中老年人就医的目的，同时根据用户使用时获取到的信息和数据进行反向分析，了解用户习惯与疾病发展趋势，进行时空分析。论文首先对毕业设计过程中应用到的技术进行介绍，进一步以此为基础进行系统设计、前端页面设计、数据库设计和后端接口设计。紧接着是项目开发阶段，在这一过程中建立目标发现问题解决问题并完善整个项目，当项目完成后对整个过程进行了总结并发现了缺点与不足。

1.5 结构安排

第一章：绪论。主要就毕业设计研究背景、意义、国内外发展现状及研究主要内容做概要性介绍。

第二章：WebGIS 基础与前后端应用技术。主要就毕业设计开发过程中使用到的具体技术做调研和分析，包括 WebGIS 基础知识、前端基础知识、前端基础类库、数据库基础知识和后端基础知识几块内容。

第三章：老有所医 WebApp 设计与实现。主要介绍老有所医 WebApp 的前端实现包括

功能设计、页面设计、Backbone.js 和 Require.js 在项目中的应用、前端相关优化点、数据库设计和后端服务器与接口设计几大部分。

第四章：论文的总结与展望。主要进行了项目的工作内容与成果汇报、项目中存在的缺点和不足汇总与日后发展的期望。

第二章 WebGIS 基础与前后端应用技术

2.1 WebGIS 技术

2.1.1 WebGIS 概念

WebGIS 是指通过互联网对地理空间数据进行采集、编辑、发布、共享进一步产生应用，以实现空间数据的共享和交互操作，具体使用例子如在线地图，在线地形分析等等[8]。WebGIS 与以往的常规 GIS 系统不同在于采用了 B/S 模式，用户使用常见的浏览器如 Chrome、Safari、360 浏览器等等进行相关操作，无需安装专业软件，使得专业化技术性功能实现起来更方便。顾名思义，WebGIS 是借助 Web 这一平台来实现 GIS 相关功能的技术，通过网站与网页的方式来实现借助浏览器来管理，操作，发布和应用空闲信息的功能。WebGIS 可以通过多种方式来进行部署，实现形式多样。

2.1.2 WebGIS 原理

从名字上来看，WebGIS 只是搭载环境由本地转为了 Web 平台，实现的功能与原有的 GIS 软件并没有太多的区别，因为现在 Web 服务器与本地客户端(浏览器)的计算能力越来越强，原本大强度的计算已经可以满足；它的优点在于解决了早期 GIS 软件网络相关功能的缺失，能够更好地进行实时的交互，信息的交换更为方便，使用条件更低，而它的基础就是 Web 相关基础技术。WebGIS 将常见的浏览器作为客户端，用户通过发送相关请求来获取后端服务器上的普通信息与地理信息数据库中的地理信息文件来实现相关功能。后端返回的内容可以分为两种，一种是直接将查询到的数据返回到客户端，另外一种是将渲染好的页面发送给客户端。与此同时地理信息文件也可以分为多种类型，例如数据型，栅格图片型，矢量图片型等。

2.1.3 WebGIS 实现形式

主要有 3 种 WebGIS 体系结构模式：

1）基于服务器端的模式。这种情况下无论是数据的获取还是相关的计算都由后端服务器来进行，当后端服务器完成数据获取和计算操作后将最终的结果以图片的形式或矢量图 SVG 或 Canvas 画布形式传达到用户的浏览器中，此时浏览器只是作为展示工具

而不进行相关的实现。

2）基于客户端的模式。当采用这种模式时当用户需要某些 GIS 数据时就会向后端服务器发起请求，服务器接收到请求后会将数据返回到浏览器中，浏览器通过自身的计算能力来进行相关操作，这对浏览器的性能和用户自己的计算机性能有一定的要求，所以一般只是进行简单的数据操作，操作完成后将结果展示给用户。

3）分布式模式。这种模式相比较前者而言更为符合我们平时的使用要求，对于复杂的功能比如地形分析，相关性分析，插值等都由服务器来进行操作只返回最终的结果；而简单的操作比如地图的展示，拖动，查询，简单编辑等则由浏览器本身来进行实现。基于这样的操作形式使得服务器与客户端(浏览器)都得以物尽其用，充分发挥了两者的功能，同时之所以能实现这样的变化是因为浏览器的不断发展和完善，能实现的功能与计算能力越来越强大。

2.1.4 HTML5 GeoLocation 功能介绍

HTML5 在设计之初就公开了 Geolocation 相关 API 接口，它可以借助用户硬件设备和网络信息来获取当前位置，使用获取的位置可以进行一些进一步相关的操作。Geolocation 的位置信息来源包括 GPS、IP 地址、RFID、WIFI 和蓝牙的 MAC 地址、以及 GSM/CDMS 的 ID 等等。规范中没有规定使用这些设备的先后顺序，在 HTML5 的实现中，手机等移动设备优先使用 GPS 定位，而笔记本和部分平板，最准的定位是 WIFI，至于网线上网的台式机，一般就只能使用 IP 来定位了，这个准确度最低。HTML5 本身提供的方法并不是十分便捷的，所以我自己进行了封装并以此来说明具体该如何使用 HTML5 的这一新特性：

```
var geo = {
    checkGeo: function() {
        if (navigator.geolocation) {
            return true
        } else {
            alert('浏览器该更新了~');
            return false
        }
    },
    getPosition: function(){
```

```
        var that = this;
        navigator.geolocation.getCurrentPosition(that.successFun，that.errorFun);
    },
    successFun: function(position){
        var coords = position.coords;
        var lng = coords.longitude;
        var lat = coords.latitude;
        var pos = {
            lng:lng，
            lat:lat
        }
        return pos;
    },
    errorFun: function(error){
        alert(error.msg);
    }
}
```

2.1.5 高德地图 JavaScript API 介绍

高德地图是国内知名的地图服务供应商之一，其提供的 JavaScript API 暴露了很多可以由客户端直接发起访问的应用程序接口，比如基本的地图显示，漫游缩放、标记点的相关临近搜索、出行路线规划、添加标记、实时路况等功能，与此同时我们可以基于已有的功能进行二次开发，比如重载地图点击事件，调整标记点的显示内容等。高德官方提供了完整且详细的使用指南及示例中心，在开发过程中随时可以进行查阅和了解，具体的使用方法在第三章中会进行介绍。

2.2 前端核心技术

2.2.1 HTML

HTML 是网站的脊梁，它是构建一切网页和网站的基础超文本标记语言。HTML 是一种基础技术，常与 CSS、JavaScript 一起被众多网站用于设计令人赏心悦目的网页、

网页应用程序以及移动应用程序的用户界面。网页浏览器可以读取 HTML 文件，并将其渲染成可视化网页。HTML 描述了一个网站的结构语义随着线索的呈现，使之成为一种标记语言而非编程语言。HTML 元素是构建网站的基石。HTML 允许嵌入各种多媒体，并且可以用于创建交互式表单，通过具有语义化的标签包裹相应的内容来实现信息的结构化。HTML 以尖括号来定义标签，同时大部分标签都成对使用，标签内包裹具体的内容。HTML 可以嵌入如 JavaScript 的脚本语言，它们会影响 HTML 网页的行为。网页浏览器也可以引用层叠样式表（CSS）来定义文本和其它元素的外观与布局。维护 HTML 和 CSS 标准的组织万维网联盟（W3C）鼓励人们使用 CSS 替代一些用于表现的 HTML 元素。近几年来，随着 Web 开发标准版本的不断更新 HTML5 也被提出并得到了广泛的应用。和以往的 Web 开发标准不同，它是完成了标准化历史的一次飞跃，HTML5 除了用来表示 Web 内容，他的另一个任务是实现包括视频、声音、图片等在内网络资源标准化交互。HTML5 新内容有以下几方面(但不限于)：

1）语义特性

HTML5 赋予网页更好的意义和结构。更加丰富的标签将随着对 RDFa 的，微数据与微格式等方面的支持，构建对程序、对用户都更有价值的数据驱动的 Web。HTML5 赋予网页更好的意义和结构。更加丰富的标签将随着对 RDFa 的，微数据与微格式等方面的支持，构建对程序、对用户都更有价值的数据驱动的 Web[9]。

2）本地存储特性

HTML5 提出了 LocalStorage 和 SessionStorage 两种本地数据存储形式，这相较于之前的 cookie 机制和 session 机制而言可以为 WebApp 提供更多的存储空间实现更高的性能，也不必每次都将相关的信息发送到服务器去导致资源的浪费，这是 HTML5 更新的最重要的技术之一。

3）设备兼容特性

HTML5 为网页应用开发者们提供了更多功能上的优化选择，带来了更多体验功能的优势，现在浏览器也能调用很多原本只有原生 APP 才能调用的接口，例如 Geolocation 功能也可以调用移动设备本身的 GPS 模块，这相比较之前而言定位精度的提升是十分明显的，HTML5 提供了前所未有的数据与应用接入开放接口。

4）连接特性

WebSockets API 被提出后使得基于网页的 IM 实时通讯功能的实现变得尤为简单，这就级高了连接的工作效率，需要持续连接的相关服务都得以优化，实时推送功能也不

在局限于原生应用，HTML5 拥有更有效的服务器推送技术，借助 Server-Sent Event 就可以实现相关的功能。

2.2.2 CSS

CSS（Cascading Style Sheets，级联样式表）是用来实现如何将我们想表达的内容通过更合理的布局和设计来展现给网站浏览者的一种标记性语言，我们给 HTML 中的元素设置不同的样式属性使得他们能按照我们想要的效果来展示。它具有以下几个突出优点：内容和表现相互分离；使网页浏览速度得到提高；维护和改版简便[10]。

CSS1.0（最早的 CSS 版本）只支持最基础的 HTML 标签元素，CSS2 添加了对媒介（打印机和听觉设备）和可下载字体的支持，随着前端模块化开发的流行，CSS3 语言也向模块化方向发展，CSS3 把之前版本的 CSS 分解成一些较小的模块，同时也加入了更多新的诸如盒模型、列表、文字特效等模块。

CSS3 使网页开发者不必再用图片处理软件或者 JavaScript 脚本控制去完成 3D 动画、圆角、用户自定义字体、多背景、盒阴影、渐变、文字阴影、透明度，从而使 Web 设计质量大有提高更具特色。

2.2.3 JavaScript

JavaScript，一种直译式脚本语言，是一种动态类型、基于原型的语言，内置支持类。它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在 HTML 网页上使用，用来给 HTML 网页增加动态功能。然而现在 JavaScript 也可被用于网络服务器，如 Node.js。

一般来说，完整的 JavaScript 包括以下几个部分：

1）ECMAScript，描述了该语言的语法和基本对象；

2）文档对象模型（DOM），描述处理网页内容的方法和接口；

3）浏览器对象模型（BOM），描述与浏览器进行交互的方法和接口。

它的基本特点如下：

1）是一种解释性脚本语言（代码不进行预编译）；

2）主要用来向 HTML 页面添加交互行为；

3）可以直接嵌入 HTML 页面，但写成单独的 js 文件有利于结构和行为的分离。

JavaScript 常用来完成以下任务：

1）嵌入动态文本于 HTML 页面；

2）对浏览器事件作出响应；

3）读写 HTML 元素；

4）在数据被提交到服务器之前验证数据；

5）检测访客的浏览器信息；

6）控制 cookies，包括创建和修改等。

2.3 WebApp 介绍

2.3.1 Web2.0

Web2.0 是相对于 Web1.0 的新的时代。指的是一个利用 Web 的平台，通过了解用户想要获得的体验来进行网站的相关开发，这里更多的考虑用户的需求点是什么，而不是盲目的进行信息的展示和传递，为了区别传统由站长或编辑人员来决定要展示什么给用户的网站类型而定义为第二代互联网，即 web2.0。Web 2.0 是一种新的互联网方式，通过网络应用（Web Applications）促进网络上人与人间的信息交换和协同合作，其模式更加以用户为中心。典型的 Web 2.0 站点有：实时通信 IM，博客，维基百科，地理服务应用，社交网站等等。Web 2.0 的支持者认为 Web 正在由最一开始的展示性传播性功能向交互性与社会性发生转变，用户更多的是想表达自己的观点而不是单纯的获取相关的资讯和新闻。某种观点认为，和传统网站相比，Web 2.0 的网站更多表现为 Point of presence 或者是用户产生内容的门户网站。Web2.0 时代所基于的技术中有一项为 LBS(Location Based Services)，它是集地理信息系统（GIS）、微博（Twitter）和移动设备（Mobile）以及 A-GPS 定位服务于一身的增强型微博系统，这与传统的信息展示不同，它将地理信息空间要素与传统的信息相结合，既可以进行时间上的筛选和聚合，也可以根据地理要素来进行汇总，实现时空相关性分析。

2.3.2 单页面应用

SPA，即 Single Page Application 单页面应用，浏览器会在一开始加载必须的 HTML，js 和 CSS，之后所有的操作都是基于这些内容，后续的操作是通过 js 来进行实现的，全程只有一个页面，显示内容有变化是因为进行了视图切换。单页面应用（SPA）是旨在对 Web 应用发动革命的软件运动中可信度很高的一种。此类应用有望进行更加模块化的开发，令应用更加容易地适配与多个设备，并拥有更好的应用生命周期管理—这些几乎是软件架构师希望的全部。单页面应用诞生于拥有更多动态页面内容的 Web 2.0 革

命。旧的超链接页面浏览模型给用户带来了不和谐的体验，而 Web 2.0 原则允许数据驱动时间在一个页面内创建，并让页面内容在需要的时候更新。单页面应用诞生于拥有更多动态页面内容的Web 2.0革命。旧的超链接页面浏览模型给用户带来了不和谐的体验，而原则允许数据驱动时间在一个页面内创建，并让页面内容在需要的时候更新。这意味着应用似乎可以运行得更加流畅，乃至于到达可仿真桌面与本地资源接口的地步。

单页面应用的优点：

1）用户体验好、速度更快快，内容的改变不需要重新加载整个页面；

2）良好的前后端分离。由于后续只做数据请求，后端不再负责模板渲染、输出页面工作，后端 API 通用化，很多时候 iOS/Android/H5 可以共用一套 API，降低了开发成本。

缺点：

1）不利于页面 SEO(搜索引擎优化) ；

2）当所有资源被压缩为一个文件后，初次加载时耗时相对增多；

3）由于单页应用，视图的前进和后退就成了问题。

2.4 前端辅助框架

2.4.1 jQuery

jQuery 是一套跨浏览器的 JavaScript 库，目的在于简化 HTML 与 JavaScript 之间的操作。JQuery 是继 prototype 之后又一个优秀的 Javascript 库。虽然功能强大但是其体积却很轻量，同时提供了压缩版的 min 版本以供生产环境使用。不同的浏览器间本身是存在差异的，jQuery 的存在基本消除了这些差异，对常见的浏览器如 IE6+、Safari、Opera、Chrome 以及国内的 360 浏览器等都具有很好的兼容性。后期的最新版本由于功能的升级和优化所以舍弃了对低版本 IE 的兼容，所以在使用过程中要注意是否需要对低版本浏览器进行优化。如今，jQuery 已经成为最流行的 javascript 库，在世界前 10000 个访问最多的网站中，有超过 55%在使用 jQuery[11]。正是由于使用用户众多这也就带来了一个十分显著的优点：可参考的文档及使用的插件众多，有些你想要实现的功能已经预先有人完成了实现并封装好了方法，只要学习是如何实现的相关功能就能很快的上手使用，这大大提高了开发效率，缩短了组件的开发时间。

jQuery 是一个兼容多浏览器的 javascript 库，核心理念是 write less，do more(写得

更少，做得更多)。同时它是免费、开源的，使用 MIT 许可协议。jQuery 的语法设计可以使开发更加便捷，例如操作文档对象、选择 DOM 元素、制作动画效果、事件处理、使用 Ajax 以及其他功能。除此以外，jQuery 提供 API 让开发者编写插件。开发者可以根据自己的需要对 jQuery 进行相关的扩展，添加相关的辅助方法或者组件，使得重交互性的网页可以被轻易地实现。

jQuery，顾名思义，也就是 JavaScript 和查询（Query），即是辅助 JavaScript 开发的库。

2.4.2 Require.js

Require.js 的诞生是为了解决两大问题，第一实现 js 文件的异步加载，避免网页失去响应，第二是管理模块之间的依赖性，便于代码的编写和维护。最早的时候，所有 Javascript 代码都写在一个文件里面，只要加载这一个文件就够了。后来，代码越来越多，一个文件不够了，必须分成多个文件，依次加载。这样的写法有很大的缺点。首先，加载的时候，浏览器会停止网页渲染，加载文件越多，网页失去响应的时间就会越长；其次，由于 js 文件之间存在依赖关系，因此必须严格保证加载顺序，依赖性最大的模块一定要放到最后加载，当依赖关系很复杂的时候，代码的编写和维护都会变得困难。在使用了 Require.js 之后我们只需要在页面中引用这一个 js 就够了，例子如下：

```
<script data-main="main.js" src="Require.js"></script>
```

main.js 是实现模块化加载的入口，我们在其中定义了相关依赖的项，例如：

```
require.config({
    baseUrl:'widget',
    paths: {
        jquery:'ui/jquery/jquery-2.2.2.min',
        template:'ui/template/template',
        underscore:'ui/underscore/underscore',
        Backbone:'ui/Backbone/Backbone',
    }
});
```

baseUrl 定义了模块加载的基准路径，而当我们自定义一个模块时则需要这样进行：

```
define(['Backbone','template','background/count/tpls','ui/helper/helper','ui/map/map'],
function(Backbone， T， tpls，helper，map) {
```

```
    var obj = {
        ...
        ...
    };
    return obj;
});
```

define 中可以预先说明这一模块依赖的项，并在后面的 function 中暴露出相应的对象以供该模块进行使用，详细的用法在第三章老有所医 WebApp 设计与实现中进行叙述。

2.4.3 Backbone.js

Backbone.js 是一种轻量级的 JavaScript 库，用于将结构添加到客户端代码。它可以让应用程序的关注点管理和解耦工作变得容易，从长远来看，它可以使代码更易于维护。开发人员通常使用像 Backbone.js 这样的库来创建单页面应用程序(SPA)，上文也已经提到了 SPA 是一种将页面加载到浏览器，然后就不需要从服务器刷新整个页面，就可以在客户端完成交互的 Web 应用程序。很多现代 JavaScript 框架为开发人员提供使用 MVC(Model-View-Controller，模型-视图-控制器)模式的辩题来轻松组织代码的方法。MVC 将应用程序中关注点分为以下三个部分：

1）模型

模型(Model)代表了应用程序中的特定领域知识和数据。可以将它想象为一个可以建模的数据类型，如一个用户、一张图片或备忘录。模型在其状态发生改变时可以通知其他观察者。

2）视图

视图(View)通常用于在应用程序中构成用户界面(如标记和模板)，但不是必须的。它们观察着模型，但并不与它们直接通信。

3）控制器

控制器(Controller)用于处理输入(单击或用户操作)以及更新模型。

因此，在 MVC 应用程序中，用户的输入受控制器的支配，而控制器更新模型。视图观察模型，并在模型发生更改时更新用户界面。然而 JavaScript MVC 框架并不总是严格遵守这种模式，一些解决方案将控制器的任务合并到了视图，而其他方法也混入到额外的组件中，因此更合适的称呼方式为 MV*，有其他的组件会负责完成 controller 的

工作。Backbone 中的主要组成部分包含视图(View)、模型(Model)、集合(Collection 模型的聚合物)，路由(Router)等几部分，形成了自己独特的 MV*模式。在视图和模型之间支持事件驱动的通信，在任何模型的属性上添加事件监听器、针对视图中的变化为开发者提供细粒度控制是相对简单的；Backbone 只强制依赖了 underscore.js 这一个 javascript 库，其他需要的中间件或者要实现的功能都可以通过自己的需要来进行加载。

2.4.4 art-Template

由于借助 Backbone.js 实现了 SPA 应用，因此页面的渲染和加载都需要由前端来进行而不是直接展示后端拼接好的页面，所以选择一个合适的前端模板是十分必要的。Backbone 本身强制依赖的 underscore 提供了微型模板可以进行使用，但是由于语法过于粗糙且实现的功能单一，所以就开始寻找合适的前端模板。art-template 是新一代 javascript 模板引擎，它有如下一些特性：

1）性能卓越，执行速度通常是 Mustache 与 tmpl 的 20 多倍，更遑论 underscore 的微模板；

2）支持运行时调试，可精确定位异常模板所在语句，便于进行调试；

3）对 NodeJS、Express 友好支持(后端选择了使用 node 来搭建)；

4）安全，默认对输出进行转义、在沙箱中运行编译后的代码（Node 版本可以安全执行用户上传的模板）；

5）支持 include 语句，实现加载子模板；

6）可在浏览器端实现按路径加载模板(更符合 Require.js 要求)；

7）支持预编译，可将模板转换成为非常精简的 js 文件；

8）模板语句简洁，无需前缀引用数据，有简洁版本与原生语法版本可选；

9）支持所有流行的浏览器。

2.5 CSS 预编译语言

2.5.1 Less

Less 是一门 CSS 预处理语言，它扩充了 CSS 语言，增加了诸如变量、混合（mixin）、函数等功能，让 CSS 更易维护、方便制作主题、扩充。作为一门标记性语言，CSS 的语法相对简单，对使用者的要求较低，但同时也带来一些问题：CSS 需要书写大量看似没有逻辑的代码，不方便维护及扩展，不利于复用，尤其对于非前端开发工程师来讲，

往往会因为缺少 CSS 编写经验而很难写出组织良好且易于维护的 CSS 代码，造成这些困难的很大原因源于 CSS 是一门非程序式语言，没有变量、函数、SCOPE（作用域）等概念。LESS 为 Web 开发者带来了福音，它在 CSS 的语法基础之上，引入了变量，Mixin（混入），运算以及函数等功能，大大简化了 CSS 的编写，并且降低了 CSS 的维护成本，就像它的名称所说的那样，LESS 可以让我们用更少的代码做更多的事情。

作为一种动态样式语言，对比 CSS 而言有以下几点优势：

1）嵌套规则，通过选择器名间的嵌套（包裹）来实现层次化的赋予选择器样式，这样使得样式层次清晰，增强了层次间的联系也避免了出现由于层次结构不清而导致的样式设置错误情况的出现；同时减少了代码量，在进行串联选择器的时候使用&是十分高效的，尤其是针对:hover，:selected 等伪类样式的设置。

2）提供了变量的概念，可以单独定义一系列样式，在后续样式中进行使用，使得全局调整更为简便；同时提升了 CSS 语义化，在协同工作时可以通过变量名来了解前期开发者想表达的意图。

3）mixin 混合（也包含了函数），可以将已经定义好的样式引入到新的选择器中，实现了样式的继承；同时可以实现函数模式带参数进行调用，能够更好的根据需求进行修改，实现了样式代码片段的复用。

2.6 后端技术介绍

2.6.1 Node.js

JavaScript 是一种运行在浏览器的脚本，它简单，轻巧，易于编辑，这种脚本通常用于浏览器的前端编程，但是一位开发者 Ryan 有一天发现这种前端式的脚本语言可以运行在服务器上的时候，一场席卷全球的风暴就开始了。

Node.js 是一个基于 Chrome JavaScript 运行时建立的平台，用于方便地搭建响应速度快、易于扩展的网络应用[12]。Node.js 使用事件驱动，非阻塞 I/O 模型而得以轻量和高效，非常适合在分布式设备上运行的数据密集型的实时应用。

Node 是一个 Javascript 运行环境（runtime）。实际上它是对 Google V8 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 对一些特殊用例进行了优化，提供了替代的 API，使得 V8 在非浏览器环境下运行得更好。

Node 是单线程单进程的，而在 node 底层其实是有进程池的。单进程主要负责对事

件队列进行监听，一旦发现有事件出现就将该事件分配到底层的进程池去运行(比如 I/O)操作，同时告诉进程池一个回调函数；当阻塞的事件执行完毕后会将执行结果和回调函数一同返回到事件队列，由单进程负责进行启动回调函数和后续操作的运行，Node 就是通过这样的事件轮询机制实现了高效的服务器（通俗的例子就像在外吃饭点餐后会给你一个叫号器，你在等待的同时就会有人在给你制作，完成后会叫你来取）但是由于回调函数中的任务还是在负责轮询的单线程中执行，这就注定了它不能执行 CPU 繁重的任务。

2.7 数据库应用技术介绍

2.7.1 NoSQL

关系型数据库中的很大一部分操作都是为了保证事务的一致性(Consistency)，而这也是关系型数据库的灵魂，这个特性使得关系型数据库可以用于几乎所有对一致性有要求的系统中，如典型的银行系统；剩下的三个特性原子性(Atomicity)、隔离性(Isolation)、持久性(Durability)都是为了一致性来服务的。

但是，在网页应用中，尤其是 SNS 应用中，一致性却不是显得那么重要，用户 A 看到的内容和用户 B 看到同一用户 C 内容更新不一致是可以容忍的，或者说，两个人看到同一好友的数据更新的时间差那么几秒是可以容忍的，因此，关系型数据库的最大特点在这里已经无用武之地，起码不是那么重要了。

相反的，关系型数据库为了维护一致性所付出的巨大代价就是其读写性能比较差，而像微博，facebook 这类 SNS 的应用，对并发读写能力要求极高，关系型数据库已经无法应付，因此，必须用新的一种数据结构化存储来来代替关系数据库。

而另外一个特点是关系型数据库都是具有固定的表结构的，这就使得一旦有数据结构发生改变原有的数据库就无法满足要求而需要新的存储结构来进行存储，而现如今的 SNS 应用中经常会有这方面的变化，这就导致关系型数据库无法很好的满足这一需求。

当代典型的关系数据库在一些数据敏感的应用中表现了糟糕的性能，例如为巨量文档创建索引、高流量网站的网页服务，以及发送流式媒体。关系型数据库的典型实现主要被调整用于执行规模小而读写频繁，或者大批量极少写访问的事务。

于是，非关系数据库（NoSQL）应运而生，它的存储结构并不一定是固定且不可变的，因此它不具备固定的表结构，从这样的意义上来说非关系型数据库更应该被称为

是一种存储结构化的集合而非数据库[13]。

NoSQL 是指'Not Only SQL'，即不只是数据库系统，这与传统的关系型数据库是存在本质性区别的。这两者存在许多显著的不同点，而其中最重要的是 NoSQL 不使用 SQL 作为查询语言。其数据存储可以不需要固定的表格模式，也经常会避免使用 SQL 的 JOIN 操作，一般有水平可扩展性的特征[14]。NoSQL 的实现具有二个特征：使用硬盘，或者把随机存储器作存储载体。

但是如果需要数据的持久存储，尤其是海量数据的持久存储，关系型数据库的作用还是无法被取代的，在实际的数据库运用过程中往往都是两者相辅相成的进行，各负责一部分数据存储的内容。

2.7.2 MongoDB

MongoDB 是一种文档导向数据库管理系统，由 C++撰写而成，以此来解决应用程序开发社区中的大量现实问题[15]。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。它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。

MongoDB 的特点如下：

1）MongoDB 的提供了一个面向文档存储，操作起来比较简单和容易；

2）你可以在 MongoDB 记录中设置任何属性的索引 (如：FirstName="Sameer"，Address="8 Gandhi Road")来实现更快的排序；

3）你可以通过本地或者网络创建数据镜像，这使得 MongoDB 有更强的扩展性；

4）如果负载的增加（需要更多的存储空间和更强的处理能力），它可以分布在计算机网络中的其他节点上这就是所谓的分片；

5）Mongo 支持丰富的查询表达式。由于存储对象都是 BSON 格式的，所以既可以通过某一字段进行查询也可以查询内嵌的对象和数组，亦可通过正则表达式来进行查询，十分便捷；

6）MongoDb 使用 update()命令可以实现替换完成的文档（数据）或者一些指定的数据字段；

7）MongoDB 允许在服务端执行脚本，可以用 Javascript 编写某个函数，直接在服务端执行，也可以把函数的定义存储在服务端，下次直接调用即可。

第三章 老有所医 WebApp 设计与实现

3.1 系统整体设计

根据毕业设计初衷，最初的设计方案以及后期通过与一些中老年人进行交流对 App 的整体需求进行了细化，完成了对各个功能点的基本设计，基本确定了整个系统的基本架构如图 3-1 所示。老有所医 WebApp 在总体架构上主要分三个部分：前端页面设计，后端服务器与接口设计，数据库设计。前端页面主要分两大块，分别是用户使用页面(移动端)及后台管理页面；根据需求主要分为推送文章，医疗点查询，老年活动室和个人信息四块内容，每块内容又细分若干功能。后端服务器使用 Node.js 进行搭建，同时借助了 Express 框架以简化相关代码编写，使用了 body-parser、Mongoose、path、crypto 等中间件，同时采用了类 RESTful 风格接口，使得数据交换更简单。数据库方面选择了 NoSQL 中的 MongoDB 来存储相关信息，MongoDB 与一般的关系型数据库不同，它存储的数据格式为 BSON，修改起来更为方便。之所以选择了这样一整套技术架构是出于这样的原因：前后端包括数据库都是使用 JavaScript 来写的，在相互切换编写代码的过程中不需要语言的切换，使得代码编写过程更流畅，学习调研成本相对更低，效率更高。

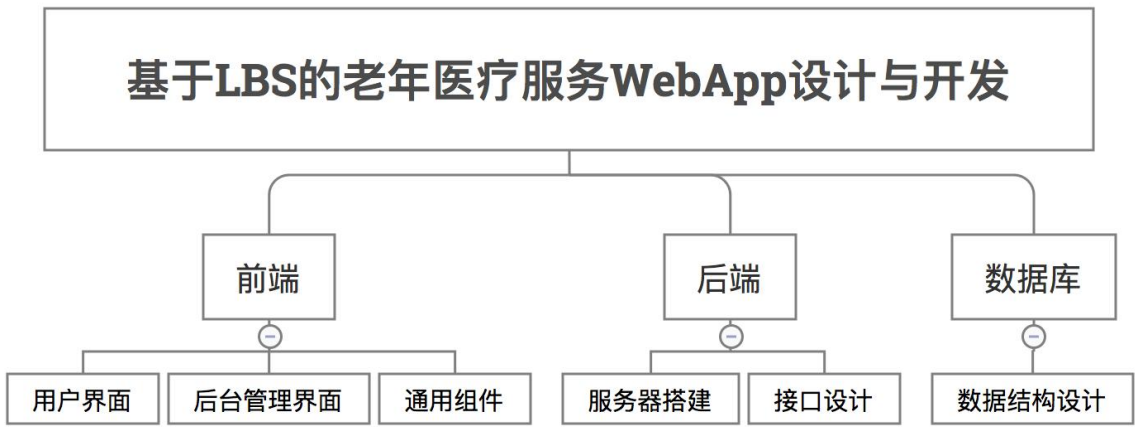

图 3-1 基于 LBS 的老年医疗服务 WebApp 设计与开发基本架构图

3.2 前端设计与实现

3.2.1 页面自适应

由于是 WebApp 开发所以无法避免的问题就是可能会有不计其数的屏幕尺寸出现，

如何解决多设备多屏幕适配是在开发之前亟待要解决的问题。我使用的方案是：通过用户进入页面时获取当前页面尺寸，动态设置根节点 HTML 的字体号大小，同时每当页面尺寸发生变化时也重新设置根字体大小，而后续的所有元素使用 rem 为单位，以 1080px 的设计稿尺寸为基准进行动态取值，主要实现代码如下：

```
(function () {
    var completed = function () {
        setTimeout(function () {
            var RADIX = 10.8;
            var HTML = document.documentElement;
            var windowWidth = HTML.clientWidth;
            HTML.style.fontSize = windowWidth / RADIX + 'px';
        },  16);
    };
    completed();
    window.onresize = completed;
})();
```

样式的实现上主要借助了 less 语法中的函数来实现，如下所示：

```
@radix: 100;
.px2rem(@name,  @px) {
@{name}: @px / @radix * 1rem;
}
```

这里运用到的概念为 dpr，即'屏幕像素尺寸/设计稿尺寸'可以计算出屏幕上 1 像素相当于设计稿上的像素大小，再使用 rem 设置每个元素的相对尺寸，通过这样的方式可以实现针对不同的设备都显示相同效果的目的。

3.2.2 通用组件设计

前端界面中主要涉及的通用组件如图 3-2 所示。

图 3-2 通用组件

1)Geolocation

第二章中对 HTML5 提供的 Geolocation 进行了简要的介绍，这里的 Geolocation 主要是对该方法进行了相应的封装，需要使用时只要载入该对象即可，可以实现地理位置获取及地图初始化的功能。部分功能代码如下：

```
getPosition: function(){
        var that = this;
        navigator.geolocation.getCurrentPosition(that.successFun，that.errorFun);
    },
    successFun: function(position){
        var coords = position.coords;
        var lng = coords.longitude;
        var lat = coords.latitude;
        var pos = {
            lng:lng，
            lat:lat
        }
        map.init(pos);
        return pos;
    },
```

2)Map

Map 组件主要是对高德地图 API 的二次封装和调用。由于多个视图中用到了相关方法，出于优化代码量的目的将所有需要的方法都进行了提取，同时进行了对象封装，主要的方法功能有：init 主要负责页面地图的初始化；initBack 后台使用地图的初始化操作；getMarkerPosition 获取当前地图上标记点的坐标；nearBySearch 临近搜索相关功

能；walkingSearch 步行导航与出行路线规划；drivingSearch 驾车导航与出行路线规划；busSearch 公交出行路线规划。

3)Help

Help 组件主要是定义了一些通用的工具方法以便各个页面进行使用，比如实现了对 localStorage 判断、读/取数据的封装，randomKey 生成随机字符串以确保发送数据的唯一性，queryString 获取查询字符串等。

4)Nav

Nav 组件主要是针对移动端来设计的。每个页面都需要引入底部的 nav 导航而如果在每一个视图中重复载入会导致底部导航的不断刷新，同时也会浪费相应资源，因此将这一部分进行了抽出，只在初次进入页面时加载一次，后续只需要根据用户点击的 nav-name 不同进行 active 切换即可。

3.2.3 用户登录与信息记录机制

由于老有所医 WebApp 是单页面应用，这也就决定了只有第一次进入页面或者用户刷新时才会由后端返回完整的页面回来，其他情况都是由前端来进行页面填充渲染，所以用户信息的获取和保存就不能按照传统形式每次由后端传回。这里使用的方式是定义了全局变量 conf 来保存诸如用户登录信息，用户相关资料等，同时将这些信息保存在了 localStorage 中，这样就避免了出现用户刷新页面致信息丢失的可能，部分实现代码如下：

```
window.conf = {};
conf.is_login = JSON.parse(localStorage.getItem('is_login')) || false;
conf.user_data = JSON.parse(localStorage.getItem('user_data')) || {};
conf.end = {};
conf.begin = {};
conf.is_login = true;
conf.user_data = response.userInfo;
helper.setItem('is_login'，true);
helper.setItem('user_data'，response.userInfo);
```

3.2.4 移动端模块与界面设计

移动端模块划分主要如图 3-3 所示共分为首页、医疗点查询、老年活动室和我的四

个模块，接下来会依次进行分析。

图 3-3 移动端模块与界面设计

1)首页模块

首页是用户进入整个 App 最先接触的模块，主要功能为展示 Web 推送的相关健康医疗文章，这时会加载 index-list 视图，同时向后端请求推送文章列表信息，并为每一条推送信息添加点击事件，点击后就会跳转到相关的文章详情页面去。用户进入文章详情页面时会将文章的相关 id 添加到 url 地址中，通过 help 组件中的 queryLocationSearch 方法获得，并通过该 id 向后端请求相关的文章详情，成功返回信息后会加载 index-info 视图，展示相关文章信息。

2)医疗点查询模块

医疗点查询是老有所医 WebApp 中的功能重点，在使用这一页面功能的时候会先进行判断用户是否进行了登录，如果没有则会直接跳转到登录页面去，这一模块的基本功能与使用流程如图 3-4 所示:

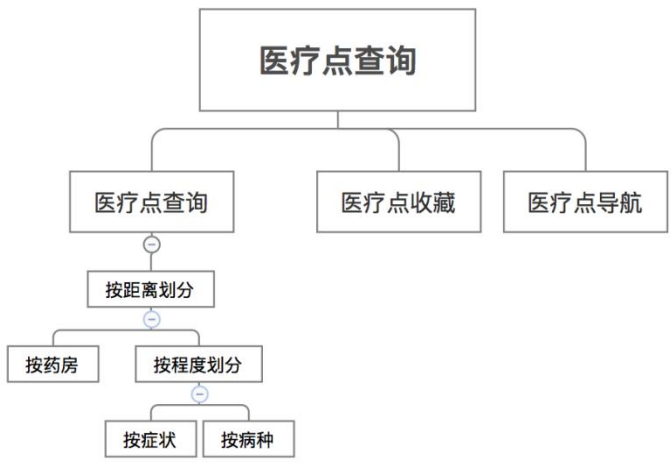

图 3-4 医疗点查询功能与流程图

用户进入这一视图时地图会进行初始化，地图范围为用户所在城市，Marker 点位

默认为城市中心。初始化结束后用户可以进行定位，定位有两种形式：根据 Geolocation 获取结果定位和用户手动拖动地图上的 Marker 来标识个人位置。确定个人位置无误后用户可以进行医疗点的查询，查询分为三种类型：药房、症状和病种。这里的类型和范围都是固定的，具体的查询参数可以由用户进行选择，点击确定按钮后进行查询，同时会向后端发送一条用户查询记录以进行后期数据分析。查询完成后查询结果会在地图上进行标出，用户如果对本次查询结果比较满意可以进行收藏，这样就会添加到用户个人的医疗点收藏列表中，后期可以再次执行相关查询条件。如过想到某一查询结果地点去，则可以点击相关位置，这时会弹出提示'已成功获取该点地理坐标，可以进行导航'。导航同样分为三种:步行导航、公交导航和驾车导航。选择后会规划出相应的最好的路线以供用户选择。

3)老年活动室模块

这一模块主要提供了两个功能：显示活动列表和添加新的活动。起初提出这样的需求的原因是：给中老年人一个认识周边志趣相投的同龄人的平台，大家可以在平台上发布想要组织的活动，有共同爱好就可以凑到一起。用户首先看到的是按时间顺序排列的发布活动，同时可以点击右上角的添加来发布新的活动。这里需要强调的是新发布的活动并不会立刻显示到已有列表中，而是需要后台审核通过后才会显示出来。

4）我的模块

我的模块实现功能如图 3-5 所示。主要分为登录前和登陆后两块，在进入这个页面时会先进行判断，看用户是否已进行了登录，如果没有则会跳转到用户账户模块去，这一模块主要负责的功能是用户登录和注册。注册时除需要填写常见的用户名，昵称，密码，用户手机账号外还有性别、出生年份，个人爱好和疾病史。填写出生年份的目的是在于能够动态的计算用户当前年龄，个人爱好和疾病史则是为了后期实现的推送功能做准备，这样可以做到消息的定向投放，效率更高。用户登陆后会看到个人信息的展示，这里没有提供头像上传功能，只是根据性别设定了两个默认头像，这里同时会显示用户名信息，这些都是在用户登录成功后由后端返回的数据，记录在了 localStorage 和 conf 全局变量中，以便在各个视图中都可以用到。密码修改功能比较常见，除了自行修改外可以联系开发人员进行后台修改。这里还提供的功能是查看用户已收藏的医疗点查询结果，这里可以显示用户的查询时间，查询类型，查询范围及当时自己设定的备注，如果有需要还可以进行二次查询，方便用户复现当初的查询内容。

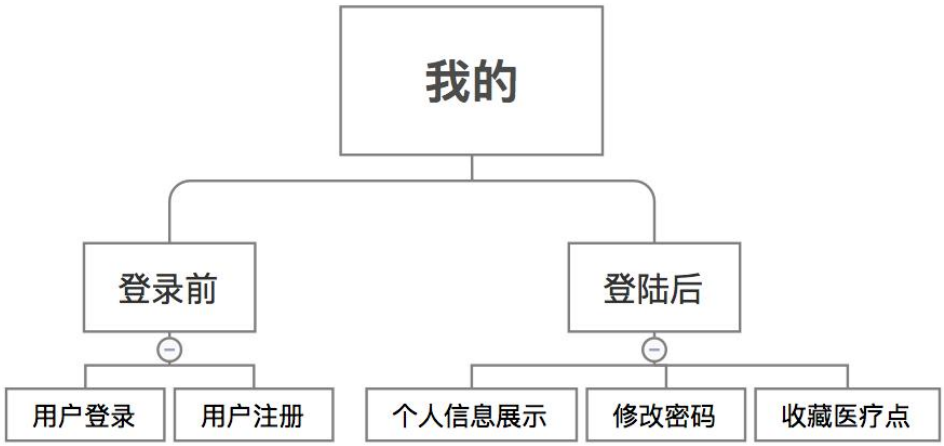

图 3-5 我的模块功能展示

3.2.5 后台模块与界面设计

后台模块划分主要如图 3-6 所示分别为登录、用户统计、推送文章和活动审核与发布四个模块，由于登录模块没有附加的功能，所以接下来会依次对后三个模块进行分析。在这其中所有的数据展示都做了分页设计，同时也可以根据用户的需求返回全部数据结果。

图 3-6 后台管理界面模块展示

1）用户统计

用户统计模块主要有两个功能：一个是注册用户展示，另外一个是查询统计分析功能。注册用户展示视图主要是通过列表的形式按照时间顺序来列出全部注册用户，这其中会展示的内容有：用户名、昵称、年龄、性别、联系电话、兴趣爱好和疾病史，这一部分提供了用户信息修改和删除用户功能。

查询分析功能是整个后台管理系统中的核心功能，这里会负责汇总用户每次查询时的记录，记录的数据有：查询时间、查询类型、查询详情、查询程度和经纬度。这里可以对显示的数据进行筛选，比如过去一天，一周，一月的相关数据或展示出全部的数据，管理人员可以选择其中的部分数据或全部数据将其展示到地图上，当数据量积累到一定程度后可以做时空相关性分析，比如：在过去一月内咽喉症状查询情况激增同时地理位

置集中在邯郸市西北部地区，这时就可以根据这里反应的情况来查找这种情况出现的原因，是地形因素还是天气原因，与周围的工厂有没有关系，同时可以将获取的数据进行导出，利用更专业的地理分析软件做更精确的相关性分析。根据已有的一些材料和文件分析得出地理信息分析难得不是信息分析过程或公式导出，而是没有需要的数据或数据难以获取，而这次毕业设计可能就提供了一个新的思路来解决这一问题：通过设计既能收集数据又能服务调查对象的应用来实现相关信息的获取。

2）推送文章

这一模块主要是针对首页的推送文章所设计的，这里可以显示已发布的文章列表，查看文章详情，对文章进行更新修改以及删除。发布文章模块中现在仍存在这样两个问题：一是没有提供图片上传，需要键入的是其他网站的外链地址，出于本地存储不便，当文章发布管理员发布到本地后还要转到其他类似七牛等云存储平台去，虽不是那么便捷但是节省了存储空间和图片管理费用；而是出于多端多尺寸设备统一显示相同的目的并没有使用富文本编辑器，而是使用了 textarea 来将每个段落分隔开将整篇文章存储为数组的形式，便于移动端文章详情视图的展示。

3）活动审核与发布

这一模块主要是针对老年活动室的活动发布所设计的，这里可以显示已发布活动列表，同时可以发布新的活动。活动列表显示的信息有：题目、发布日期、活动日期、发布人及是否通过审核，这里可以对活动详情进行查看并对内容进行审核，审核通过后就会出现在移动端的活动列表上。发布活动时会自动记录当前时间并存储在该活动的数据结构中，以便于后面进行排序显示。

3.3 网站结构与 Require.js Backbone 使用

3.3.1 网站结构

基本的网站结构如图 3-7 所示。

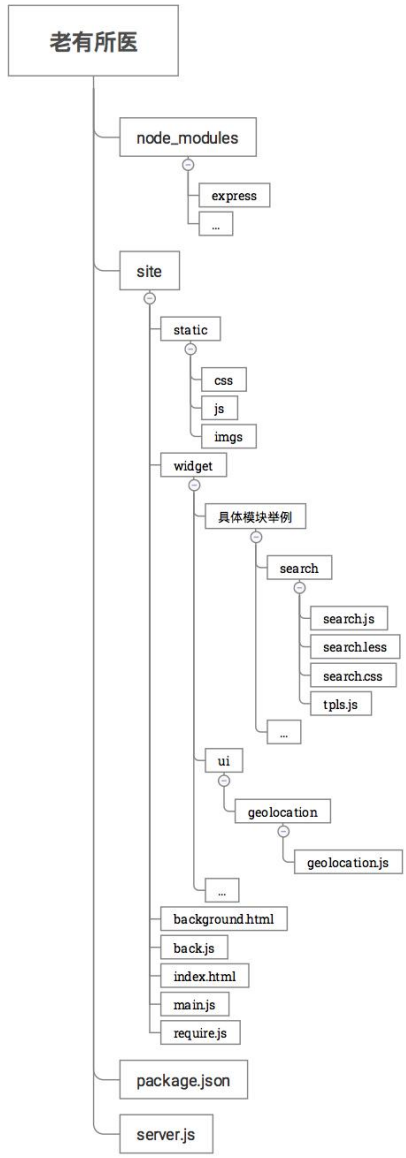

图 3-7 网站基本结构

接下来对每一块内容做基本的内容介绍：

node_modules：用于存放 Node.js 使用的 npm 工具包；

package.json：用于记录当前 npm 包的依赖项，使用 npm install 命令后会自动加载相关包依赖；

server.js：node 后端服务器、接口定义及数据库连接相关内容；

site：前端静态资源文件夹；

background.HTML：后台管理系统入口；

back.js：后台管理系统的 require 依赖项定义；

index.HTML：移动端入口；

main.js：移动端的 require 依赖项定义；

Require.js：Require.js 模块依赖加载；

static：存储了网站基本的 js、CSS 和 images；

widget：分为 ui 通用模块和具体模块两种，存放了所有的组件模块；

ui：存放了通用 ui 组件，比如 Geolocation help 组件等，也包括基础库如 Backbone.js jQuery 等；

search：作为例子的具体模块，即医疗点查询模块，其中的 search.js 模块定义了 Backbone 视图，模型以及页面上的相关事件绑定，search.less 是 search 模块的样式定义，而 search.CSS 是其编译后的版本，tpls.js 是其相应的前端模板。

3.3.2 Require.js 使用实现模块化加载

无论是移动端还是后台都是只在页面上引入了 Require.js 这一个 js 文件，其他 js 都是通过它来进行依赖加载的，下面以移动端为例来展示 Require.js 的使用方法及其便利之处。通过 main.js 来定义基准路径和依赖的前端基础库：

```
(function(){
    require.config({
        baseUrl:'widget',
        paths: {
            jquery:'ui/jquery/jquery-2.2.2.min',
            template:'ui/template/template',
            underscore:'ui/underscore/underscore',
            Backbone:'ui/Backbone/Backbone',
        },
        shim: {
            'underscore': {
                exports: '_'
            },
            'jquery': {
                exports: '$'
            },
            'template': {
```

```
                exports: 'T'
            },
            'Backbone': {
                deps: ['underscore',  'jquery'],
                exports: 'Backbone'
            }
        }
    });
    require(['Backbone',  'underscore',  'ui/router/router'],  function(){
        Backbone.history.start();
    });
    })(window);
```

这里定义了引用的基础库 jquery art-template underscore 和 Backbone，并声明了 Backbone 依赖于 jQuery 和 underscore，这样可以保证按照正确的加载顺序进行。后面的 require 方法是加载并运行了我们在基础路径 widget 下的 Backbone 路由 router，这使得网站整体得以运行，Backbone 的运行机制会在下一节中进行讲述这里就不展开了。

下面通过‘我的’这一模块定义来展示 require 的模块定义方法：

```
define(['Backbone','template',  'my/tpls','ui/helper/helper'],function(Backbone,  T,
tpls,  helper) {
    var View = Backbone.View.extend({
        ...
        ...
    });
    return View;
});
```

通过 define 方法来实现某一模块的定义，第一个参数为依赖的或需要使用的其他模块项，第二个参数为对该模块的定义，这里同时暴露出了需要使用的四个对象，可以直接在模块中进行使用。在模块的最后 return 了 View 对象以供外部调用。通过这样的模块化加载和依赖方法使得我们在定义 js 时可以分模块进行，每一模块内容的耦合性更低复用程度变高，可以把更多的公用组件抽取出来作为公共组件，而将具体的业务逻辑

放在相关的业务组件中进行。

3.3.3 Backbone.js 使用与 art-template

Backbone.js 的使用时构建整个 WebApp 的核心，这里举一个虽然简单但具有代表性的模块首页-列表页及 router 路由来说明它的几大特点和真正运用到项目中的内容，一下为 index/list/list.js 中的内容：

```
var $canvas = $(document.body).find('#canvas');
```

这里获取了 Backbone 视图切换的容器 canvas，后续的视图渲染和切换都是在这个 Dom 容器中进行的。

```
var Model = Backbone.Model.extend({
    defaults: function() {
        return {
            list:null
        }
    },
    getList: function() {
        var that = this;
        $.ajax({
            url: '/artical',
            type: 'GET',
            dataType: 'json'
        })
        .done(function(res) {
            if(res.ret ===1){
                that.set('list', res.list);
            }else{
                alert(res.msg);
            }
        })
    }
});
```

这里定义了当前 View 中用到 Model 对象，Model 通常是与后端服务器进行交互的数据处理对象，从这里就可以看出使用 Ajax 方法向后端发起了请求，当请求成功后将获取的 list 数据赋值给了 Model 对象，defaults 属性是用来设置一些默认属性和值得，这样使得当获取数据时某些字段出现问题也会有默认值来填充。

```
var View = Backbone.View.extend({
    tagName: 'div',
    model: null,
    className: 'index-list',
    events: {
        'click .list-item': 'actInfo'
    },
    initialize: function() {
        this.model = new Model();
        this.renderSkeleton();
        this.renderHeader();
        this.listenTo(this.model,  'change:list',  this.renderList);
        this.model.getList();
    },
    renderSkeleton: function() {
        this.$el.HTML(T.compile(tpls.skeleton));
        $canvas.append(this.$el);
    },
    renderHeader: function() {
        this.$el.find('.main .header').HTML(T.compile(tpls.header)({ title: '老有所医
' }));
    },
    renderList: function() {
        var that = this;
        var list = that.model.get('list');
        this.$el.find('.main .list').append(T.compile(tpls.list)({ list: list }));
```

```
        },
        actInfo: function(e) {
            var id = $(e.currentTarget).data("id");
            Backbone.history.navigate('index/info?id='+id，{trigger: true， replace:
false});
        }
    });
```

这一部分进行了视图的定义，同时向视图中引入了事前定义好的 Model 对象，events 和 this.listenTo 方法都是继承自 Backbone 中 Event 对象的，两者都负责进行相关事件的绑定，不同的是前者负责视图元素上事件的绑定，比如 click dblclick 等事件，tagName 指当前视图的标签类型，默认为 div。initialize 方法是视图被加载时会自动触发的的方法，通常是进行数据的过去和基本页面的渲染，页面的渲染和填充借助了 art-template 这一前端模板，前面引入了 index/list/tpls 这一模板内容，里面是拼接好的字符串。通常将从后端获取好的数据接触 template.compile(tpls.版块名)(data)来进行渲染，data 为从后端获取的数据。

```
var App = Backbone.Router.extend({
        routes: {
            'index': 'pageIndex'，
            'index/info': 'pageInfo'，
            ...
            ...
        },
        currentView: null，
        mainNav: null，
        initialize: function () {
            this.mainNav = mainNav;
        },
        clean: function (options) {
            if (this.currentView) {
                if(this.currentView.close) {
```

```
                this.currentView.close();
            }
            this.currentView.remove();
            this.currentView = null;
        }
        var defaultOptions = {
            hideMenu: false，
            nav: 'index'
        };
        options = $.extend(defaultOptions，  options);
        this.hideMenu = options.hideMenu;
        if(this.mainNav) {
            if(options.hideMenu) {
                this.mainNav.hide();
            } else {
                this.mainNav.show(options.nav);
            }
        }
    },
    getQuerys: function(queryString){
        ...
    },
    pageIndex: function(queryString){
        var that =this;
        var querys = that.getQuerys(queryString);
        require(['index/list/list']，function(view){
            that.clean({
                hideMenu: false，
                nav: 'index'
            });
```

```
                that.currentView = new view({querys:querys});
            });
        },
        ...
        ...
```

以上为 router 路由中截取的一部分代码，之前介绍过老有所医 WebApp 的实现方式是单页面应用，因此并没有页面的跳转而只是视图的切换，视图间切换的匹配就是通过路由来实现的，在 routes 中定义需要匹配的 hash 锚点，即#后面的部分，当发现匹配的内容后就执行后面对应的方法例如：http://localhost:4711/#index?name=LiMing 就会匹配 pageIndex 方法，这里就利用了 require 的模块加载机制加载了对应的 list 视图同时移除了原有视图将 currentView 设置为了当前视图。

3.4 前端页面与功能展示

3.4.1 移动端页面展示

如图 3-8 所示为用户进入时看到的页面，点击老有所医用户指南即弹出用户指南页图 3-9，而点击任意一篇文章则进入相关文章详情页，如图 3-10，文章详情页之所以选择这样的样式布局是为了提供更纯粹的阅读体验，点击左上角即可返回。

图 3-8 首页显示

图 3-9 用户指南

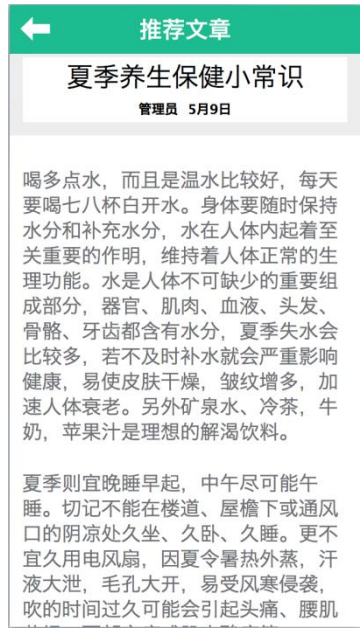

图 3-10 文章详情

如图 3-11 所示为用户进入医疗点查询时看到的页面，此时页面中心默认为当前城市中心并带有一个 Marker 标记，用户可点击左上角定位来获取当前位置，或者直接拖动地图上的 Marker 标记。

图 3-11 进入查询医疗点时页面

用户确定好个人位置或需要查询的位置后即可点击查询按钮如图 3-12，查询条件有： 药房、按症状和按病种三种类型，这样设计的原因是中老年人可能对自己患有的疾病类型可能是无法确定的，但是症状是很显而易见的，所以这个方向上有两种查询方式。确定方式后即可选择查询详情、病情程度和查询范围，完成选择后点击确认。这时同时触发两个事件：查询和向后端发送此次用户查询相关数据。

图 3-12 确定位置后进行搜索

查询完成后会将查询结果显示在地图上如图 3-13，用户可以点击右上角的列表开关 来选择是否显示查询结果列表，点击地图上的每一个 Marker 即可看到相关详情。同时这时显示出收藏按钮，如果用户对当次查询结果比较满意可以加入收藏，后续可以到‘我的’视图中查询已收藏查询记录。

图 3-13 出现查询结果可进行收藏

确定要前往的地点后用户可以点击相关地点，此时弹出提示： 已获取到该点坐标，可以进行导航 ，此时出现导航按钮。点击导航按钮后会弹出导航类型选择：步行导航、公交导航和驾车导航，用户选择相应类型后会进行相关导航，此时地图上会标出相应路线，右上角的列表中也会显示相应的出行方案，如图 3-14 所示，这就是医疗点查询模块的基本功能。

图 3-14 选择目的地可进行导航

用户登陆后的老年活动室如图 3-15 所示，与未登录的区别在于右上角出现了添加按钮。这个页面主要负责已通过审核的活动列表展示，用户可以通过列表上的内容了解相关活动详情，有兴趣的活动可以自行前往参加。

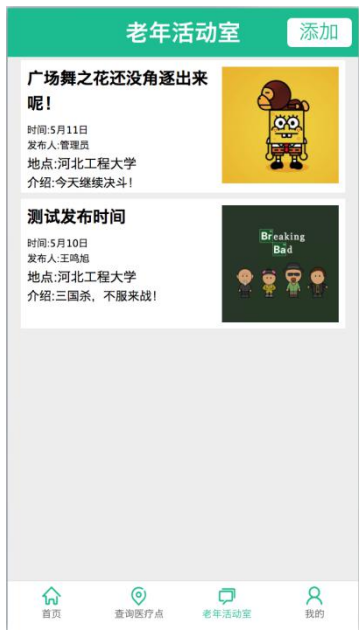

图 3-15 老年活动室

点击添加后进入到添加活动页面如图 3-16 所示。这一页面主要需要用户输入的是：题目、时间、地点和主要内容 ，发布人为自动从用户信息中获取，无法进行更改。用

户确认输入结果无误后即可点击右上角的发布按钮，此时会弹出提示：‘发布成功！通过审核后就会显示啦~ ’，此时用户即可等待后台工作人员进行审核。

图 3-16 添加活动

如图 3-17 所示为用户登录后看到的我的页面，由于登录、修改密码和注册页面设计和功能比较简单就不再进行单独展示了。这一页面中右上角为 登出 按钮，顶部显示了默认头像和用户昵称，下方列有收藏医疗点和修改密码两项功能，点击后分别进入相关页面。

图 3-17 我的视图

如图 3-18 为点击收藏医疗点后显示的页面。这一页面显示的内容为按时间由近到远的用户收藏的医疗点查询记录，分别显示了当时的查询时间，类型，具体情况，范围，程度状态和用户个人设置的备注。用户可以点击每一条记录右上角的再次查询来再次复现当时的查询效果。

图 3-18 我的收藏

3.4.2 后台页面展示

由于登录页面和首页比较的简单，所以后台页面的介绍主要从用户统计、文章发布和活动发布三个视图进行介绍。如图 3-19 所示为用户信息视图，这里展示所有的注册用户信息，同时可以进行用户密码修改和删除用户的操作。

图 3-19 后台用户信息视图

如图 3-20 所示为查询统计分析视图，这里用来展示用户的查询记录信息，显示的内容可以针对时间进行筛选，同时可以将选择结果展示到地图上做初步的时空分析，了

解所选择时间段内的疾病情况。数据显示在默认情况下进行了分页操作，每页显示 10 条数据。

图 3-20 查询分析统计视图

如图 3-21 为文章列表视图，主要用来列出已发布的文章，点击查看按钮可以看到相关文章详情，同时可以实现更新，删除两项操作。

图 3-21 文章列表视图

如图 3-22 所示为推送文章发布视图，这里可以发布新的文章，分别需要输入题目，图片链接地址，摘要和文章正文。这里出于存储方便的目的会提前向管理人员提供七牛云存储的账号以用来存储图片，上传完成后可以获得外链地址；文章正文的实现方式是

出于优化显示格式的目的，所以选择了分段键入而没有提供富文本编辑。

图 3-22 新增发布文章页

如图 3-23 所示为活动列表视图，这里会列出已有的活动同时列出该活动相关题目、发布时间、活动时间、发布人及是否已通过审核。点击查看按钮可以实现了解活动详情，编辑活动及活动审核功能。

图 3-23 活动列表视图

如图 3-24 所示为活动发布视图，这里需要输入活动题目、时间、是否通过审核、活动类型、地点及内容。输入完成后点击右上角的发布即可发布成功。

图 3-24 新增活动

3.5 数据库设计

数据库之所以没有选择关系型数据库而是选择 NoSQL 中的 MongoDB 是出于以下几点目的的：

1）没有像银行系统那样高的同步性要求。做为 WebApp 这种在线服务类 App，对数据的同步性并不高，用户可以接受短暂的数据不同步情况的出现；

2）WebApp 中涉及的数据结构简单，并没有复杂的关联表查询。关系型数据库的一个优点在于能够很好地实现关联表间的查询服务，但是在这款 WebApp 中并没有这样的需要，都是独立的表或者说集合，因此选择 NoSQL 更合适；

3）考虑到当应用进行推广后可能出现的大量用户同时查询的高并发情况出现，NoSQL 在这方面上更加擅长并且速度更快，因为 MongoDB 采取的是 BSON 的存储格式，查询时是通过 key-value 的方式来进行，效率更高实现更好；

4）MongoDB 虽然底层是通过 C++来实现的，但是操作语言上选择了 JavaScript，因此对整个系统的操作和编写上提供了更多的便利。

整个数据库中总共有五个集合结构，下面依次对每个集合的字段进行介绍。

1）管理账户 adminaccounts

管理账户集合管理了后台管理工作人员的相关账号，以下为该集合定义：

```
AdminAccountSchema = new Mongoose.Schema({
    username: { type: String，  unique: true }，
    password: { type: String }，
    name: { type: String }
});
```

这里只定义了用户名，密码和昵称三个字段，其中要求用户名字段为唯一。

2）用户账户 accounts

用户账户管理了用户相关信息，这是五个集合结构中相对比较复杂的一个，以下为该集合定义：

```
AccountSchema = new Mongoose.Schema({
    username: { type: String，  unique: true }，
    password: { type: String }，
    name: { type: String }，
    gender: { type: String }，
    birth_year: { type: Number }，
    phone: { type: Number }，
    hobbies: { type: [] }，
    disease: { type: [] }，
    collections:{type:[]}
});
```

用户账户集合定义了用户名，密码，昵称，性别，出生年份，联系电话，爱好，疾病史和收藏列表共九个字段。这其中前六个字段为简单类型(String 或 Number)，后三个为复杂类型 Array。这里体现出了使用 MongoDB 的好处在于：与 JavaScript 本身的数据结构是相同的无需做另外的转换，存储和读取都更为方便。

3）推送文章 articals

推送文章管理了后台发布的推送文章相关信息，以下为该集合定义：

```
ArticalSchema = new Mongoose.Schema({
    title: { type: String}，
```

```
    date: { type: String },
    author:{ type: String },
    summary: { type: String },
    artical:{type:[]},
    img_src:{type:String}
});
```

推送文章集合定义了文章题目，发布日期，发布人，概要，文章正文和图片外链地址六个字段，这其中 artical 字段为 Array 类型是由于文章是按段落进行存储的，读取时只要将数组传递给前端就可以依次取出每一个段落并置入相应的位置。

4）活动 activities

活动集合管理了由后台发布的或者由用户发布相关活动的信息，以下为该集合定义：

```
ActivitySchema = new Mongoose.Schema({
    title: { type: String},
    date: { type: String },
    locale_time:String,
    author:{ type: String },
    summary: { type: String },
    type:{type:String},
    site:{type:String},
    show:{type:Number}
});
```

活动集合定义了活动题目、活动时间、发布时间、发布人、活动内容、活动类型、活动地址及是否通过审核八个字段，这其中 locale_time 字段为活动发布时自动获取的当地时间，show 字段为是否通过审核。

5）用户记录 records

用户记录集合管理了用户进行查询时向后端发送的相关数据，以下为该集合定义：

```
RecordSchema = new Mongoose.Schema({
    user_id:String,
    date:Number,
```

```
        locale_time:String，
        position:{
            lng:Number，
            lat:Number
        }，
        condition:{
            types:String，
            detail:String，
            level:String，
            area:String
        }
});
```

用户记录集合定义了查询用户 id、查询时间(时间戳)、查询时间(当地时间)、地理定位信息和查询条件五个字段，这其中 position 和 condition 字段又是两个复杂的对象结构。position 中记录了查询位置的经纬度坐标，condition 中记录了查询类型，查询详情，程度等级和查询范围。

3.6 后端服务器搭建与接口设计

3.6.1 服务器搭建

Web 服务器一般指网站服务器，是指驻留于因特网上某种类型计算机的程序，可以向浏览器等 Web 客户端提供文档，也可以放置网站文件，让全世界浏览；可以放置数据文件，让全世界下载。常见的 Web 服务器 Apache、Nginx 和 IIS，而在 Linux 平台上使用最广泛的是 Apache 和 Nginx。Express 是一个基于 Node.js 平台的极简、灵活的 web 应用开发框架，它提供一系列强大的特性，帮助你创建各种 Web 和移动设备应用;通过 Express 搭建的 Web 服务器无需依赖其他特定的 Web 服务器软件(如 Apache、Nginx)，老有所医 WebApp 就是基于这个框架搭建了简单的后端服务器，搭建服务器只需要简单的几行代码，如下所示：

```
var application_root = __dirname，
    express = require('express')，
```

```
        path = require('path')，
        Mongoose = require('Mongoose')，
        crypto = require('crypto');
    var app = express();
    app.configure(function() {
        app.use(express.bodyParser());
        app.use(express.methodOverride());
        app.use(express.static(path.join(application_root， 'site')));
        app.use(express.errorHandler({ dumpException: true， showStack: true }));
    });
    var port = 4711;
    app.listen(port， function() {
        console.log("Express server listening on port %d in %s mode"， port，
app.settings.env);
    });
```

3.6.2 接口设计

接口风格设计基本按照 RESTful 接口风格规范来进行，涵盖了 GET、POST、UPDATE 和 DELETE 四种基本类型，但考虑到国内环境因素，因此将后两种统一到了 POST 类型中，只从参数上进行判别。下面为接口列表：

表 3-1 接口列表

接口功能	类型	地址	参数及说明
用户注册	POST	/account/register	Object 用户相关信息
用户登录	POST	/account/login	Username password 用户名及密码
管理员登陆	POST	/account/admin/login	Username password 用户名及密码
新增收藏	POST	/userInfo/addRecord	Object 查询相关信息
收藏列表	POST	/userInfo/record	Id 用户_id 字段
修改密码	POST	/account/changePassword	Id oPwd nPwd 用户_id 字段 新/旧密码
用户列表	GET	/userInfo	Skip limits 跳过数及每页显示数量
发布文章	POST	/publish/artical	Object 文章相关信息
文章列表	GET	/artical	Skip limits 跳过数及每页显示数量
文章详情	POST	/artInfo	Id 文章_id
删除文章	POST	/artInfo/delete	Id 文章_id

更新文章	POST	/artInfo/update	Object 文章相关信息
发布活动	POST	/publish/activity	Object 活动相关信息
活动列表	GET	/activity	Skip limits 跳过数及每页显示数量
活动详情	POST	/activity/info	Id 活动_id
删除活动	POST	/activity/delete	Id 活动_id
更新活动	POST	/activity/update	Object 活动相关信息
用户记录	POST	/record	Object 用户记录相关信息
记录列表	GET	/record/search	Skip limits 跳过数及每页显示数量

第四章 总结与展望

在整个毕业设计的制作过程中主要分为了四个过程：需求了解，程序设计，部署开发，论文撰写。在需求了解过程中首先通过个人理解分析了要实现的核心功能，针对中老年人需要进行的功能优化点，后期与身边的中老年人进行了多次的沟通，只有真正了解切实的需要才能做出有用的设计而不只是空谈；程序设计阶段经历了技术选型，技术调研，应用技术调整这样几个阶段，大部分时间花在了技术调研上，通过这点可以得出的经验是只有充分了解所使用的技术才能在后期开发过程中明确能实现什么和需要怎样去实现的问题；部署开发过程主要是进行了页面设计，前端实现，数据库设计和后端接口设计这几项工作；最后在论文撰写过程中不断对所做进行了反思和优化，弥补了一部分在最初设计时考虑不周的地方。

论文研究的具体工作和成果主要表现在以下几个地方：

1）了解毕业设计相关技术的发展情况及国内外研究现状，在开始设计工作之前有了初步的宏观认识；

2）对开发过程中使用的技术进行了调研与介绍，这其中主要包括：WebGIS 相关概念和原理、HTML5 中的 Geolocation API、高德地图 JavaScript API、前端核心技术 HTML、CSS JavaScript 基本应用、Web2.0 与单页面应用、jQuery 库、Require.js 模块化加载、Backbone.js 前端 MV*框架、art-template 前端模板、Less—CSS 预编译语言、服务器端运行的 JavaScript—Node.js 和 NoSQL 与 MongoDB；

3）根据上述技术进行了 WebApp 的设计与实现：通过 Backbone.js 实现前端视图结构搭建，Require.js 实现按需模块化加载，使用 Node.js 搭建了后端服务器并设计了相关接口，利用 MongoDB 来存储相应数据；

4）分析了 Backbone 存在的一些缺陷与不足，利用提出 currentView 的概念解决了视图切换时的内存占用问题，使用 queryString 的方式优化了页面间参数传递的问题；

5）借助 localStorage 实现了浏览器端的用户登录记录与信息保存机制；

6）提出了一种新的获取地理信息与用户数据的方式：通过设计实现 WebApp 并推广来主动获取相关信息，积累一定的数据后进行相关分析。

同时对系统进行分析和研究后发现存在以下部分缺点和不足：

1）在发布项目时没有对 js 和 CSS 文件进行压缩处理，这可能会浪费一部分用户流量；

2）文章发布功能没有进行优化，按段落发表文章操作复杂，不利于使用；

3）暂未开发相关图片上传功能，后期需要进行优化；

4）安全性没有进行考虑，相关数据传输过程中没有进行加密，可能存在安全性隐患。

如今 Web 技术还在不断发展，WebGIS 借助这样的一个平台也在不断改进和优化。只有不断更新个人技术能力，了解时代发展现状才能保持自己的先进性，相信 WebApp 在日后的应用领域可以占据一席之地，定能够与原生应用相媲美。

参考文献

[1] 黄成礼，庞丽华.人口老龄化对医疗资源配置的影响分析[J]. 人口与发展. 2011(02)，2011.

[2] 黎楚湘. 中国老年人卫生服务需要与利用研究[D]. 复旦大学，2006.

[3] 熊伟. 地理空间信息技术应用的发展条件研究[J].《测绘与空间地理信息》2012,35(8),2012.

[4] 宋颖. 北京市城镇无保障老年人医疗保障问题研究[D]. 首都经济贸易大学，2009.

[5] 朱吉鸽 刘晓强. 国外老年医疗保障体系进展与启示[J]. 《国外医学：卫生经济分册》2008，25(3)，2008.

[6] 陈勇. Web App 现状分析及展望[J]. 《通信与信息技术》2012，(4).2012.

[7] 宋关福，钟耳顺，王尔琪. WebGIS—基于 Internet 的地理信息系统[J]. 中国图象图形学报. 1998(03)，1998.

[8] 林延平. 空间要素服务器（WFS）关键技术研究[D]. 北京工业大学，2004.

[9] 郭建明. 基于移动开发平台的机场办公应用系统研发[D]. 大连海事大学，2013.

[10] 张赵辉. 探析网页前端技术及其升华[J]. 《科技致富向导》2015，(2)，2015.

[11] 赵驼. 基于 Skyline 的三维景观系统的构建[D]. 东北林业大学，2013.

[12] 沈昕. 基于 Node.js 及 Mongodb 的在线学习测试系统设计[J]. 《无线互联科技》2015，(4)，2015.

[13] 孙中廷. 基于 NoSQL 数据库的大数据存储技术的研究与应用[J]. 《计算机时代》2014，(7)，2014.

[14] 郭匡宇. 基于 MongoDB 的传感器数据分布式存储的研究与应用[D]. 南京邮电大学，2013.

[15] 江民彬. 非关系型与关系型空间数据库对比分析与协同应用研究[D]. 首都师范大学，2013.

致谢

时光飞逝，刚入学的情景还历历在目，转眼间大学四年已经过去，我的学生生活也渐渐进入了尾声。四年的大学生活不仅让我丰富了文化知识，增长了阅历和见识，也让我改变了很多，成长了很多。我不再像以前一样毛手毛脚，遇到事情慌不择路，而是变得多了一份自信和稳重，这些都是其他东西所无法比拟和替代的，是我人生最宝贵的财富。

四年的大学生活中，我不仅有家人一直的鼓励和支持，还遇到了让我尊敬的老师，友好的舍友和一群乐于助人的朋友，我十分感谢他们一直以来对我的陪伴和帮助。

这次的毕业设计是在王冬利老师的指导下完成的，要感谢王老师的指导和帮助。从最初的选题、技术的选择、程序结构设计和论文的编写，整个毕业设计完成过程中的各个环节我都遇到了很多的问题，但王老师都耐心的帮我寻找原因并解决问题。他会给我提供相关设计方法和思路，找出我存在的问题和不足，经常关心我的毕设进度，投入了他大量的精力，对我毕业设计的完成提供了很大的帮助。所以在毕业设计完成的时候，我要向王冬利老师表示衷心的感谢。

感谢地理信息系统专业的老师们，大学四年的学习里总是耐心解答提出的问题，教会了我很多知识和道理，正是有了这些知识作为理论基础，才能是我的毕业设计能顺利完成。

感谢地理 1202 班的全体同学，四年的大学学习正是因为有了他们才变得不一样，这次的毕业设计过程中，有很多问题都是在他们的帮助下才解决的，我要感谢他们一直以来对我的帮助。

感谢同宿舍的兄弟们，四年以来我们一直和谐相处，创造了良好的宿舍气氛。生活中的困难都是在他们的帮下克服，他们对我的帮助不是只言片语能概括的，真心感谢他们。

最要感谢的是我的家人，除了二十多年的养育，他们给予我更多的是任何东西都不能替代的亲情的关爱，无论我做什么决定，他们永远都会在背后给我莫大的支持和鼓励，让我在学习和生活中有了努力向前的信心和决心，感谢他们。

感谢所有帮助过我的老师、亲人、同学和朋友。