计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(2):52−62 [doi: 10.15888/j.cnki.csa.007791]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Node.js 与 REST 风格的医保定点零售药店履约考核系统①

许思特[1], 黄子硕[2], 马振凯[1], 吴　斌[3], 刘佳兴[4], 盛　韬[3], 戴瑞明[1], 罗　力[1], 张天天[1]

[1](复旦大学 公共卫生学院, 上海 200032)
[2](同济大学 土木工程学院, 上海 200092)
[3](复旦大学 计算机科学技术学院, 上海 201203)
[4](复旦大学 软件学院, 上海 200082)
通讯作者: 张天天, E-mail: 18111020033@fudan.edu.cn

摘　要: 为客观评估医保定点零售药店的履约情况, 为医保经办部门的监管工作提供信息化的管理手段, 本研究研发了一种医保定点零售药店履约考核系统. 根据医保定点零售药店设置标准, 基于 Node.js 与 REST 风格, 采用 Vue.js 与 Django Rest Framework 框架, 构建评估系统. 所建系统完成上海市 881 家医保定点零售药店履约考核, 完成全市定点药店履约数据库的构建, 并出具定制化的考核结果报告. 本系统使零售药店考核相关管理工作变得简单便捷, 全市考核周期由 4 周缩至 1 周, 减少行政人力 128 人月/年. 为药店管理者、专家考核组和行政监管者提供了工作辅助与决策依据, 满足各方信息化需要, 助力医保服务更好惠民利民.

关键词: 医保监管; 零售药店; 考核系统; Vue; Django; RESTful

引用格式: 许思特,黄子硕,马振凯,吴斌,刘佳兴,盛韬,戴瑞明,罗力,张天天.基于 Node.js 与 REST 风格的医保定点零售药店履约考核系统.计算机系统应用,2021,30(2):52–62. http://www.c-s-a.org.cn/1003-3254/7791.html

Evaluation System of Medicare-Designated Pharmacy Based on Node.js and REST Style

XU Si-Te[1], HUANG Zi-Shuo[2], MA Zhen-Kai[1], WU Bin[3], LIU Jia-Xing[4], SHENG Tao[3], DAI Rui-Ming[1], LUO Li[1], ZHANG Tian-Tian[1]

[1](School of Public Health, Fudan University, Shanghai 200032, China)
[2](College of Civil Engineering, Tongji University, Shanghai 200092, China)
[3](School of Computer Science, Fudan University, Shanghai 201203, China)
[4](School of Software, Fudan University, Shanghai 200082, China)

Abstract: A performance evaluation system is developed for medical insurance departments to objectively supervise the medicare-designated pharmacy. According to the standard set of the medicare-designated pharmacy, Vue.js and Django Rest Framework are selected to build an evaluation system in Node.js and REST styles. The system has evaluated the performance of 881 medicare-designated pharmacies in Shanghai. It builds a database of Shanghai's pharmacy performance and issues a customized assessment report. The system makes the management and assessment of the medicare-designated pharmacy more efficient, reducing the supervision cycle from 4 weeks to 1 week and administrative manpower by 128 person-months/year. As an information-based tool, it can support pharmacy managers, expert assessment teams and administrative supervisors and help medical insurance services better benefit people.

Key words: health insurance supervision; medicare-designated pharmacy; evaluation system; Vue; Django; RESTful

① 收稿时间: 2020-06-25; 修改时间: 2020-07-27; 采用时间: 2020-07-31; csa 在线出版时间: 2021-01-27

近年来, 国家简政放权治国理念逐步落实, 对基本医疗保险定点医药机构的管理方式由“行政审批”转向“全面实施协议化管理”. 协议规定, 基本医疗保险定点零售药店 (以下简称“定点药店”) 只要符合基本的医保准入条件就可取得相应的医保定点经营资格. 该项政策的实施, 提升了药店医保资格的审批速度, 全国药店经营者反响积极, 全国定点药店的数量呈现成倍增长的趋势.

长期以来, 医保零售药店以数量多、连锁率低、骗保行为层出不穷等, 成为医保基金潜在的威胁和监管难题. 为解决这一医保定点机构管理领域的症结, 国内各地都做出了相应尝试, 将信息化手段运用于定点零售药店管理. 如西安建立的“医保智能监控审核系统”以及天津市采用的“创新型协议管理考核系统”, 都以信息化手段对医疗服务行为开展了事中监控与事后追踪, 一定程度有效监控了医疗服务行为.

随着信息管理系统的构建推广, 相关问题也随之而来. 如何将科学的机构管理理论、方法与信息技术进行有效结合, 仍是当下亟待解决的难点. 如今的国内医保领域, 相关管理系统就普遍存在着以下两方面的问题.

(1) 随着数据量的几何增长, 所涉及的相关部门、人员不断增多, 管理部门对信息系统的健壮程度、安全性、并发速度等有着越来越高的要求. 而采用传统技术栈构建的软件, 明显无法满足上述要求.

(2) 随着信息领域技术不断革新, 越来越多优秀高效的组件库、插件工具等应运而生. 而采用传统技术栈构建的软件, 受困于自身兼容性, 无法对新的技术进行有效应用, 造成软件自身成长的停滞.

因而, 本文从基本医疗保险的功能出发, 以《上海市基本医疗保险定点零售药店服务协议》为基础, 转化为科学的考核指标. 在充分考虑了操作便捷性、数据安全性、平台可塑性、后续的维护、未来发展性等要求, 摈弃前后端一体的传统网页开发模式, 基于 Node.js 与 REST 风格, 选取 Vue.js 与 Django Rest Framework 框架, 独立开发了医保定点药店履约情况考核评估系统. 实现了自我评议与专家评议功能, 助力医保经办机构贯彻落实参保人员意志, 提高医疗保险日常监管水平, 确保基本医疗保险制度稳健运行的目的.

1 系统需求

鉴于考核内容和考核结果的数据具有一定的敏感性, 医保机构需要建立一套完整、独立、安全的考核系统. 为保证后续可塑性、兼容性、拓展性、以及多平台的联动性, 须重新开发考核系统并持续维护.

本系统平台作为供医疗保险经办机构或其他监管部门使用的工具, 用以辅助开展基本医疗保险定点零售药店履行情况的考核. 适用对象包括药店管理者、专家考核组和行政监管者.

1.1 模块功能需求

主要模块包括, 自评模块与飞行检查模块. 主要模块功能需求如下:

(1) 自评模块, 由定点药店使用.

① 系统需将基础信息、地理信息与数据库联动, 确认后正式进入填报内容.

② 系统需能自动生成完整报告, 方便填报人核查填报内容.

③ 填报内容确认后, 系统自动评星评级, 告知相关人员待整改内容.

(2) 飞行检查模块, 由市、区级管理者使用.

① 可查看所有药店具体填报与评级情况.

② 可对药店失实的填报内容进行更正.

③ 可对考核结果进行图表展示.

④ 可对考核结果进行有针对性的查询, 并将考核结果进行导出.

⑤ 可进行跨区考核评分.

1.2 系统开发需求

针对模块功能需要, 对系统开发的技术点需求归纳如下:

(1) 数据库

① 选择含有友好的地理功能的数据库. 用以满足地理相关信息的操作功能.

② 进行数据库表逻辑设计. 用以满足后续数据累积需要.

(2) 前端实现

① 对框架选取. 选取合适的框架, 用以合理实现系统开发.

② 参数传输. 选择合适的参数传输方式, 用以实现药店填报、管理者修改检查等功能.

③ 统计图表可视化. 实现统计图表的可视化, 用以实现直观展示考核结果功能.

④ 地理信息可视化. 结合地理信息可视化, 用以确认药店地理相关基本信息功能.

⑤ 考核结果表格导出. 方便医保管理者对考核结果进行管理.

⑥ 界面的美化. 用以方便使用者直观便捷地运用本系统.

(3) 后端实现

框架选取. 选取合适的后端, 配合前端合理实现系统开发.

(4) 缓存与状态管理

选取合适的方式对用户缓存与状态进行管理.

(5) 安全性

① 注册时逻辑匹配. 用以解决药店身份确认问题.

② 修改密码时加密. 用以解决用户个人信息安全性问题.

③ 自定义修改密码页. 用以保障修改密码的过程安全.

④ 利用 token 对访问等操作进行校验. 用以解决涉密数据的安全性问题.

最终, 整个系统应实现各数据库的汇总、存储与数据表反馈. 根据测试上传更新的数据库, 通过常模识别方法, 调整基于考核主体的常规模式的更新. 输出多维序列数据的可视化结果, 用于反馈报告. 并对多维序列数据结果与常模情况做提示报告, 识别异常指标. 为自评考核和飞行检查提供了可直观、便捷、安全的信息化平台, 通过量化得分的方式辅助实现了医疗保险协议化管理, 提出干预意见供医保部门参考, 用于有针对性的飞行检查、联合治理或常模算法的调整.

2 系统开发

本研究在进行了充分的文件查阅和探索后, 摈弃了当下大多数考核系统的传统开发模式, 选取了前后端分离的网页开发模式. 原因如下:

常见 Web 应用模式分两种: 一是前后端不分离. 在前后端不分离的引用模式中, 前端页面效果由后端页面渲染或者重定向, 前后端的耦合度很高, 但是后端对接 APP 时, 并不总需要后端返回 HTML 网页, 而仅是数据本身[1]. 所以后端原本返回网页的接口不再适用前端 APP 应用, 为了对接 APP 后端还需再开发一套接口.

二是前后端分离. 在前后端分离的应用模式中, 后端仅返回前端所需要的数据, 从后端请求的数据如何加载到前端中, 都由前端自己决定, 后端仅需开发一套逻辑对外提供数据即可. 在前后端分离的应用模式中,前端与后端的耦合度相对较低. 该模式下, 通常将后端开发的每一视图都成为一个接口,或者 API,前端通过访问接口来对数据进行增删改查[2].

根据不同模式的特点, 我们选取前后端分离模式, 该模式具有以下优势: 前端静态资源与后台 API 分流, 互不影响; 前后台同步开发, 减少沟通成本; 方便开发调试, 不影响工作进度; 易于维护扩展.

本文针对项目需求, 提炼关键技术点进行系统开发, 从数据库、前端、后端、缓存、安全性 5 个方面着手解决对关键技术点的要求.

2.1 数据库

(1) 数据库选用

本履约考核系统的数据库选用了 PostgreSQL, PostgreSQL 是一个功能强大的开源对象关系数据库系统, 它结合了许多安全存储以及扩展复杂数据工作负载的功能, 使用和扩展了 SQL 语言[3].

PostgreSQL 凭借其经过验证的架构, 可靠性, 数据完整性, 强大的功能集, 可扩展性, 使得 PostgreSQL 在所有主要操作系统上运行, 自 2001 年以来一直是符合 ACID 标准的, 并且具有强大的附加功能, 例如流行的 PostGIS 地理空间数据库扩展器[4].

(2) 数据库逻辑设计

基于 PostgreSQL 的特点, 我们对数据库进行了逻辑设计, 见图 1.

authorization_rolecode 表和 shdistrict 表用以确定市级、区级、药店的角色与权限. authorization_user 表用以储存用户注册信息, 包含药店注册信息和市区级信息. 继而根据角色分为自评表 Yd_in 表与区评表 Shdistrict_in 表, 由药店自评填入与区级管理者区评填入. 最终各表汇总于 Yd_show 表.

经验证, 该数据库设计满足巴斯范式, 在正常使用中不存在修改异常、插入异常和删除异常.

2.2 前端实现

(1) Node.js

整个系统前端基于 Node.js 完成, Node.js 的官方说明, 介绍其是一个 JavaScript 的运行环境. 采用了一个事件驱动设计, 并且它是非阻塞式 I/O 模型, 这样使得它不仅变得轻量而且愈发高效. 它的最初设计的目标, 是实现高性能的 Web, 能够高并发地处理网络请求的连接, 快速地进行可扩展互联网应用的搭建.

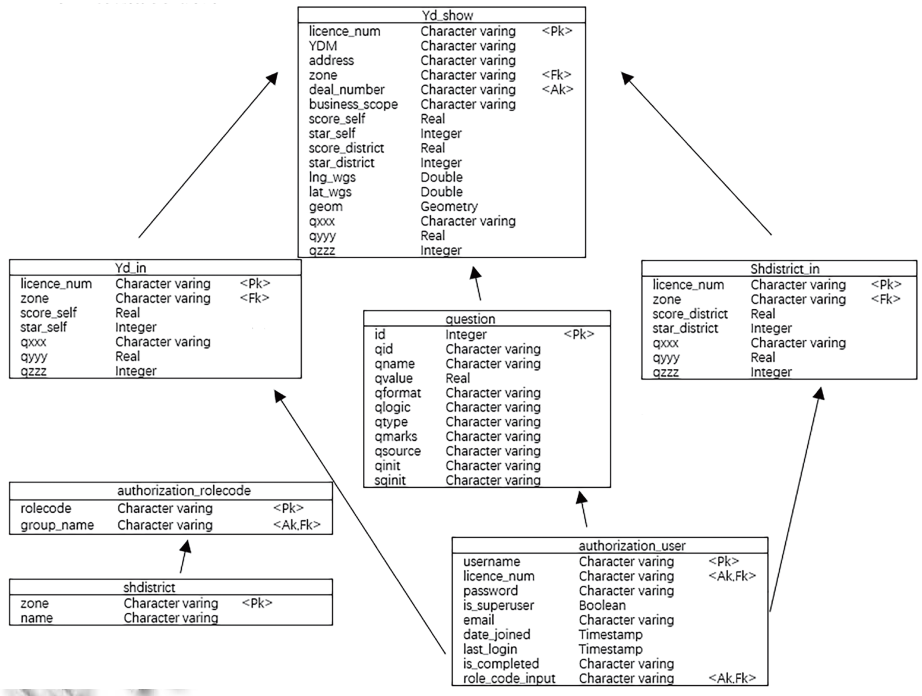

图 1 数据库关系表

(2) Vue 框架特点

基于 Node.js 环境, 本文选择 Vue.js 框架进行开发. 在多个方面, Vue.js 都与其它主流框架有着差异以及优势. 以框架体积为例, 相比较 React 的 43 kb 和 Angular 的 143 kb, 只有 23 kb 的 Vue.js 明显是更为的精简[5].

此外, 组件化是 Vue 的主要特点之一. 在前端开发中, 组件就是视 UI 样式和对应功能为独立的小整体. 无论该小整体在哪个模块被调用, 它都可以具备一样的功能和样式, 从而实现灵活性以及复用性.

Vue 将前端的 HTML+CSS+js 文件都放到一个文件中.vue 里面.

封装格式如下:

HTML (结构): 被 Vue 封装在<template>中,

CSS(样式): 对应<style>,

JavaScript: 对应<script>.

数据绑定是 Vue 的另一大特点. 以往的应用开发中, 前端状态数据、文档结构数据的管理存储是相当复杂和棘手的一个部分. Vue 将底层数据与视图进行对应, 进入页面时会参照 el 属性将挂载的 DOM 元素进行实例化, 转为成 Vue 实例. 数据与文档 DOM 结构绑定在一起, 在数据和结构 UI 之间建立响应式的映射关系. 使用特殊的 v-model 指令, 实现双向绑定关系. 将开发者对 DOM 的操作性大大降低, 避免客户端压力的同时, 将开发者的工作经历重心集中于逻辑的处理.

关键代码如下:

```
<div id="app">
  <input type="text" v-model="message" placeholder="输入…">
  <p>输入的内容: {{message}}</p>
</div>
<script>
  var app = new Vue({
    el:'#app',
    data:{
    message:""
      }
      })
</script>
```

(3) 前后端交互传参

确定前后端分离的开发模式后, 整个系统的药店填报、区级改查、跨区互评大多是基于前端往后端接口传输交互. 以 post 接口为例.

关键代码如下:

```
axios({
  method: "POST",
  url: globalBus.base_local_host+"",
  dataType: "json",
  headers: {
    "Content-Type": "application/json",
  }
```

(4) V-charts 实现统计图

在使用 echarts 进行图表的生成时, 常常无可避免地进行数据类型的繁琐转化、复杂配置项的修改, 而 V-charts 的出现解决了这个问题. 基于 Vue2.0 和 echarts 封装的 V-charts 图表组件, 只需要统一提供一种对前后端都友好的数据格式设置简单的配置项, 便可轻松生成常见的图表. V-charts 已经处理了关于 echarts 依赖引入的问题, 保证所使用的图表, 都是最小的文件.

本文借助 Vue 组件化开发, 封装 echarts 的图表库, 抽象出标题、系列数据、图例、坐标轴、背景颜色、系列颜色、字体颜色等配置信息, 实现可复用的图表组件.

关键代码如下:

```
data() {
this.typeArr = ["histogram", "pie", "funnel", "bar"];
this.index = 0;
return {
  cityData: {
    columns: [""],
    rows: []
  },
  chartSettings: { type: this.typeArr[this.index] },
  };
},
methods: {
  changeType: function() {
    this.index++;
  if (this.index >= this.typeArr.length) {
    this.index = 0;
  }
  this.chartSettings = { type: this.typeArr[this.index] };
  }
```

(5) Leaflet 实现地理定位

Leaflet 是一个为移动设备设计的交互式地图的开源的 Javascript 库, 并只有 38 k, 涵盖了大多数情况下需要的地图特点. 根据药店经纬度坐标返回的 GeoJSON 编码, 利用 Leaflet 插件将药店坐标打点, 并渲染至前端.

关键代码如下:

```
function changePoint(points){
// assert(points.length>=2)
return [points[1],points[0]]
}
import baseinfo2 from '../value/baseinfo2'
export default {
 name: "changeInfo",
data(){
  return {
    center:[31.143 243 4, 121.515 048 3],
    zoom:13,
    map:null
  }
},
methods:{
  load(){
    this.map = L.map("map").setView(this.center, this.zoom);
#导入底层地理图层
  L.tileLayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png").addTo(
   ‘’this.map
    );
#在图层上打点
  L.marker(points).addTo(mapApp)
   .bindPopup("<b>"+name+"</b><br />地图显示").openPopup();
  }
}
```

(6) 表格导出与筛选

下载 Blob.js 和 Export2Excel.js, 在 src 目录下新建 Excel 文件夹, 里面放入 Blob.js 和 Export2Excel.js 两个 JS 文件, 并在 main.js 引入文件.

组建中使用 methods 添加. tHeader 是表头, filterVal 中的数据是表格的字段, tableData 中存放表格里的数

据, 类型为数组, 里面存放对象, 表格的每一行为一个对象[6].

关键代码如下:

```
exportExcelcity() {
import("@/vendor/Export2Excel").then(excel => {
  const tHeader = [""];
  const filterVal = [""];
  const list = this.allyd;
  const data = this.formatJson(filterVal, list);
    excel.export_json_to_excel({
    header: tHeader,
    data
    });
  });
}
```

(7) 界面 UI 美化

选取 Element.UI, 该 UI 库是一套为开发者、设计师和产品经理准备的基于 Vue 2.0 的桌面端组件库, 简约美观, 能满足大多数前端开发者的需求.

关键代码如下:

```
<!-- 引入样式 -->
<link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
<!-- 引入组件库 -->
<script src="https://unpkg.com/element-ui/lib/index.js"></script>
```

2.3 后端实现

(1) 后端框架选取

本系统后端选择 Django Rest Framework 框架. 它是一个强大且灵活的工具包, 用以构建 Web API. Django REST Framework 可以将 Django 为基础, 快速实现 REST 风格 API, 并且自身还带有 WEB 的测试页面, 可以便捷地测试自己的 API.

(2) DRF 应用

Django Rest Framework 的流程大致如下:

建立 Models; 依靠 Serializers, Parse 数据库取出的数据为 API 的数据; ViewSet 是一个 views 的集合, 根据客户端的请求, 进而返回 Serializers 处理的数据; 权限 Premissions 也在这一步做处理; ViewSet 可在 Routers 进行注册, 注册后会显示在 API Root 页上; 在 urls 里注册 ViewSet 生成的 view, 指定监听的 URL[7].

关键代码如下:

```
class YdinViewSet(viewsets.ModelViewSet):
  serializer_class = YdinSerializer
  queryset = YdIn.objects.all()
class YdinUpdateAPIView (generics. UpdateAPIV-iew, mixins. UpdateModelMixin, mixins. Retrieve-ModelMixin, ):
  queryset = YdIn.objects.all()
  serializer_class = YdinUpdateSerializer
```

2.4 缓存与状态管理

Store.js 是一个 localStorage 的包装器, 兼容所有的浏览器, 并不需要借助 Flash 或是 Cookie. store.js 会根据浏览器自动选择使用 userData、globalStorage 或者 localStorage 来实现一个本地存储功能[8].

Vuex 通过 state、mutation、getter 进行状态管理.

Vuex 与 Globalbus 类似, 但在条理性上优于后者. 以将需要用的信息存入 localStorage 中, 并根据不同情况判断是否清除这些信息为例.

关键代码如下:

```
const store = new Vuex.Store({
  state: {
    //几个变量
    //用户
    authUser: {},
    //获得的 jwt
    jwt: localStorage.getItem('token'),
  }
  mutations: {
    setAuthUser(state, {
    authUser,
  }) {
  Vue.set(state, 'authUser', authUser)
  // 保存注册状态
  localStorage.setItem("authUser", authUser);
},
  getters: {
    getAuthUser: state => {
    return localStorage.getItem('authUser')
    },
  }
})
```

2.5 安全性

(1) 注册时逻辑匹配

根据不同情况向前端返回不同的 HTTP 状态码.

注册成功: 若是数据库中有信息的老药店, 则返回 201, 接着进入基本信息确认、地理信息确认, 继而进入填报系统; 若是数据库中没有信息的新药店, 则返回 202, 并直接进入填报系统.

注册失败: 若是基本信息不匹配, 如许可证号与交易代码不对应, 或是用户已注册过, 则返回 401 并在前端 alert; 若是角色不匹配, 则返回 402 并在前端 alert.

关键代码如下:

```
def post(self, request):
# successful registration:
# 201: the pharmacy's information has been stored in YDshow
# 202: the pharmacy's information has not been stored in YDshow
# failed registration:
# 401: the deal number and license number do not match/ the user has already registered
# 402: role code input error(could not find role code foreign key)
user = request.data
serializer = self.serializer_class(data=user)
serializer.is_valid(raise_exception=True)
licence_number = user["username"]
stored = True
# validate deal number
try:
user_exist = User.objects.get(username=licence_number)
return Response(serializer.data, status=status.HTTP_401_UNAUTHORIZED)
except User.DoesNotExist:
pass
try:
# check if the pharmacy's information has been collected in database
user_info = YdShow.objects.get(licence_num=licence_number)
except YdShow.DoesNotExist:
# the database dose not collect the pharmacy's information
stored = False
# validate if the username matched with deal number
# if it has already been in stored in database
if stored and user_info.deal_number != user['deal_number']:
return Response(serializer.data, status=status.HTTP_401_UNAUTHORIZED)
# create user object
try:
# set foreign key
role_code_input = serializer.validated_data['role_code_input']
role_code_instance = RoleCode.objects.filter(role_code=role_code_input)[0]
user_instance = User.objects.create_user(
username=serializer.validated_data['username'],
email=serializer.validated_data['email'],
password=serializer.validated_data['password'],
role_code_input=serializer.validated_data['role_code_input'])
user_instance.roleCode = role_code_instance
user_instance.save()
except IndexError:
return Response(serializer.validated_data,
status=status.HTTP_402_PAYMENT_REQUIRED)
if stored:
# TODO: return geometric information
geo_info = serialize('geojson', [user_info], geometry_field='geom',
fields=('geom', ))
return Response(geo_info, status=status.HTTP_201_CREATED)
else:
return Response(serializer.data, status=status.HTTP_202_ACCEPTED)
```

(2) 修改密码时加密

若用传统 update 接口修改密码, 会造成密码直接暴露的隐患, 因而需要再次序列化加密.

关键代码如下:

```
#set_password also hashes the password that the user
```

```
will get user=User.objects.get(username=serializer.data.get("username"))
    user.set_password(serializer.data.get("password"))
    user.save()
    return Response("Success.", status=status.HTTP_a_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

(3) 自定义修改密码页

将 Django 自带的密码相关 html 封装出来并引入. 切不可直接改库文件, 以免在部署时出现问题. 可在 settings 引入用以替代.

关键代码如下:

```
REST_AUTH_SERIALIZERS = {
  'PASSWORD_RESET_SERIALIZER': 'authorization.serializers.PasswordResetSerializer',
  'PasswordResetCompleteView': 'authorization.views.PasswordResetCompleteView',
}
```

(4) Token

可在 settings 中引入 JWT.

关键代码如下:

```
'DEFAULT_AUTHENTICATION_CLASSES': (
'rest_framework_jwt.authentication.JSONWebTokenAuthentication',
)
#继而可在 urls 中直接使用 token 接口:
  url(r'api-token-auth/', obtain_jwt_token),
  url(r'^api-refresh-token/', refresh_jwt_token),
  url(r'^api-token-verify/', verify_jwt_token)
```

可利用 Django 本身防止 XSS 注入攻击和防止 CSRF 跨站攻击的特性防止相应攻击. 需要加密的数据也会提交到处理与加密部分减少由于数据泄露造成的二次危害.

另外, 还可以根据情况自定义 jwt 的返回值并封装.

关键代码如下:

```
def jwt_response_payload_handler(token, user=None, request=None):
  return {
    'token': token,
    'role_code': user.role_code_input,
  }
```

另外可通过与当前时间的对比, 对 Token 按需进行更新、清除等操作.

关键代码如下:

```
#以更新 Token 为例
  updateToken(state, data) {
    // TODO: For security purposes, take localStorage out of the project.
    localStorage.setItem('token', data.token);
    state.jwt = localStorage.getItem('token');
},
#根据当前时间刷新 Token, 另写时间函数.
  updateTokenByCheckExp(state) {
    // 获得现在的时间
  var curtime = new Date();
  var newCurtime = Math.floor(DateAdd("h ", 0, curtime) / 1000);
  var newCurtimePlus = Math.floor(DateAdd("h ", 0.75, curtime) / 1000);
  var jwt = VueJwtDecode.decode(state.jwt);
  var interTime = newCurtime - jwt["exp"];
  if (interTime >= 0) {
    globalBus.$emit("SendMessage", {
    title: "token 过期",
    content: "您很长时间没有登陆了, 请重新登陆"
  });
  }
```

继而可通过 var interTime = newCurtimePlus - jwt["exp"] 判断是否需要对 Token 进行操作[9].

除了在 headers 中加 Token 进行验证外, 也可设置白名单. 在切换白名单外的 URL 路径时, 往 Django 自带的后端验证接口进行 post 操作, 验证 localStorage 中的 CsrfToken. 若密钥过期, 则需重新登陆.

关键代码如下:

```
const whiteList = ['/']
router.beforeEach((to,from,next)=>{})
```

3 系统展示

对系统关键环节进行截图展示, 敏感信息已被打码.

3.1 药店填报

(1) 登录注册

注册后, 用户会跳至基本信息确认界面. 通过 Leaflet

定位药店位置, 进行位置信息确认, 见图 1. 如若出现基本注册信息不匹配现象, 后台数据库会返回警示提醒.

(2) 信息填报

左侧导航栏是各项填报内容, 从上到下依次点击不同的问题组件填写, 可随时保存与重置, 见图 2. 全部项目填报完成后, 可进行结果核查. 最终系统根据考核权重进行计算得出自评星级与分数, 见图 3.

图 2 药店定位

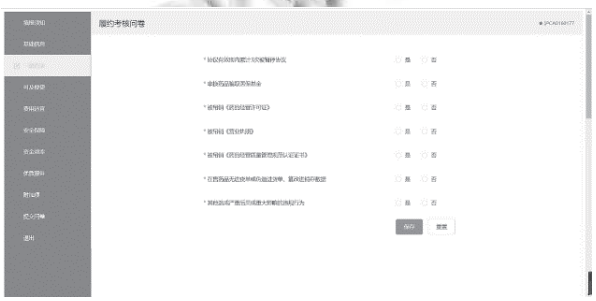

图 3 信息填报

3.2 管理界面

(1) 总体情况

输入管理员账号, 进入管理界面. 账号不同, 权限不同, 分为市、区两级.

登录后, 数据统计了所有医保定点药店 (医保协议化管理药店)、完成考核医保定点药店以及未完成考核医保定点药店的数目. 点击“完成考核医保协议化管理药店总数”, 可查看完成医保协议化管理药店的星级以及详细名单, 见图 4. (市、区级操作相同)

(2) 统计结果可视化

左侧导航栏是功能导航栏. 点击“图表结果”, 再“点击切换图表类型”可以切换可视化类型. 见图 5.

(3) 飞行检查

点击“详细结果”, 会将所有医保协议化管理药店列出. 可根据需要进行条件筛选. 点击“导出 excel”可将医保协议化管理药店名称, 医保交易代码和履约考核分数导出. 可进行快速搜索、详细信息查看、填报信息查看. 区级管理部门点击“修改填报结果”, 可对药店进一步飞行检查, 进入修改界面进行不实内容修改. 见图 6.

图 4 打分评级

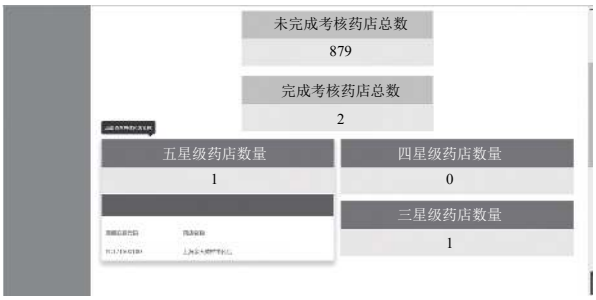

图 5 总体情况

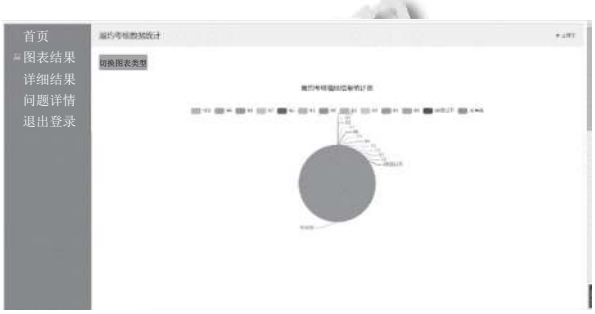

图 6 结果可视化

(4) 关键扣分项甄别

点击“问题详情”, 可以查询每道问题的填报结果, 区分为满分和非满分两类进行统计, 甄别关键扣分项. 见图 7 和图 8.

4 结果

本系统界面友好简洁, 操作方便, 所需功能一目了然, 实现了基本的信息管理功能, 使得考核数据的筛选、统计与分析工作等信息管理工作变得简单便捷.

相较于以往人力上门检查, 通过纸质或 Excel 等办公软件来管理填报信息的方式, 免去了大量人力、信息更新与查询统计等繁琐复杂的操作, 也为其他非专业人士省去了大量学习时间. 经测试及试运行显示系统鲁棒性强, 运行良好, 不仅为药店管理者、专家考核组和行政监管者提供了工作上的辅助, 更使得能便捷且全面的了解各种考核指标履行情况的信息, 为自评考核和飞行检查提供了可直观、便捷、安全的信息化平台, 通过量化得分的方式辅助实现了医疗保险协议化管理. 同时, 也为人民群众享受医疗保险带来的权利提供帮助.

图 7 飞行检查

图 8 关键扣分项甄别

具体而言:

(1) 将往日繁琐的数据填报信息化, 通过简洁的评分、排名、图表等形式将大量信息进行直观的表达, 助力决策者快速把握全局. 同时, 数据可以快捷地导出.

(2) 针对医保协议化管理药店的考核标准进行了清晰的定义, 包括打分原则、权重. 不仅以传统医保协议化管理药店视角进行自查, 更以实际需求为导向, 对决策者进行分级, 可辅助决策者实现更为精准的管理目标.

(3) 方便二次开发, 适应各类考核系统. 技术上, 系统采用 Python 为基本编程语言, 应用 PostgreSQL 数据库系列技术, 选用 Django 框架, 采用面对对象的分析方法设计, 用 Spring MVC 框架搭建网站构架, 完成后端的逻辑处理. 以集成系统为核心, 采用 MVC、MVVM、多层、ORM 等开发框架搭建系统, 实现功能. 采用 Node.js 和 Vue.js 进行 Vue 脚手架组件化开发, 将前端渲染的简洁有效, 并在辅助与 UI 设计, 使网站更加美观增加其实用性, 提高用户体验. 保证了对履约情况的考核评估实现科学化、规范化、信息化[10].

5 讨论

5.1 应用效果

本系统已完成上海市 16 个区 881 家的 2018 年度零售药店履约考核. 参加零售药店履约考核自评药店 881 家. 其中 880 家药店完成自评考核, 1 家药店未能完成自评考核. 在完成自评考核的药店中, 5 星药店 (≥95 分)652 家, 4 星药店 167 家 (≥90 分), 3 星药店 (<90 分)61 家

在此基础上, 各区级单位也通过本系统顺利进行了飞行检查工作, 完成了全市履约数据库构建, 并出具了全市医保定点零售药店履约考核情况报告, 助力医保工作顺利开展. 在本次零售药店履约考核专家抽查评审中, 共涵盖 163 家药店. 其中 163 家药店完成专家考核, 占所有参加考核药店的 18.5%, 108 家药店接受跨区专家考核, 占所有参加考核药店的 12.3%. 在完成区评考核的药店中, 5 星药店 (≥95 分)77 家, 4 星药店 58 家 (≥90 分), 3 星药店 (<90 分)28 家.

本轮考核周期由原先的 4 周缩短至 1 周, 减少行政人力 128 人月/年, 考核结果总体良好, 识别出服务质量不合格药店 (非 5 星药店)228 家, 占全部药店总量的 25.9%. 在所有的扣分项的履约考核指标中, 有 9 项扣分最多, 单项涵盖药店总数超过 600 家. 具体见表 1.

5.2 存在问题

在本系统运作期间, 经各区反馈, 部分药店出现无法提交的情况. 经查, 由于部分药店设备老旧, 仍在使用 Win7 系统或低版本浏览器, 导致系统部分功能失效. 因而, 本系统兼容性尚有改善空间.

综上所述, 搭建定点药店履约考核填报管理系统平台已初具成效. 在未来, 本履约考核系统可以作为常态化履约监管手段, 进行常态化纵向数据积累. 继而在与各方, 包括政府部门与民间组织的合作中, 逐步推进多平台联动运行, 最终建成可以面向政府、企业、公众的完善系统平台.

表 1 涵盖药店总数最多的指标

考核指标	未满分店数	占比(%)
统一悬挂“医保定点零售药店”标牌	829	94.2
店门口明显位置安装“定点药店夜间按铃”标识, 标识在按铃上方	748	85.0
设立24小时服务灯箱	724	82.3
设有夜间售药窗口(或大门卷帘门空隙代替)	702	79.8
有执业药师或药师24小时在岗, 并挂牌服务	647	73.5
同一商品名和规格的药品, 医保柜台的价格高于非医保柜台的药品数	641	72.8
同一通用名不同商品名的药品, 医保柜台与非医保柜台提供的品种不一致的药品数	640	72.7
医保服务区域(含休息区)靠近药店门口	639	72.6
“医保服务区”有明显标识	638	72.5

参考文献

1 孙娉娉, 李新, 史广军. 基于前后端分离的内容管理系统. 科研信息化技术与应用, 2016, 7(4): 70–75.

2 陈旭, 杨鹤标. 医疗保险数据可视化系统设计与实现. 软件导刊, 2017, 16(6): 59–62.

3 喻莹莹, 李新, 陈远平. 前后端分离的终端自适应动态表单设计. 计算机系统应用, 2018, 27(4): 70–75. [doi: 10.15888/j.cnki.csa.006311]

4 乔国兴. 地源热泵能源管理系统的设计与实现 [硕士学位论文]. 济南: 山东建筑大学, 2018.72.

5 毛炎. 基于 Vue.js 框架的 Web 方言地图的设计与开发 [硕士学位论文]. 武汉: 武汉大学, 2018.70.

6 朱二华. 基于 Vue.js 的 Web 前端应用研究. 科技与创新, 2017, (20): 119–121.

7 路雯雯. 支持前后端分离的 JavaScript 开发框架的研究及在内容管理系统中的应用 [硕士学位论文]. 济南: 山东大学, 2017.82.

8 张锣, 柯亚唯. 基于前后端分离技术的电厂信息管理系统后台程序开发. 科技创新与应用, 2018, (29): 156–157, 161. [doi: 10.3969/j.issn.2095-2945.2018.29.071]

9 Romaniello JF. jwt document maintained by José F. Romaniello. https://github.com/auth0/node-jsonwebtoken. 2015.

10 Gao BB, Zhang SD, Yao NM. A multidimensional pivot table model based on MVVM pattern for rich Internet application. Proceedings of 2012 International Symposium on Computer, Consumer and Control. Taichung, China. 2012. 24–27.