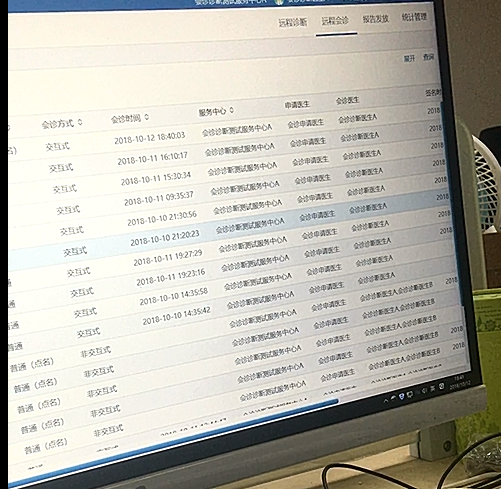
视频提取

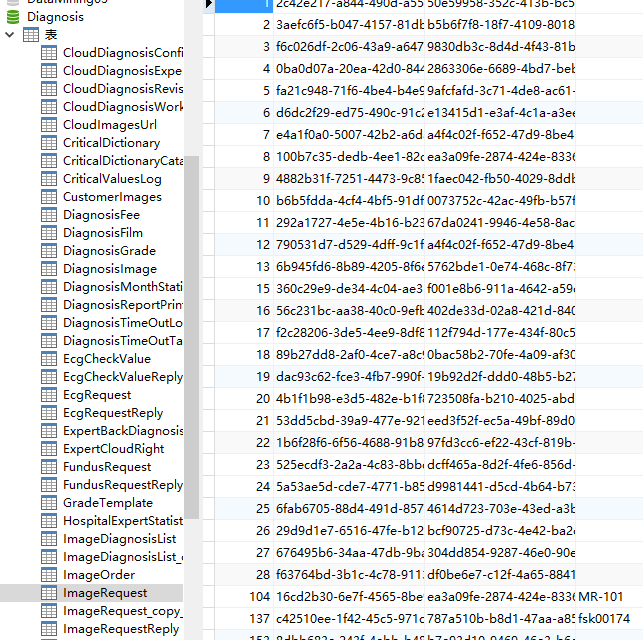
是

（1）

比如说会诊，我们首先要看医生和哪些服务中心有签约。把签约的数据查出来。



然后去会诊表里去找。我们一般只管这两个库——会诊和诊断。



要操作的两个数据库——192.168.1.200：

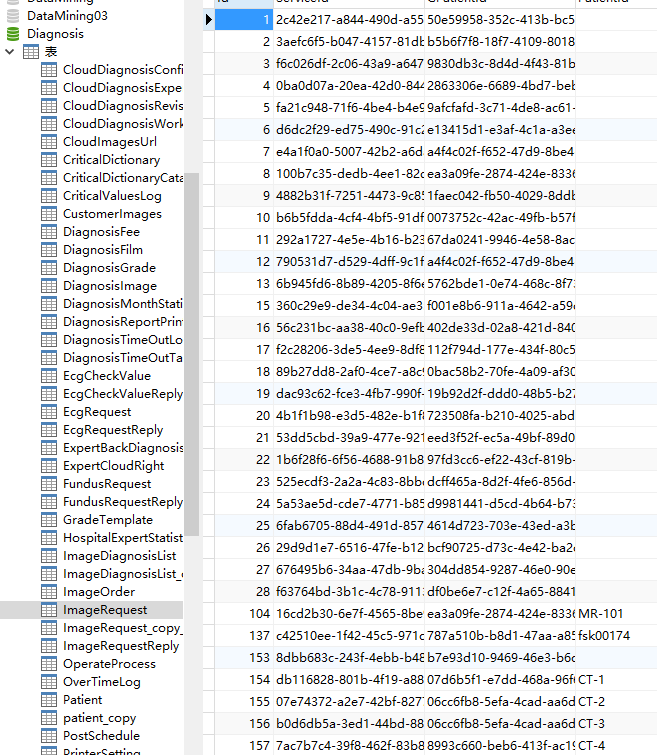
Consult（会诊）和Diagnosis（诊断）两个数据库。

Diagnosis（诊断）数据库中比较重要的三个表：

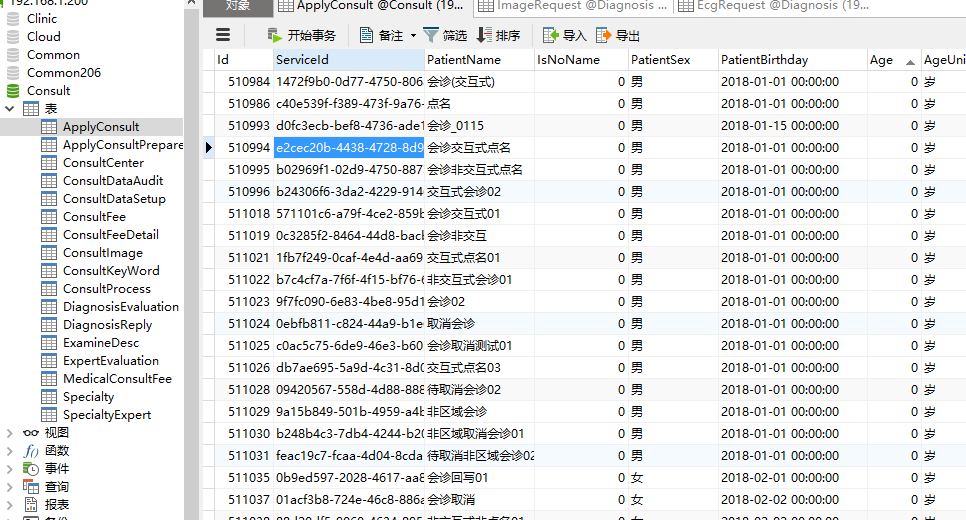
ImageRequest（影像诊断）

EcgRequest（心电诊断）

FundusRequest（眼底诊断）



Consult（会诊）数据库：

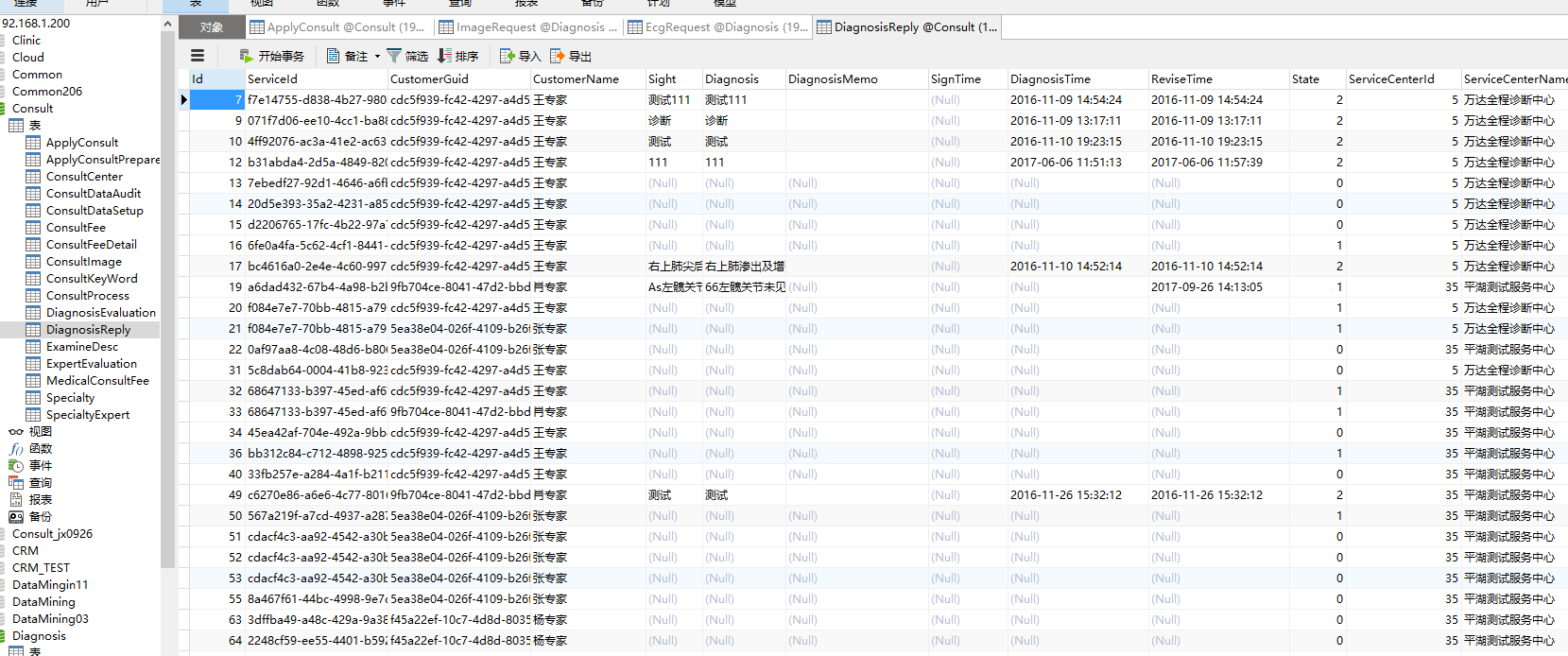


ApplyConsult：申请表。

一条会诊可能有多个医生，但每个医生只能查自己的，所以要去关联DiagnosisReply这张表。这张是医生的回复表。

前面写的医生的报告，就保存在这里。然后再加上用户的id，还有服务中心的id，各种连表，连接查询。

李阳：连是只连了两张，目前测试下来是还可以，速度。



用连表去查询。会慢。所以把 我的任务 和 任务领取给单独分开了。一个是属于你的，而另一个，非交互式是需要领取的。（相当于抢单一样？）



自己的补充：

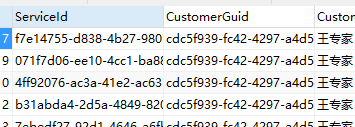
会诊的话，可以查看患者的影像，根据的是存储在数据库中的ServiceId：



url地址是

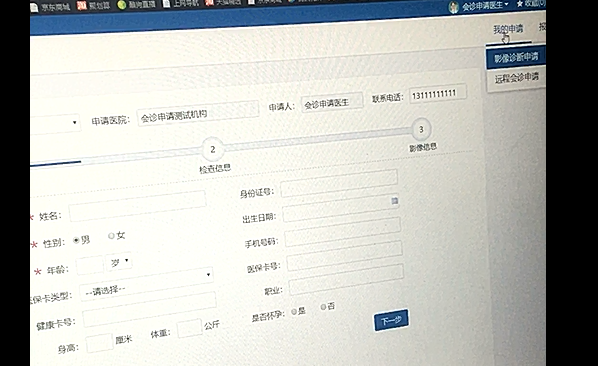
<http://192.168.1.202:9500/Doctor/ConsultV2/ConsultView?serviceId=168afb2f-ed7f-4b96-9f4f-5f542514e3ae>

推算应该是serviceId字段。

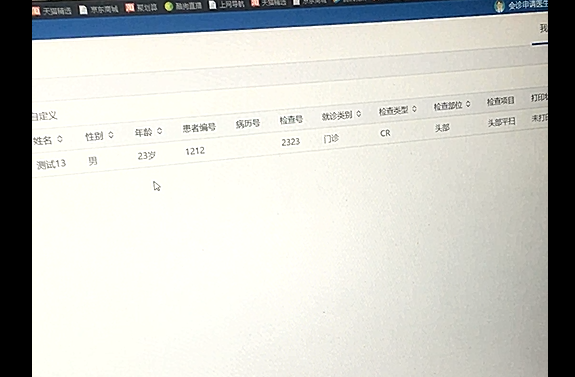


接下里是诊断，俞永红给我讲的。

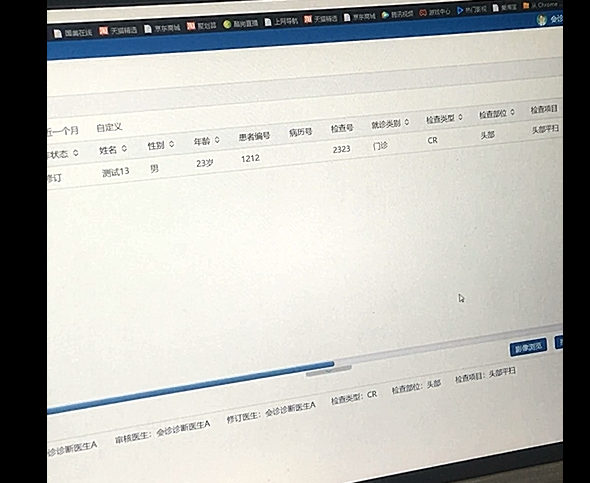
Web端就是医生申请，然后等着结果，（应该是上级的医生接单，然后给回复，我是这么想的）



申请完之后，这里就会有列表。

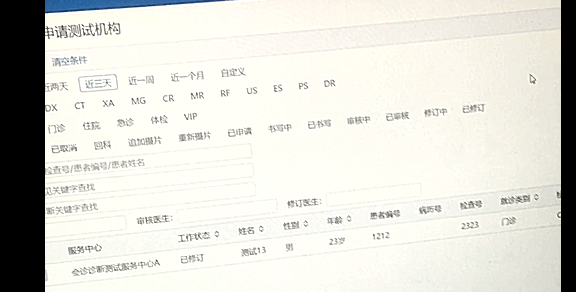


能看到其他医生处理完的列表。



根据条件去查询，俞永红表示，如果好几张表都会有关联的话，那查起来就慢，花的时间很长。我想，这个地方就要用的Lucene.net，或者说es了，每次去数据库里查，太不现实了。

反正现在的首要任务是把所有的都搞定，我也熟悉项目和流程，到时候根据需求再去做一个完美的版本！



申请端有一个统计：



他（也就是当前登陆的社区医生？）主要关心的是，他在某一个时间段，向哪些服务中心，申请了多少量。

就好比我们这家医院向你这个服务中心（省级大医院）提交了多少的量来。

俞永红又纠正，是我自己，也就是当前登陆的医生，他自己。他关心自己提交了多少的量，然后服务中心那边完成了多少。

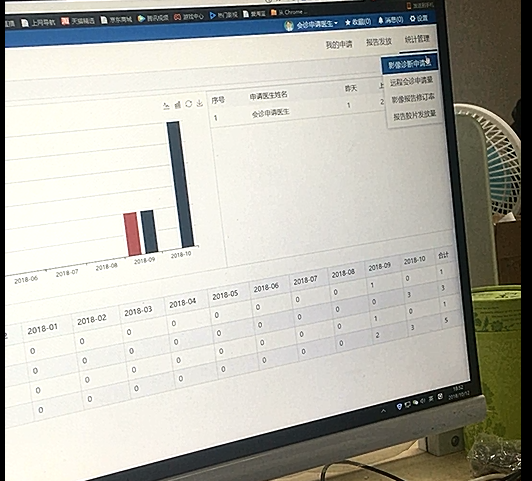
登陆医生所能申请的服务中心，必须是他所在的那家医生，和该服务中心又结对关系。

那该医生，就可以把申请给提交到结对的服务中心去。

然后服务中心的专家就可以把这些单子给做了。（我忽然想到，也不一定需要按工作流啊，因为它是非固定的，必须怎样流动的一条曲线。比方，我这边提交了申请，放入数据库，那服务中心那边的web客户端，只要有一个线程在轮询，一直去数据库里找数据不就可以了，然后显示在网页上，只要医生就可以抢任务了。我想到，这个轮询，也可以是定期地用ajax去查找）

我们的产品出去，肯定就是医院。这家医院肯定有这个能力，它买下这个产品，把区域内的卫生院都接进来。然后它开设一些服务中心出来，那小卫生院，比方是做ct的，那和该服务中心建立结对关系之后，ct都提交到服务中心这个平台，平台给你做好，那小卫生院只管拿报告。

申请，差不多就是一个申请，一个统计。



统计的话，统计他的申请量。服务统计的话，就相当于统计该医生申请上来的单子，被哪些专家给做了。然后做得怎么样。评判标准，做的快不快，然后做好之后，有没有进行多次的修改，如果多次修改了，肯定就做的一般了。如果修订率少的话，肯定说明这个专家比较专业。

患者是没有权利查看，而是依托于申请机构，它负责把你的病情往大城市的大医院去申请，替你发送一条影像诊断的申请，然后把结果反馈给你。



钱的话，他们自己去算。我们不管，我们只管服务。

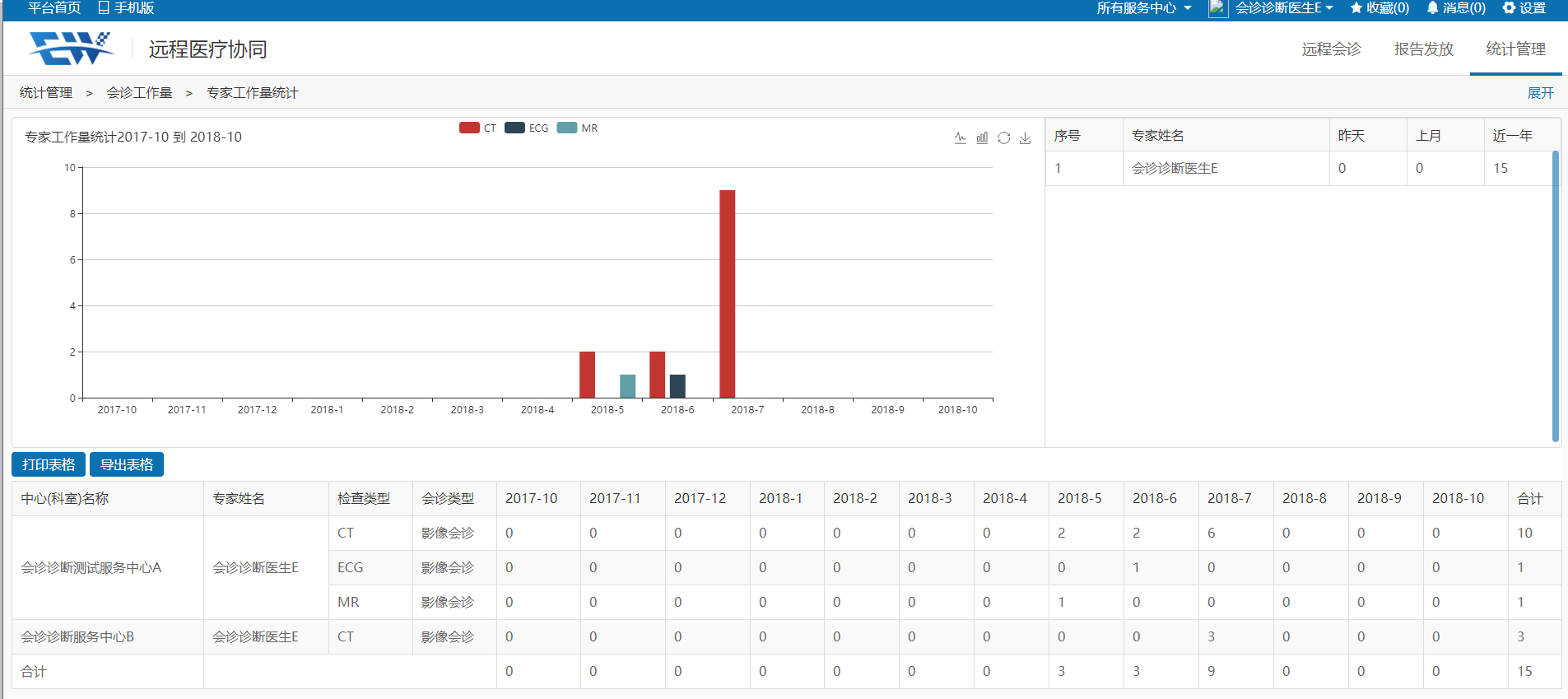
卫生院这里的统计，方便它和服务中心去结算。卫生院向服务中心支付相应的费用。它就按下面的表格，对每一家服务中心进行结算。

然后最后服务中心又会有一个统计，它去给那些专家进行结算。（做了事，总不能不给钱吧。对吧。）

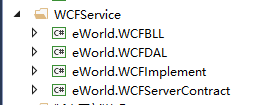
俞永红就是做这个申请和统计。原来有其他同事做过，但都走了。以后都是我们几个负责了。

老版本的问题就是，我们修改了新版本，我们觉得用不到的东西，给删了。但医院不肯，说是我们改坏了。

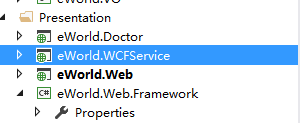
这个项目已经上线，已经有医院在用了。



wcf调契约层，契约层在这里：



我们的wcf直接来调上面这个契约：



这个契约是我们公司内部做好了，然后各个小组去调用，直接用就行了。

我们这个Presentation中的，比方Web，比方WebAPI，都是调的这个WCFService服务。

目前看到的api这些，其实都是壳儿，暂时。因为实际走的还都是下面的WCF契约。

可能以后，要把逻辑层都封到Web API中，以后都往api中获取数据。

以前都是Web直接调WCF，Web层就相当于一个寄存（没听清这个词）的口子。

之前也用过一套WebService的，太蛋疼，现在已经淘汰。

所以现在辛苦点，把Web API框架给搞出来，这样以后加新功能也就很方便。

现在搞个V2版本出来。

目前的接口还比较蛋疼，返回的都是类。

[HttpPost]

public BaseModel<DoctorInfoModel> GetDoctorInfo([FromBody]DoctorInfoParam param)

目前已经上线的状态是，也有调wcf的，也有调web api的。更老版本的，还有调web service的。所以老板已经决定，以后全部走web api。其他两个老的，慢慢的，尽可能就不去维护，给淘汰掉。

我们这边，全部提供post请求。

我们这边提供标准。维护起来也比较好维护。

之前做了新的model格式，但后来发现，这种格式也不太对。说是只有code，和message的话（没有听清，听明白），它还有个东西也要返回回去。很烦。

namespace eWorld.WebApi.Models.ImageDiagnosis

{

/// <summary>

/// 实体模型

/// </summary>

public class BaseModel

{

/// <summary>

/// 返回信息

/// </summary>

[DataMember]

public ReturnInfo ReturnInfo { get; set; }

目前是用这一套：

namespace eWorld.Core.DTO

{

/// <summary>

/// WebApi接口返回值数据模型

/// 无返回值

/// </summary>

public class BaseApiModel

{

/// <summary>

/// 返回码

/// </summary>

public int code { get; set; }

/// <summary>

/// 返回码描述

/// </summary>

public string msg { get; set; }

#region 静态方法

/// <summary>

/// 返回成功结果实体

/// </summary>

/// <param name="msg">消息</param>

/// <returns></returns>

public static BaseApiModel Success(string msg = "操作成功")

{

return new BaseApiModel

{

code = ResultStatusCode.OK,

msg = msg

};

}

如果没有返回结果的，只有code和msg，成功与否。

如果有结果传回去的话，就用这种套路：

#region 静态方法

/// <summary>

/// 返回成功结果实体

/// </summary>

/// <param name="t">对象</param>

/// <param name="msg">消息</param>

/// <returns></returns>

public static BaseApiModel<T> Success(T t, string msg = "操作成功")

{

return new BaseApiModel<T>

{

code = ResultStatusCode.OK,

msg = msg,

data = t

};

}

有分页的话就会有分页数据传回去：

/// <summary>

/// 带分页的数据集

/// /// </summary>

/// <typeparam name="T">数据集合</typeparam>

public class ApiPageListModel<T> : ApiListModel<T>

{

/// <summary>

/// 分页信息

/// </summary>

public PageInfo PageInfo { get; set; }

}

以上这里就是api的升级事宜。

说起这两个控制器的用户登录，其实一个就可以了。





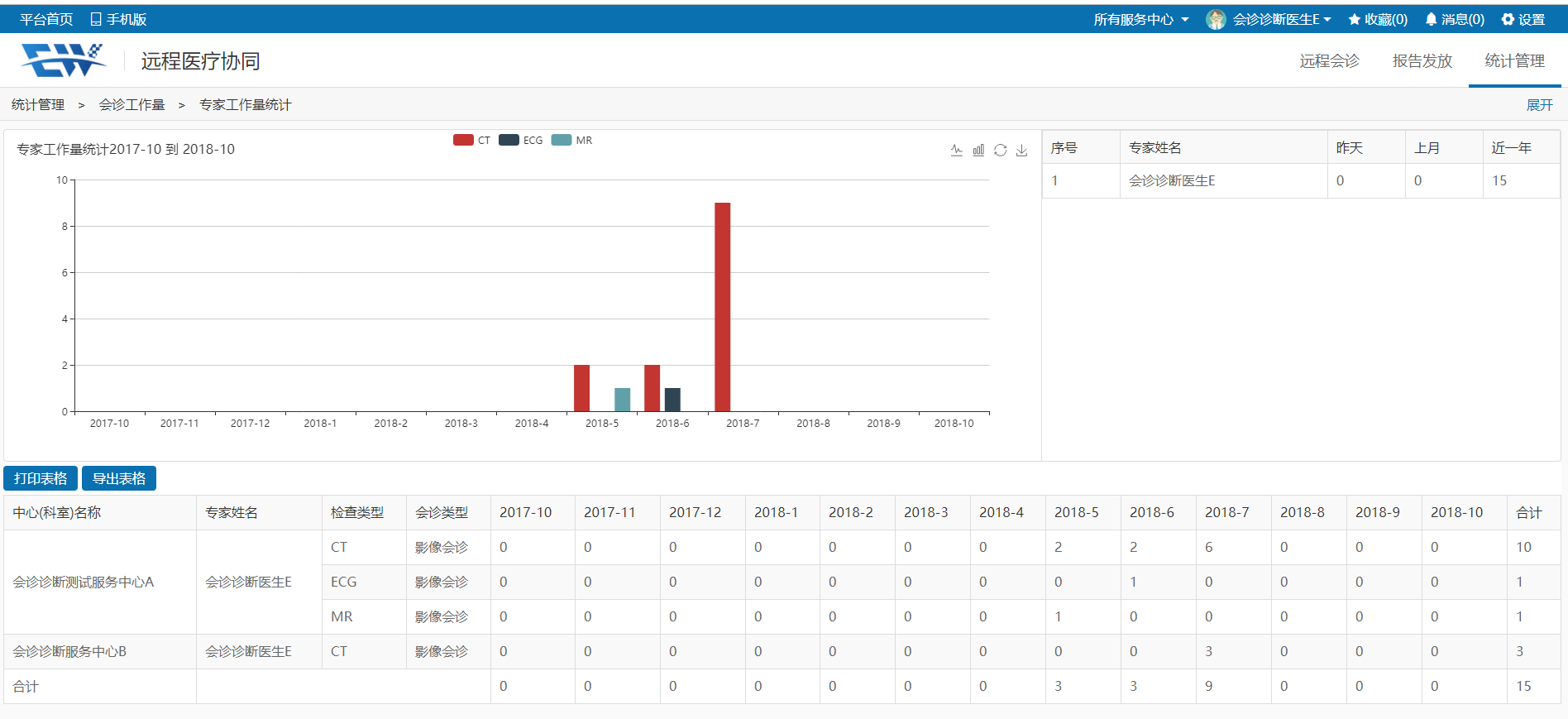




每个模块里面都有用户登录，到时候都要单独拎出来。对外登陆，到时候只需要留一个。

专家工作界面：

这里涉及专家权限问题，一般情况下，专家的权限，都是系统后台给他配置好的。



统计管理也有多个选项：

影像诊断工作量，远程会诊工作量，影像报告修订率，报告胶片发放量。



当前用户是只有一个会诊中心的身份：

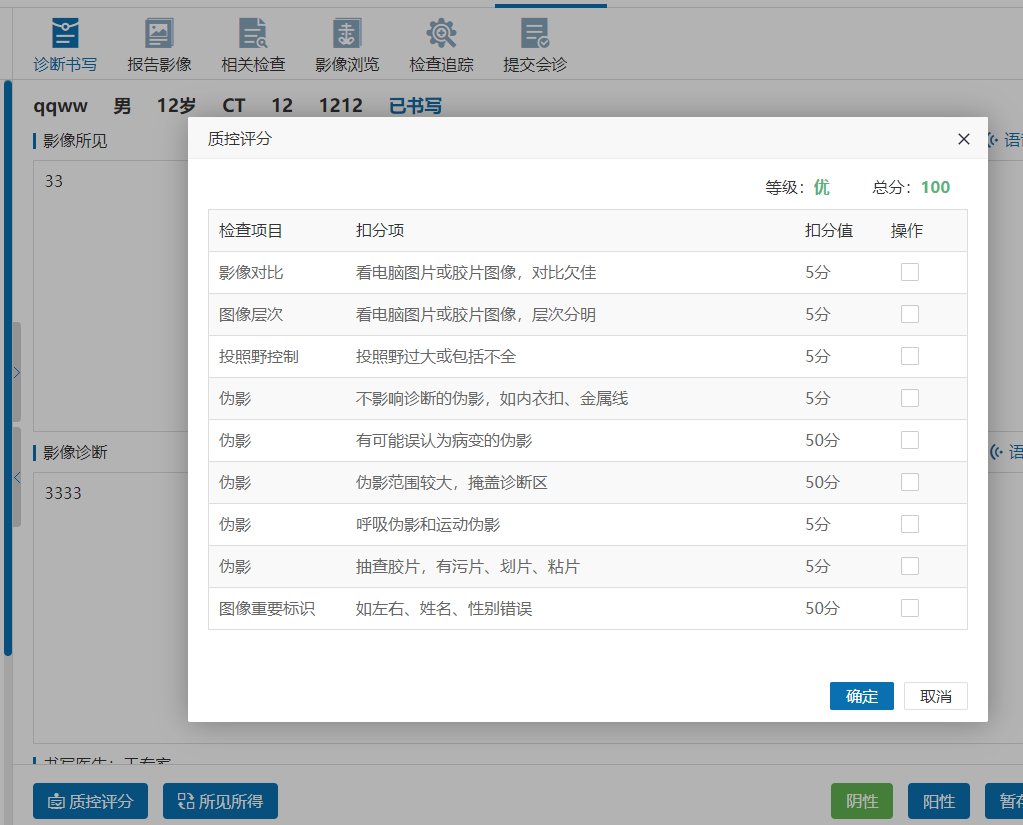


换一个专家：

另外，账户设置功能，可以设置和修改各种信息：



医生开始书写报告：



写完签名后弹出质控评分，应是对于此次诊断的评价。

其实变暗就是disabled=”disabled”



用chrome直接修改了，就又可以点击了。这种算bug吗？

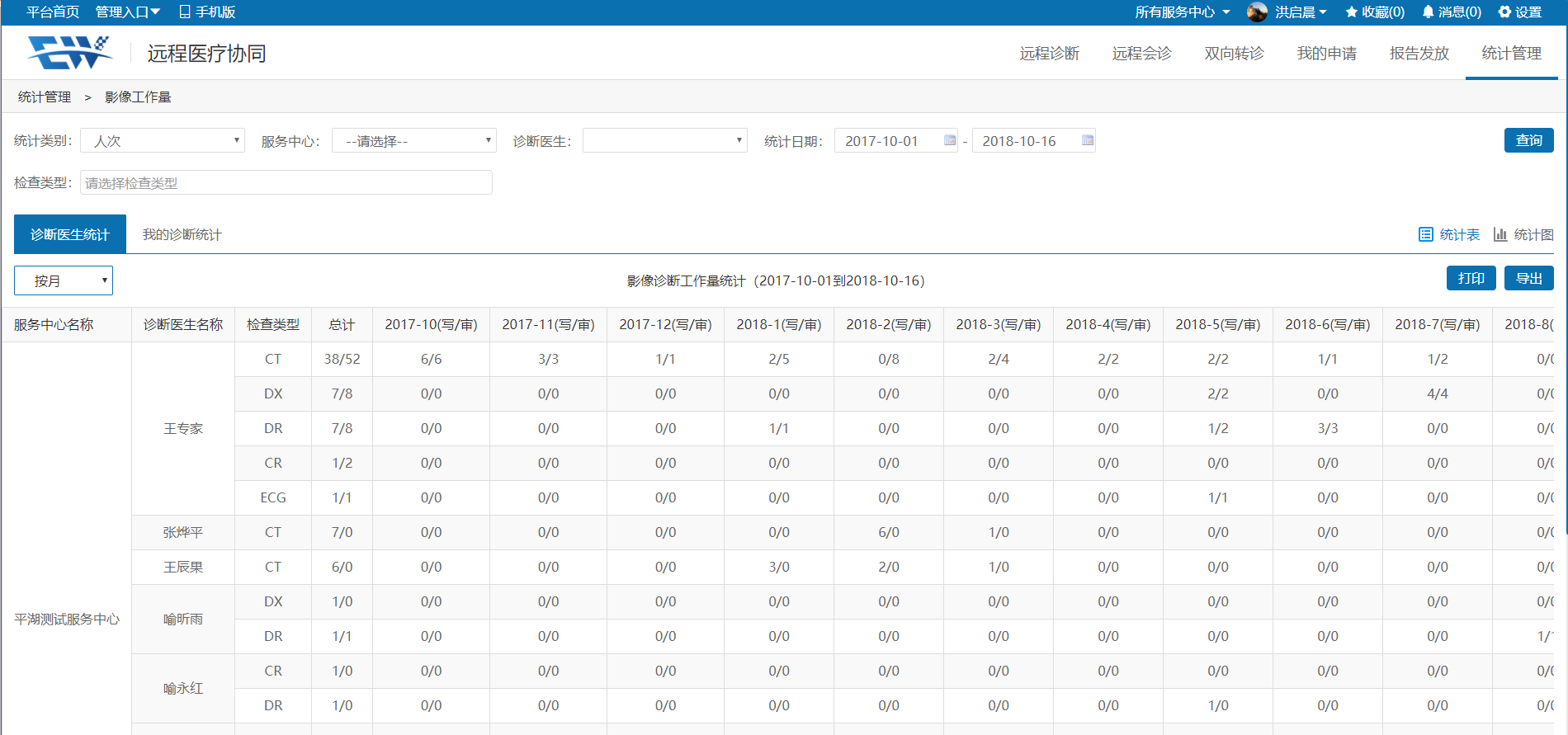
反正，骗骗医生肯定是可以了。

管理员的话，权限当然是满的：





管理员就可以看到所有对应服务中心，所有医生的工作量，方便统计软件使用率，成功率等等。



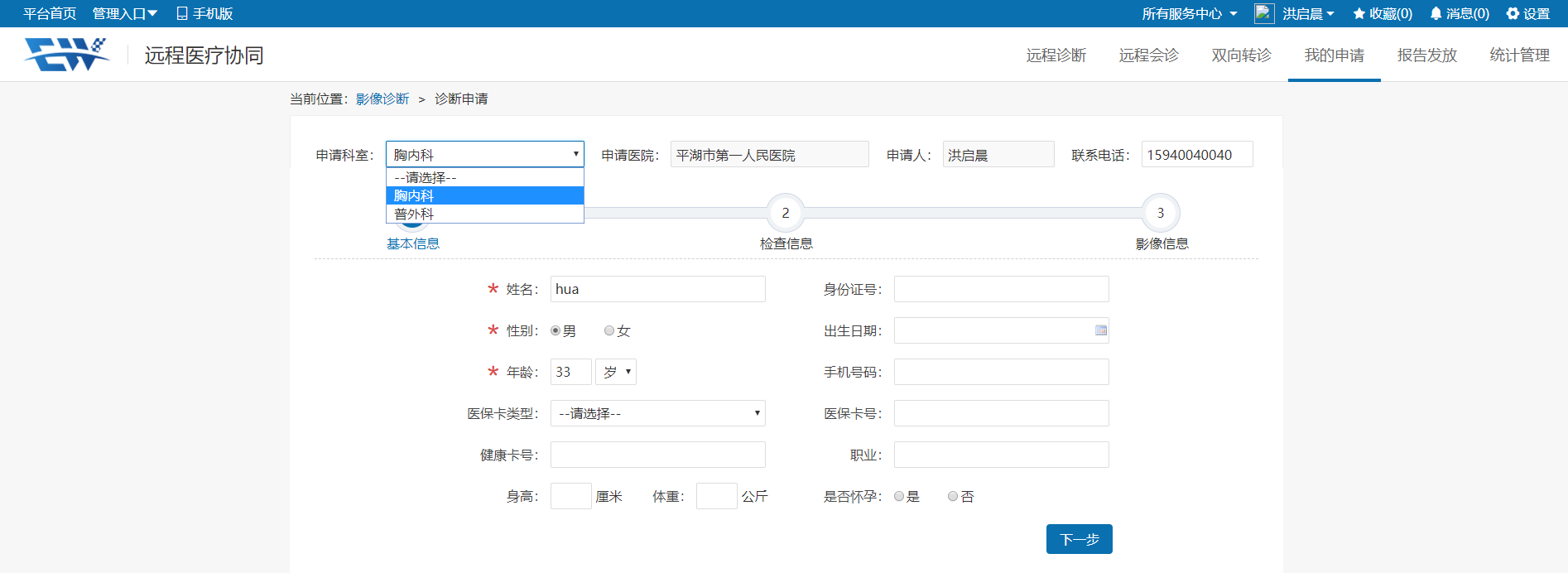
（2）

总结：如李阳所说，后台嘛，访问数据库，查到自己想要的数据，处理逻辑，然后给其他人提供接口。

确实，发现各个小组之间的交流，通信，就是接口。

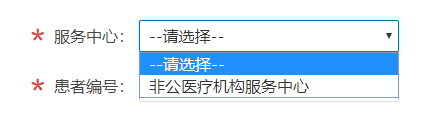
我试下看申请：

第一步，填写基本信息：

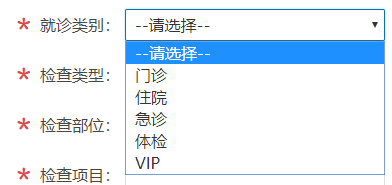


第二步，填写检查信息：

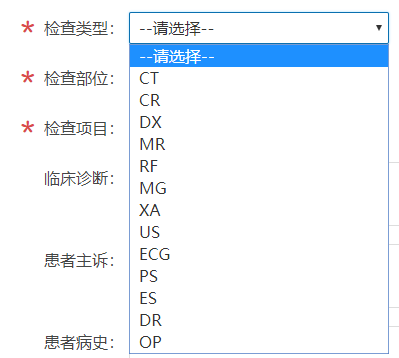
（1）服务中心的选择



（2）就诊类别的选择：



（3）检查类型的选择：



（4）将其余必填的项目补全：



第三步，选择影像资料：



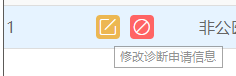
提交之后，申请成功。



这样就申请出去了：



可以修改诊断信息



还有撤回：



接下来我换专家去处理：

我换一个浏览器，用360去另外一个账号登陆：

我在360这边登陆了王专家这个账号，看到了前面的这个申请，姓名为hua。





操作有三个：

（1）影像浏览



（2）修订报告：

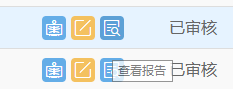
发现已申请，已书写，鼠标移动上去，没有修订报告的提示，而只有判断右边的状态为已修订，才弹出这个动态的div显示。



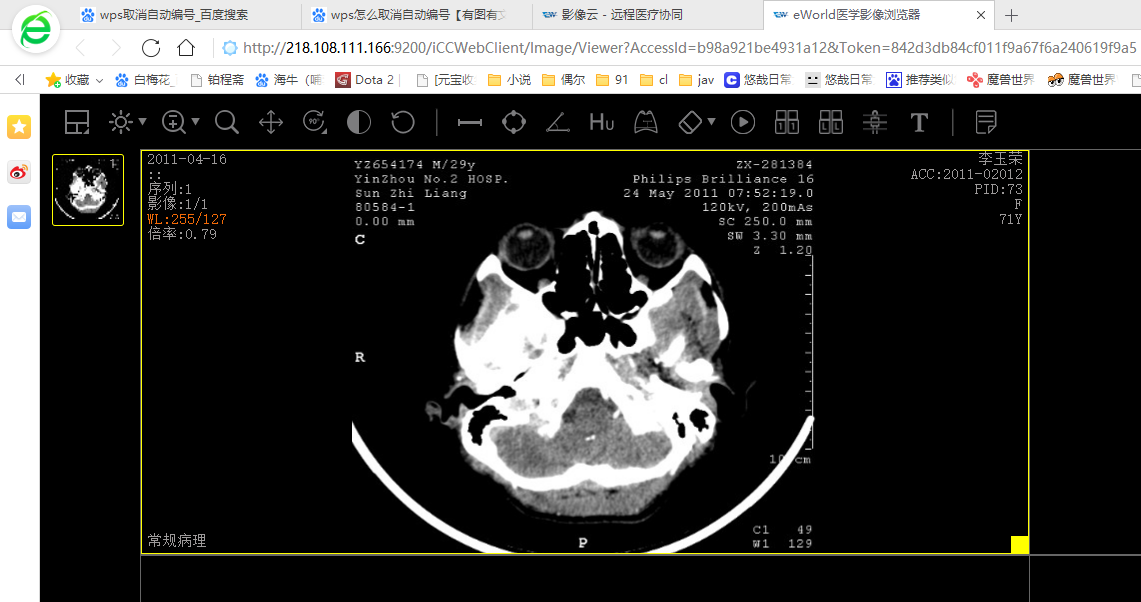
（3）领取任务：



而其他已经处理的，显示的就是查看报告这个div。



聪明的医生可以先看看这个影像，自己会不会，会的话再接下。



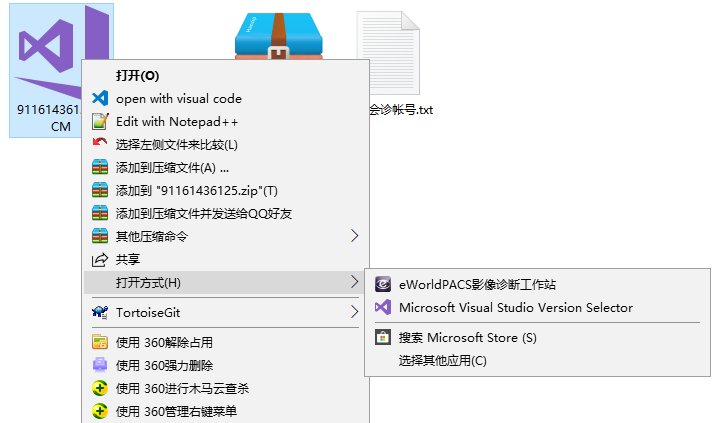
url地址：

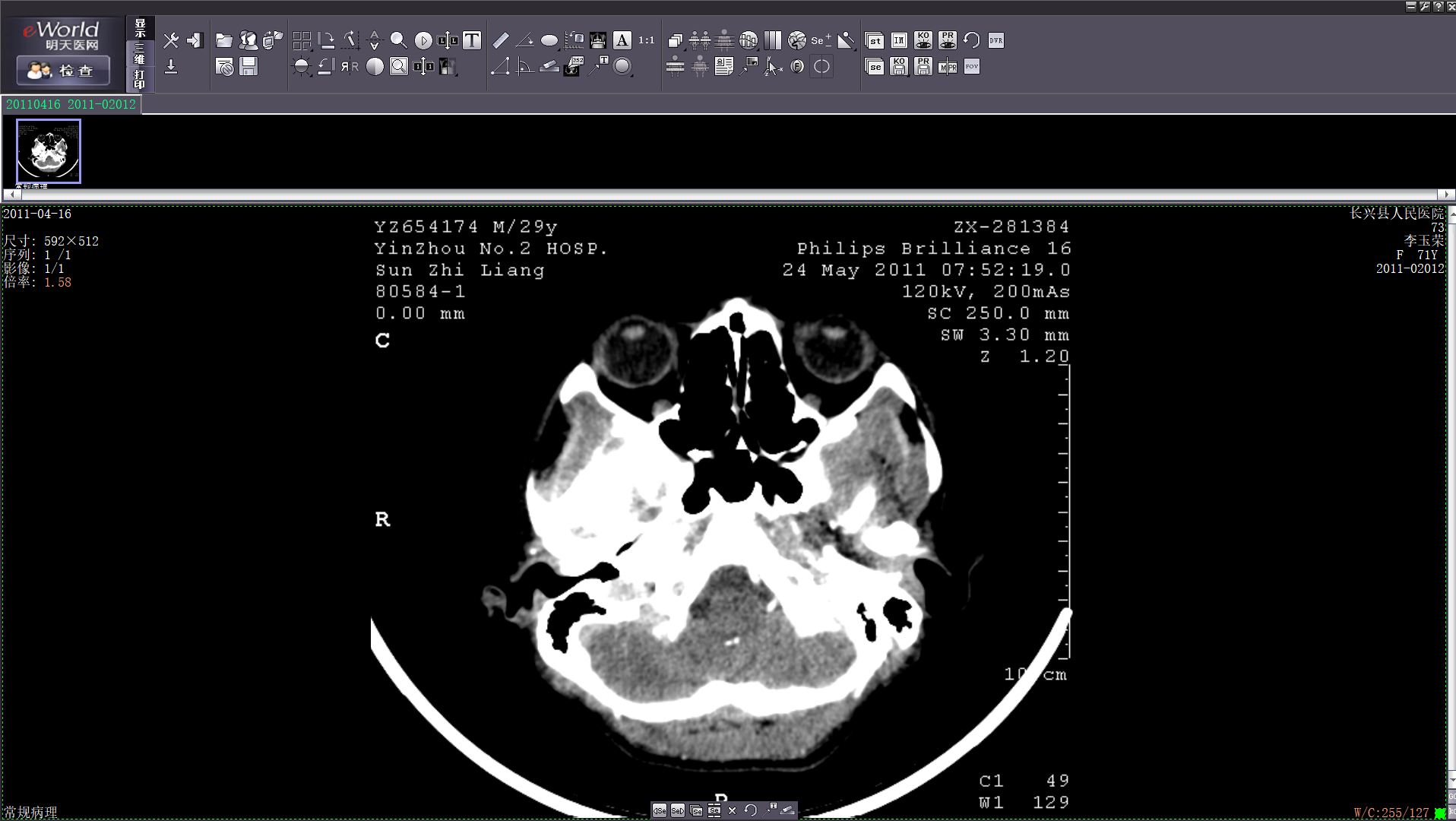
<http://218.108.111.166:9200/iCCWebClient/Image/Viewer?AccessId=b98a921be4931a12&Token=842d3db84cf011f9a67f6a240619f9a5>

这个iCCWebClient就是原本在苗哥那里，要我做的。

用打开软件打开这个测试dcm后，发现果然是同一张图片。

那可以总结出，上传的时候是通过接口放到了苗哥那里，而浏览的时候，他们直接通过AccessId去苗哥那边的文件服务器拿到这张图片，也许网页上拿到的是直接可以浏览的模式。





我尝试着用Navicat去连一下苗哥那边的数据库：

果然可以连接，测试成功的。



连上了苗哥的数据库，没找到这个AccessId的字段。

搞了半天回来，发现登陆超时了，尼玛。



又要重新登录了，所以这里的cookie要改，他每次使用网页，用到用户校验的地方，就要重新写一次，然后给保存在客户端那边。

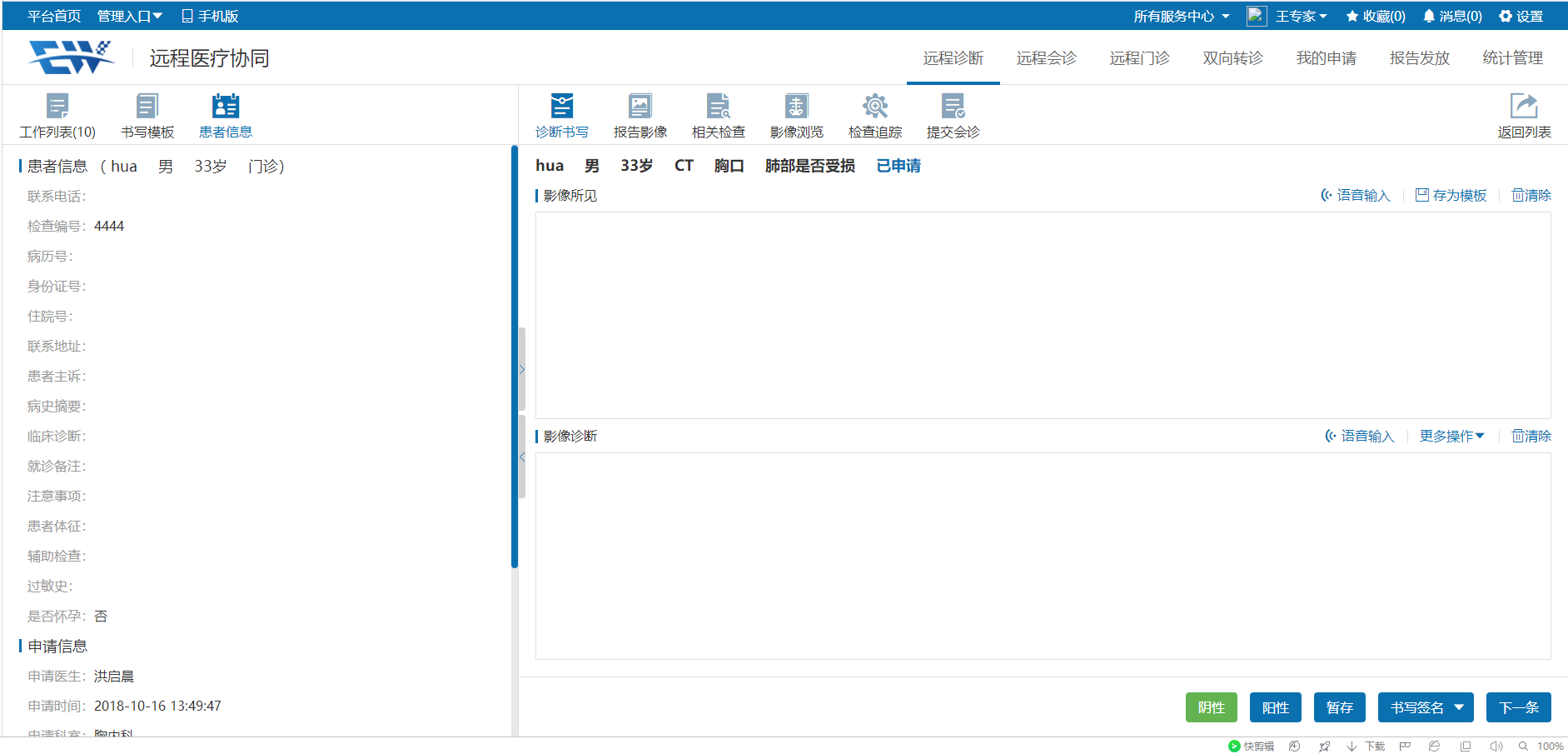
放在我们这边的Redis上是最好的，每次过来都查一次。

反正目前，这个记录登录用户的问题，可以完善一些，后期我再测试。

继续回来，我把这个hua的任务给领取了。鼠标进入，变成了“释放领取”这个触发事件。

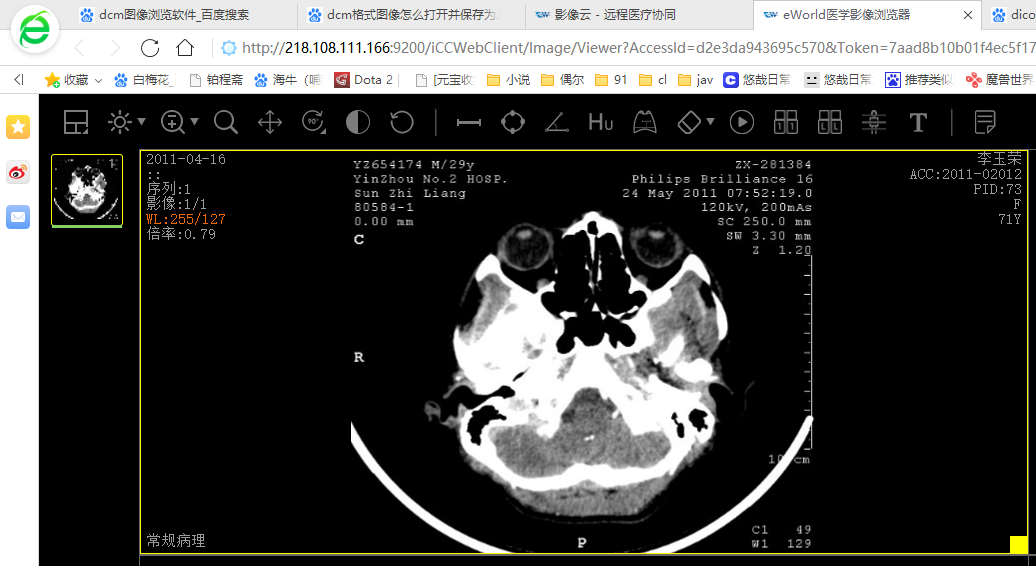


然后我开始书写这个诊断，诊断的同时默认跳出来ct影像：



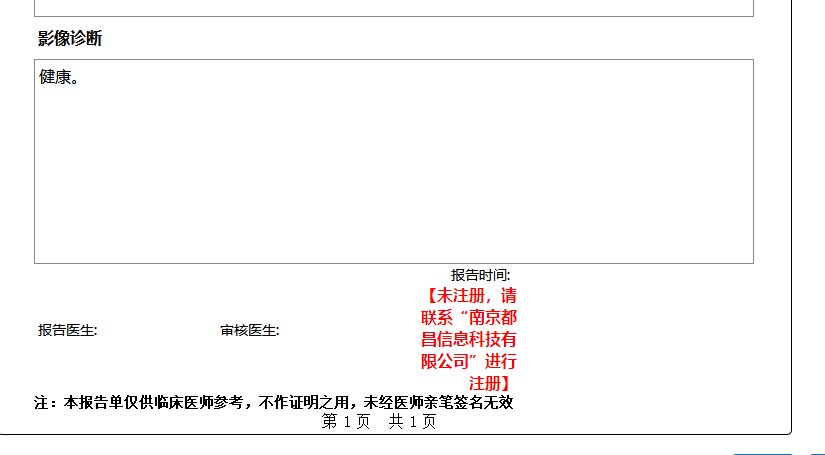
点击影像浏览继续链接到传上来这张图上（苗哥那边给力啊）：





所见所得：





点击暂存之后，我想应该是保存当前的书写。



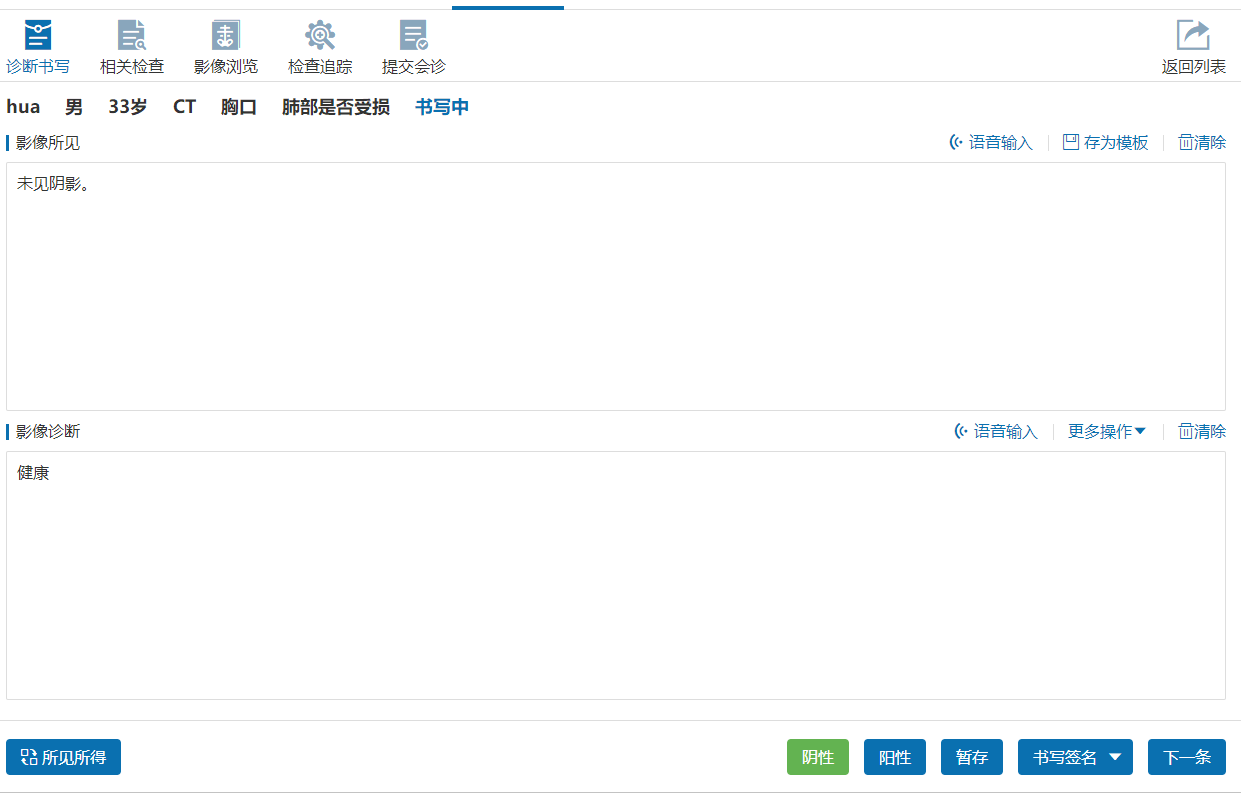
退出，果然看到此刻的工作状态变成了——书写中。



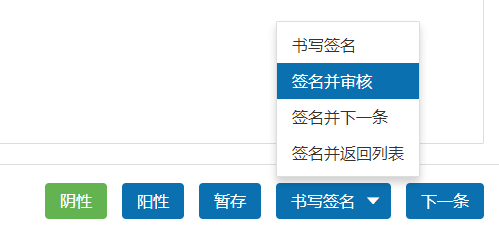


继续书写报告，发现还是原来的内容，就是保存了。感觉这种保存可以放在cookie中，还是说数据库中？

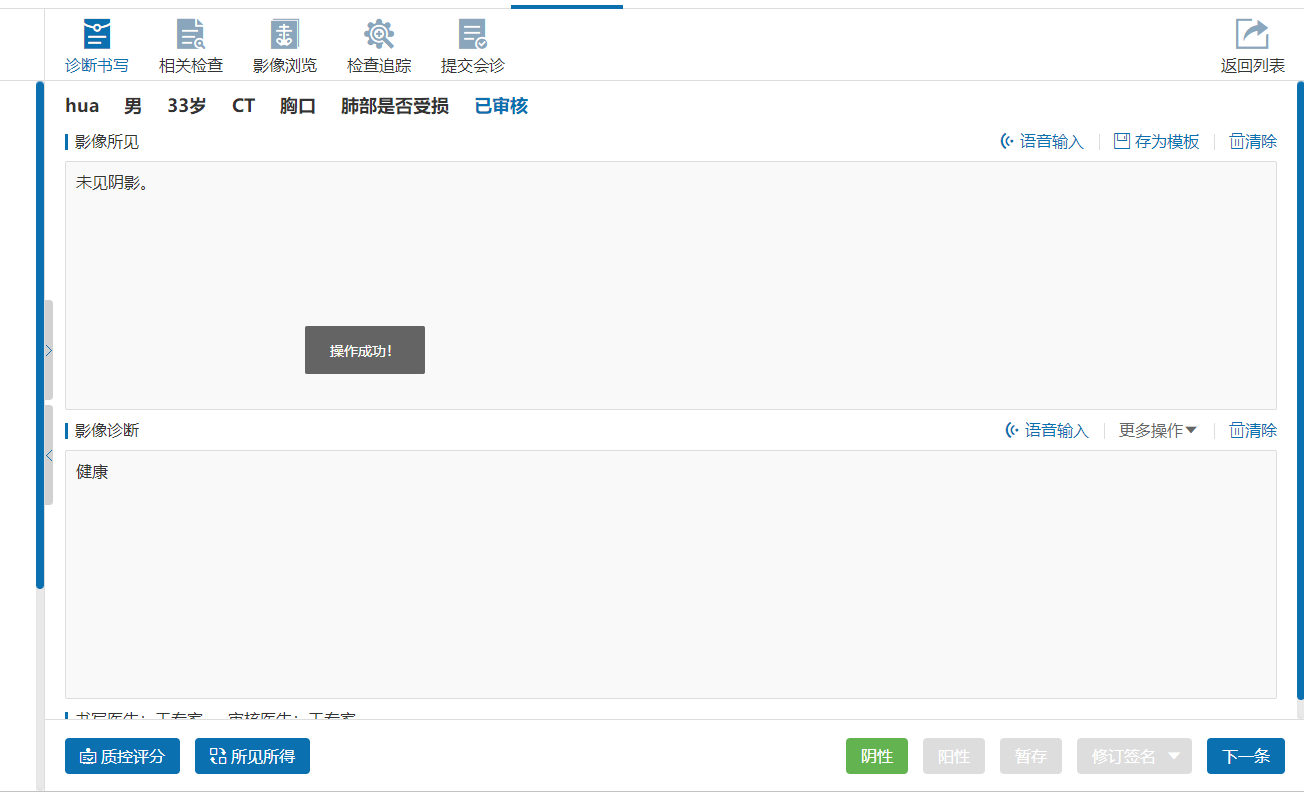
同时又点击进入诊断的时候，还是自动弹出来影像。



接下来书写签名：



选择签名并审核。

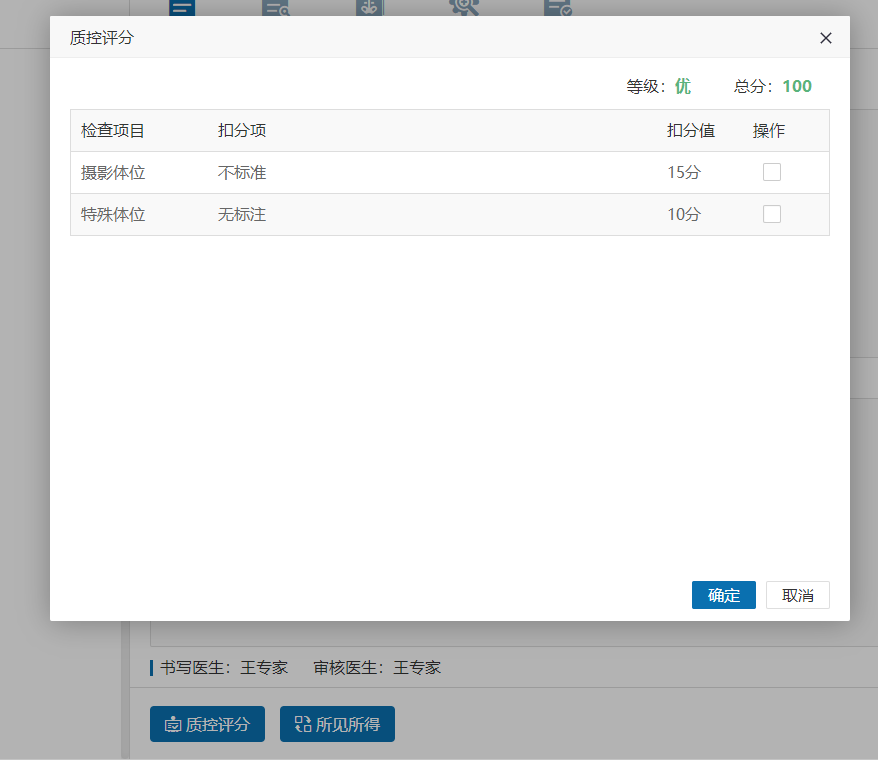


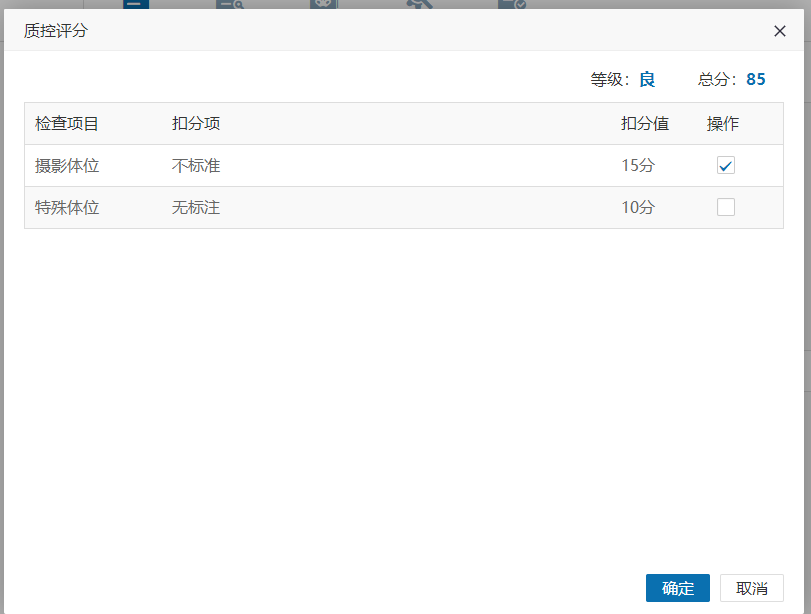
显示成功。

这时候点击修改前面这些无效，因为已经disabled。



这时候可以开始质控评分了：





接下来点击就变成 质控详情了。



这时候就ok了，可以查看报告了。



我修订了下，写了第二版。

查看报告，果然是：



就是现在社区医生和专家医生没有分开来，都在一起，你也可以申请，可以诊断，我也可以申请，可以诊断。

到时候要把权限区分开来，社区的，只能申请，等结果，而大医院的，才有资格诊断。



接下来，我换成申请医生的账号，果然看到了诊断结果，这样，我就可以跟我卫生院的，在外面等待结果的人说明检查结果了。

我想到洋安那里的妇保，訾妈去照b超和ct，然后在外面等着拿报告。医生会说明一些情况。但如果比较复杂，严重的话，当时的值班医生就可以远程申请，让省妇保的专家来做，这样又省时间，又专业。

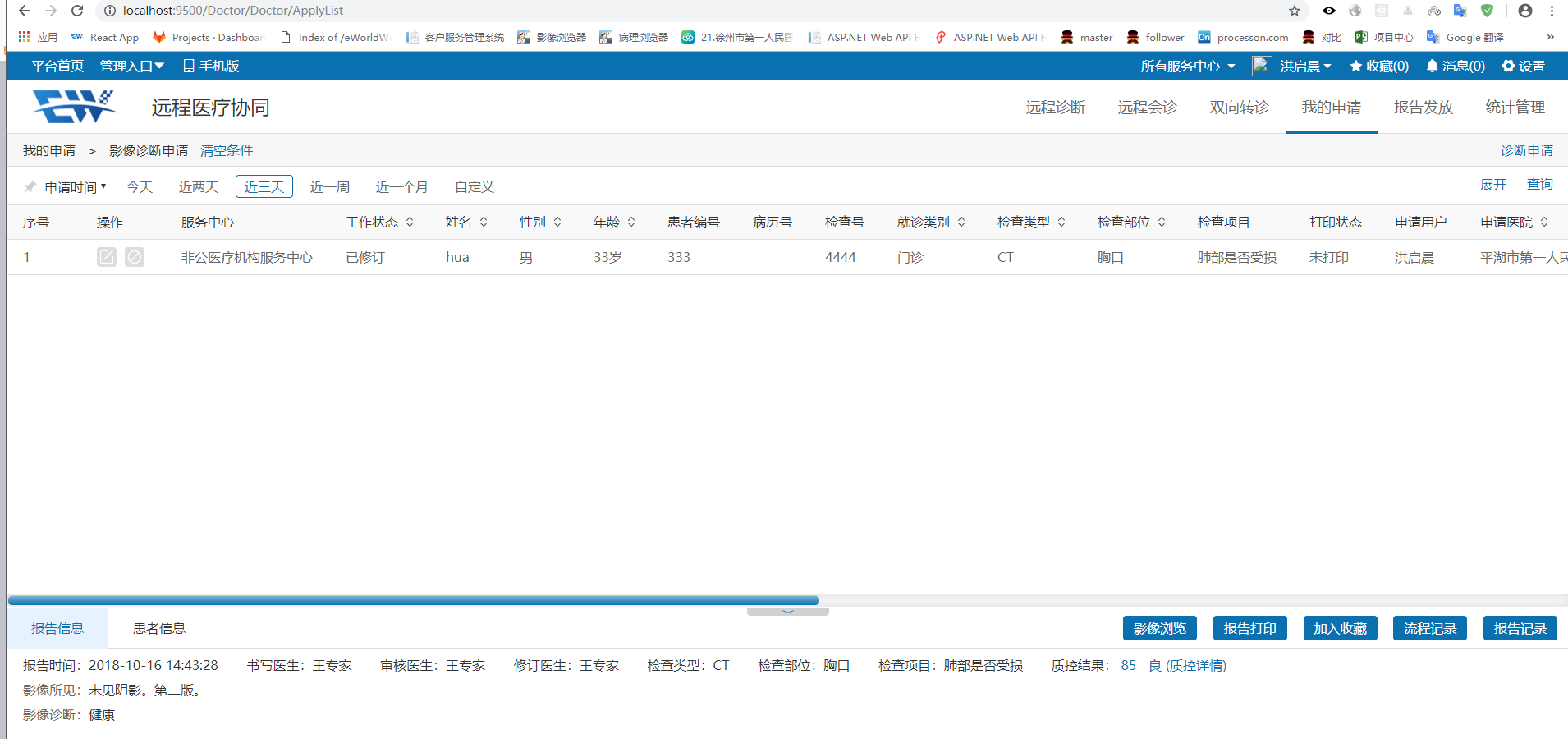
这样，城镇里面的人，也不用跑到医院了。

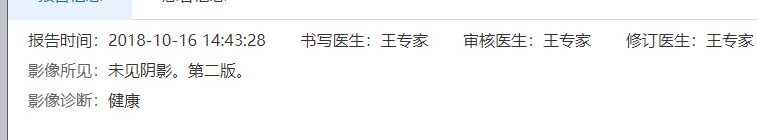
我居然有了一种使命感，你敢信。

毕竟农村看病难，让这些人跑杭州，上海，也是很不现实，因为都不肯跑大城市来，除非真的不行了。但大城市的专家也吐槽，你人都这样了，送来还有什么用呢。直接又让回去，料理后事了。

可送来的人，哪里肯。还没看，就说不行了。以为不肯治，忽悠人。

但以医学的角度，确实已经不中用了。因为太迟了。





这里面的菜单按钮。



（1）影像浏览

按影像浏览继续可以弹出影像，这里应该是前端的任务。

（2）报告打印



（3）

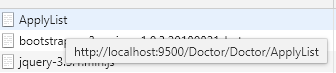
仔细深入了下：

1.点击我的申请，调到申请页面。

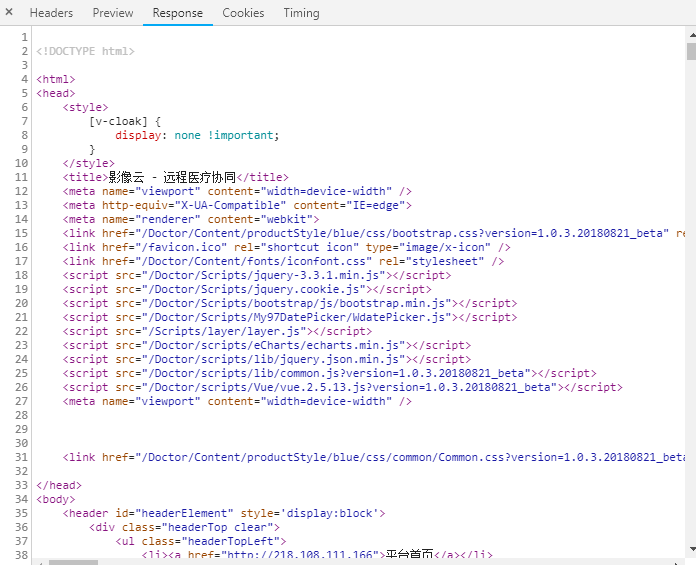


url地址：<http://localhost:9500/Doctor/Doctor/ApplyList>

chrome监控：



查看response，也就是返回当前view的整个页面

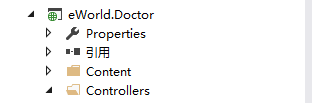


cookie这里依旧是eWorldToken。



我顺便去看下url所对应的vs中的控制器controller和action方法：

问了李阳，果然路由对应的是这个Doctor，不是下面被排除的Doctor文件夹，李阳以前也和我一样，一来公司就给包含进来。一样地犯错。



这里用了这么个技术：

namespace eWorld.Doctor

{

public class DoctorAreaRegistration : AreaRegistration

{

public override string AreaName

{

get

{

return "Doctor";

}

}

public override void RegisterArea(AreaRegistrationContext context)

{

context.MapRoute(

"Doctor",

"Doctor/{controller}/{action}/{id}",

new { controller = "Doctor", action = "DiagnosisIndex", area = "Doctor", id = UrlParameter.Optional },

new[] { "eWorld.Doctor.Controllers" }

);

}

}

}

这样，url地址就连到这里来了。

查看Action：

#region 远程诊断申请列表页面

/// <summary>

/// 远程诊断申请列表页面

/// </summary>

/// <returns></returns>

public ActionResult ApplyList()

{

return View();

}

#endregion

调到前台，仍旧是要传用户信息：

@{

ViewBag.Title = "远程诊断";

Layout = "~/Doctor/Views/Shared/DoctorLayout.cshtml";

var customerIdentity = ViewData["CustomerIdentity"] as CustomerIdentity ?? new CustomerIdentity();

}

<input type="hidden" id="hdnIsCA" value="@this.ViewData["IsCA"]" /><!--是否启用CA，0：否，1：是-->

<input type="hidden" id="hdnCACompany" value="@this.ViewData["CACompany"]" /><!--CA厂家，SHCA：上海数字认证，BJCA：北京数字认证-->

<script>

var baseData = {

CustomerGuid: '@customerIdentity.CustomerGuid',//当前操作用户UID

CustomerName: '@customerIdentity.CustomerName',//当前操作用户名称

};

</script>

这个数据应该是在控制器上面打的标签：

继承自防火墙控制器：

[MvcAuthorize]

public class DoctorController : FirewallMvcController

{

防护墙控制器又进一步继承自Base控制器：

public class FirewallMvcController : BaseController

{

Base继续接龙到最初的控制器Controller：

[WebSiteIpAddress]

[CustomerLastActivity]

[WebSiteLastVisitedPage]

public abstract class BaseController : Controller

{

摘录下授权过滤器：

namespace eWorld.Web.Framework.Controllers

{

/// <summary>

/// 登录控制器

/// </summary>

public class MvcAuthorizeAttribute : AuthorizeAttribute

{

/// <summary>

/// 授权过滤器

/// </summary>

/// <param name="httpContext"></param>

/// <returns></returns>

protected override bool AuthorizeCore(HttpContextBase httpContext)

{

CustomerPrincipal customerPrincipal = httpContext.User as CustomerPrincipal;

if (httpContext.User.Identity == null || !httpContext.User.Identity.IsAuthenticated || !EngineContext.Current.Resolve<ISsoProvider>().IsLogin)

{

httpContext.Response.StatusCode = 401;

return false;

}

if (customerPrincipal != null && !customerPrincipal.IsInRole())

{

httpContext.Response.StatusCode = 403;

return false;

}

return true;

}

/// <summary>

/// 重写HandleUnauthorizedRequest身份验证

/// </summary>

/// <param name="filterContext"></param>

protected override void HandleUnauthorizedRequest(AuthorizationContext filterContext)

{

base.HandleUnauthorizedRequest(filterContext);

if (filterContext.HttpContext.Response.StatusCode == 401)

{

string url = $"{HttpContext.Current.Request.Url.GetLeftPart(UriPartial.Authority).TrimEnd('/')}/Login";

if (!filterContext.HttpContext.Request.IsAjaxRequest())

{

filterContext.Result = new RedirectResult(url);

}

else

{

filterContext.Result = new JsonResult()

{

Data = new { code = ResultStatusCode.VALID\_TOKEN\_FAIL, msg = "登录失效", url },

ContentType = null,

ContentEncoding = null,

JsonRequestBehavior = JsonRequestBehavior.AllowGet

};

}

}

if (filterContext.HttpContext.Response.StatusCode == 403)

{

if (!filterContext.HttpContext.Request.IsAjaxRequest())

filterContext.Result = new RedirectResult("/Common/RedirectPage?msg=您无权访问该网站'</script>");

else

filterContext.Result = new JsonResult()

{

Data = new { code = ResultStatusCode.UNAUTHORIZED, msg = "非法访问", url = "/Common/RedirectPage?msg=您无权访问该网站" },

ContentType = null,

ContentEncoding = null,

JsonRequestBehavior = JsonRequestBehavior.AllowGet

};

}

}

}

}

看来都是过滤器啊，用的。和老杨的风格是一样的。

过滤器我自己的感觉，就是比方我可以提供365个事件，这些事件我都已经埋在整个框架当中，你要用的话，自行选择，去继承我的主接口，然后重写你想要的事件方法，然后在你想要触发的控制器或者方法上打上标签，那这样，它就会自动去实现了。

我得看看老杨的掌上租的项目，看看过滤器到底用到了怎样的境界。

1.5

在获取http://localhost:9500/Doctor/Doctor/ApplyList中，他会根据默认的选择自动调用这三个url地址：



1.5.1

第一个方法：

<http://localhost:9500/Doctor/Doctor/GetCenterExamineTypeList>

请求所带参数



{"serviceCenterIdList":[164,168,40,35]}

返回的response：

{

"code": 0,

"msg": "成功",

"data": [

"DX",

"DR",

"CT",

"CR",

"MR",

"RF",

"ECG",

"OP"

]

}

感觉不像是给前台提供筛选的选项：



再查看控制器中对应的Action方法：

#region 获取诊断中心检查类型

/// <summary>

/// 获取诊断中心检查类型

/// </summary>

/// <param name="serviceCenterIdList">诊断中心Id列表</param>

/// <returns></returns>

[HttpPost]

public JsonResult GetCenterExamineTypeList(List<int> serviceCenterIdList)

{

var retValue = \_crmClient.Invoke(t => t.GetCenterExamineTypeList(serviceCenterIdList));

return Json(new { retValue.code, retValue.msg, data = retValue.data?.DataList.Select(p => p.ExamineType).Distinct().ToList() });

}

#endregion

可以看到这里的参数需要的是List<int> serviceCenterIdList类型，而传过来的就是

{"serviceCenterIdList":[164,168,40,35]}

还是通过wcf契约去调的数据。

namespace eWorld.WCFServerContract

{

public class eWorldWCFClient<TChannel>

{

wcf中的方法：

public TResult Invoke<TResult>(Func<TChannel, TResult> function)

{

return this.\_serverInvoker.Invoke(function);

}

顺便查了发这里看到的代码，C#6.0新语法：

四、空值运算符

我们都知道，null值是不能调用ToString()方法的，会报“未将对象引用设置到对象实例”的错误，例如：

1 string name = null;2 Console.WriteLine(name.ToString());

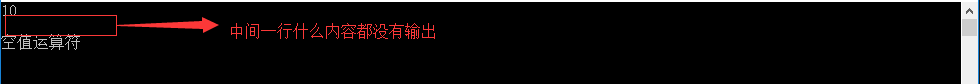
 结果：

IMG_256

在C#6.0中有了空值运算符，如果是null值也可以调用ToString()方法了，这时程序什么都不会输出，例如：

1 int? iValue = 10;2 Console.WriteLine(iValue?.ToString());//不需要判断是否为空3 string name = null;4 Console.WriteLine(name?.ToString()); // 程序不会报错，也不会输出任何值5 Console.WriteLine("空值运算符");

 结果：



这样就不需要像以前一样先判断一下变量是不是null值了。

再查找：

## **Null条件运算符**

使用方法:

Customer customer = new Customer();  
 string name3 = customer?.Name;

等同于：

Customer customer = new Customer();if (customer1 != null)

{

string name = customer1.Name;

}

可以和??组合起来使用：

if (customer?.Face2()??false)

还可以2个一起用：

int? Length = customer?.Name?.Length;

也可以方法调用：

customer?.Face();

这个语法糖的**目的是在对象使用前检查是否为null。**如果对象为空，则赋值给变量为空值，所以例子中需要一个可以为空的int类型、即int?。

如果对象不为空，则调用对象的成员取值，并赋值给变量。

代码中是用到了这么一段：

return Json(new { retValue.code, retValue.msg, data = retValue.data?.DataList.Select(p => p.ExamineType).Distinct().ToList() });

1.5.2

第二个调用：

<http://localhost:9500/Doctor/Common/GetDataDicInfoList>

依然是post请求



参数：

{"keys":["StayInsu"]}

返回的response：

{

"data": {

"DataList": [

{

"LookupKey": "StayInsu",

"DicName": "门诊",

"DicCode": "1000",

"DicExtend": null

},

{

"LookupKey": "StayInsu",

"DicName": "住院",

"DicCode": "2000",

"DicExtend": null

},

{

"LookupKey": "StayInsu",

"DicName": "急诊",

"DicCode": "3000",

"DicExtend": null

},

{

"LookupKey": "StayInsu",

"DicName": "体检",

"DicCode": "4000",

"DicExtend": ""

},

{

"LookupKey": "StayInsu",

"DicName": "VIP",

"DicCode": "5000",

"DicExtend": null

}

]

},

"code": 0,

"msg": "获取数据字典成功"

}

看样子就是动态拼接前台页面。

vs中执行：

#region 根据Key获取数据字典列表

/// <summary>

/// 根据Key获取数据字典列表

/// </summary>

/// <param name="keys"></param>

/// <returns></returns>

[HttpPost]

public JsonResult GetDataDicInfoList(List<string> keys)

{

return Json(\_commonClient.Invoke(p => p.GetDataDicInfoList(keys)));

}

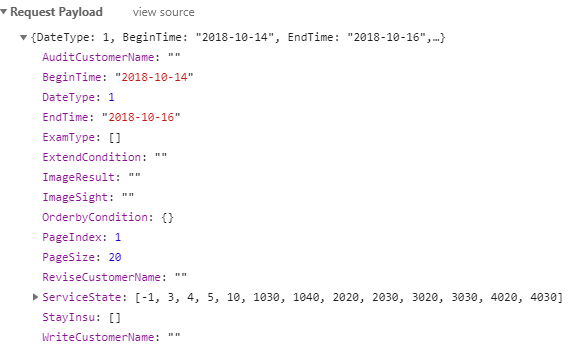
1.5.3：

第三个方法：



<http://localhost:9500/Doctor/Doctor/GetApplyList>

Request Payload居然这么多：



1. {DateType: 1, BeginTime: "2018-10-14", EndTime: "2018-10-16",…}
   1. AuditCustomerName: ""
   2. BeginTime: "2018-10-14"
   3. DateType: 1
   4. EndTime: "2018-10-16"
   5. ExamType: []
   6. ExtendCondition: ""
   7. ImageResult: ""
   8. ImageSight: ""
   9. OrderbyCondition: {}
   10. PageIndex: 1
   11. PageSize: 20
   12. ReviseCustomerName: ""
   13. ServiceState: [-1, 3, 4, 5, 10, 1030, 1040, 2020, 2030, 3020, 3030, 4020, 4030]
   14. StayInsu: []
   15. WriteCustomerName: ""

这个就是查询的条件，然后服务器处理，返回查询的结果：

response：

{

"data": {

"DataList": [

{

"PatientName": "hua",

"PatientSex": "男",

"Age": 33,

"AgeUnit": "岁",

"PatientBirthday": null,

"PatientId": "333",

"ServiceId": "b6c02faf-aa2b-4f7d-a8a2-4b7c1be82efe",

"ServiceCenterId": 39,

"ServiceCenterName": "非公医疗机构服务中心",

"ClinicDiagnose": null,

"ExamineType": "CT",

"ExamineBodyPart": "胸口",

"CheckItem": "肺部是否受损",

"RequestHospitalName": "平湖市第一人民医院",

"CustomerName": "洪启晨",

"RequestDate": "2018-10-16 13:49:47",

"RegisterDate": null,

"OccupyCustomerName": null,

"ServiceState": 4030,

"IsOccupy": null,

"OccupyUID": null,

"StayInsu": "门诊",

"MedRecNo": null,

"AccessionNumber": "4444",

"WriteDoctorName": "王专家",

"WriteTime": "2018-10-16 14:43:28",

"AuditUserName": "王专家",

"AuditTime": "2018-10-16 14:47:58",

"ReviseUserName": "王专家",

"ReviseTime": "2018-10-16 14:52:57",

"IsSign": false,

"IsPrint": null,

"IsToConsult": null,

"Qualitied": true,

"GradeLevel": "良"

}

]

},

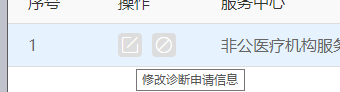
"code": 0,

"msg": "成功"

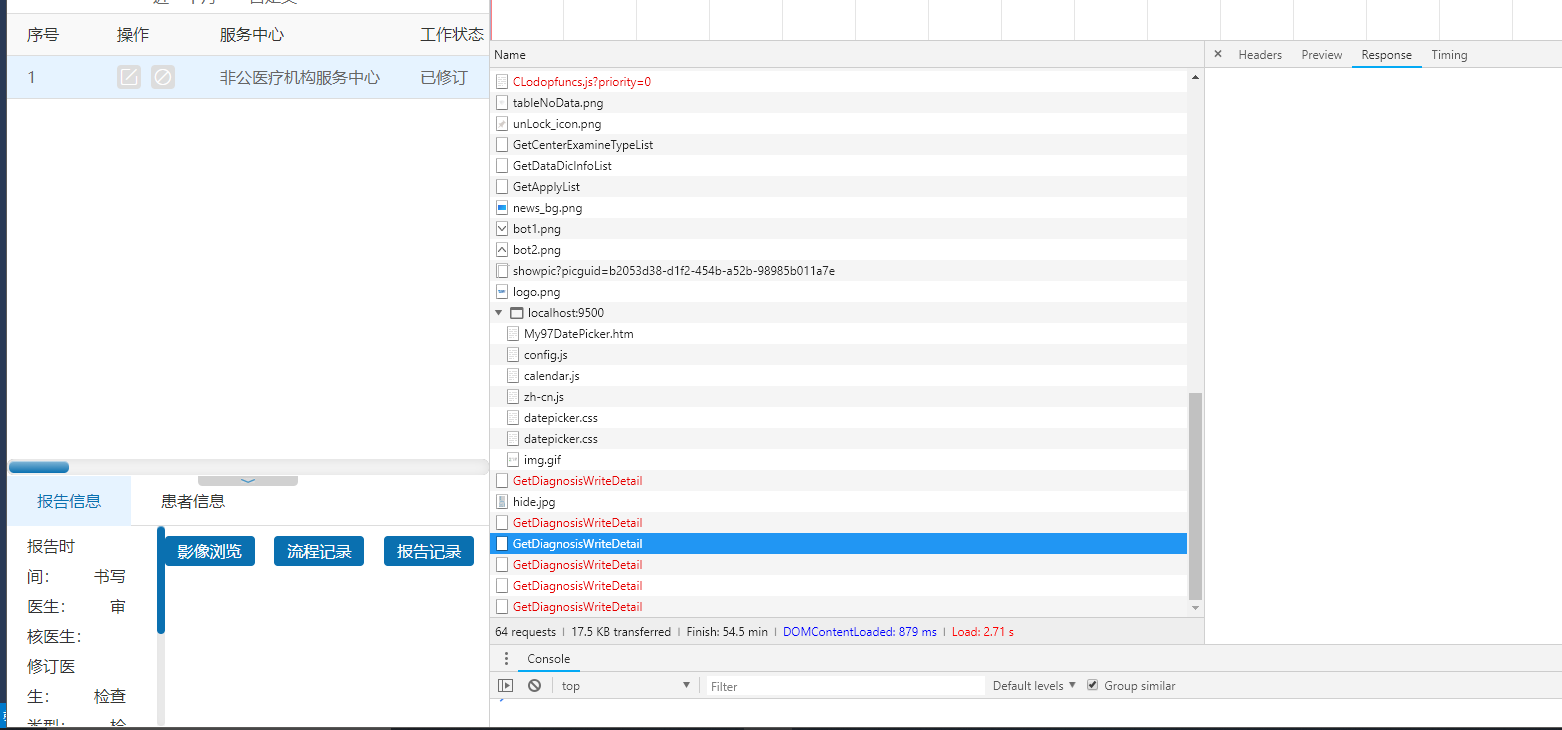
}

2.

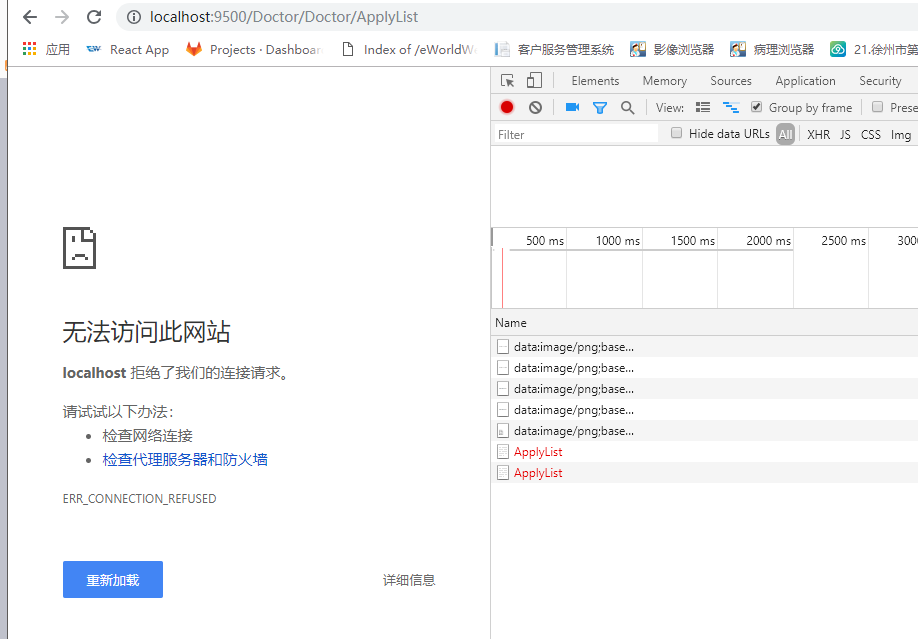
当点击这里的按钮，触发事件。



发现上面的属于1号申请的框框，只要点击，就触发一个获取详细信息的事件。但都是红色的，也许我已经过期了，当前登录的账号。



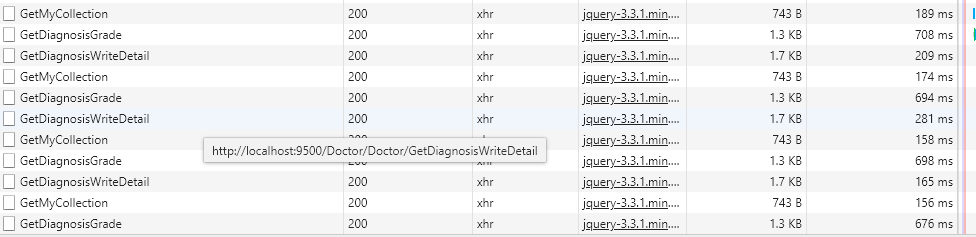
原来原因是因为我前面vs报错，然后搞了，没有把网站驾到iis上。



重新架起来之后就正常了。



但是每点一次，都向后台发送了请求，返回的结果是一样的。



三个方法：

（1）第一个



http://localhost:9500/Doctor/Doctor/GetDiagnosisWriteDetail

传递的参数：



{"ServiceId":"b6c02faf-aa2b-4f7d-a8a2-4b7c1be82efe"}

返回的response：

{

"data": {

"GPatientId": "b6f4a2f0-718c-4562-b001-f223048fea32",

"PatientName": "hua",

"PatientSex": "男",

"Age": 33,

"AgeUnit": "岁",

"PatientPhone": null,

"IdNumber": null,

"Address": null,

"PatientBirthday": null,

"ServiceId": "b6c02faf-aa2b-4f7d-a8a2-4b7c1be82efe",

"PatientDesc": null,

"MedicalRecord": null,

"ClinicDiagnose": null,

"PerDiagnosisRemark": null,

"Attention": null,

"PatientSigns": null,

"AuxiliaryDiagnosis": null,

"Allergy": null,

"IsPregnant": null,

"ExamineType": "CT",

"ExamineBodyPart": "胸口",

"CheckItem": "肺部是否受损",

"ExamCost": null,

"CheckEquipment": null,

"StayInsu": "门诊",

"RequestHospitalCode": "17330",

"RequestHospitalName": "平湖市第一人民医院",

"RequsetDeptName": "胸内科",

"CustomerName": "洪启晨",

"RequestDate": "2018-10-16 13:49:47",

"RequestPhone": "15940040040",

"RegisterDoc": null,

"RegisterDate": null,

"ExamineDoc": null,

"ExamineDate": null,

"ExamineTechnology": null,

"OriginalWriterName": null,

"ServiceCenterId": 39,

"ServiceCenterName": "非公医疗机构服务中心",

"ServiceState": 4030,

"OccupyUID": null,

"BedNumber": null,

"MedRecNo": null,

"InpatientNumber": null,

"AccessionNumber": "4444",

"IsToConsult": null,

"ImageSight": "未见阴影。第二版。",

"ImageDiagnosis": "健康",

"WriteCustomerGuid": "cdc5f939-fc42-4297-a4d5-05788a4a31e0",

"WriteCustomerName": "王专家",

"WriteTime": "2018-10-16 14:43:28",

"AuditCustomerGuid": "cdc5f939-fc42-4297-a4d5-05788a4a31e0",

"AuditCustomerName": "王专家",

"AuditTime": "2018-10-16 14:47:58",

"ReviseCustomerName": "王专家",

"ReviseTime": "2018-10-16 14:52:57",

"IsMasculine": false,

"Qualitied": true,

"IsForceQuality": false,

"DetectionProbability": null,

"DetectionPulmonaryNodule": null

},

"code": 0,

"msg": "成功"

}

控制器中对应的方法：

#region 获取影像诊断书写详情

/// <summary>

/// 获取影像诊断书写详情

/// </summary>

/// <param name="serviceId"></param>

/// <returns></returns>

[HttpPost]

public JsonResult GetDiagnosisWriteDetail(Guid serviceId)

{

return Json(\_diagnosisService.Invoke(m => m.GetDiagnosisWriteDetail(serviceId)));

}

#endregion

（2）第二个



<http://localhost:9500/Doctor/Remote/GetMyCollection>

传递参数：



{"CollectionId":"b6c02faf-aa2b-4f7d-a8a2-4b7c1be82efe","CollectionKind":111}

返回的response：

{

"data": null,

"code": 0,

"msg": "成功"

}

vs中的控制器方法：

/// 获取我的收藏

/// </summary>

/// <param name="CollectionId">业务唯一号</param>

/// <param name="CollectionKind">业务类型</param>

/// <returns></returns>

/// <remarks>add by fengmingliang 20170718</remarks>

[AcceptVerbs(HttpVerbs.Post)]

[HttpPost]

public ActionResult GetMyCollection(Guid? CollectionId, int CollectionKind)

{

var result = \_crmClient.Invoke(t => t.GetMyCollection(new MyCollectionParam

{

CustomerGuid = CustomerIdentity.CustomerGuid,

CollectionId = CollectionId,

CollectionKind = CollectionKind

}));

return Json(result);

}

（3）第三个

GetDiagnosisGrade方法，对应的是Quality控制器。

http://localhost:9500/Quality/GetDiagnosisGrade

post请求：

POST /Quality/GetDiagnosisGrade HTTP/1.1

返回的是json：

Accept: application/json, text/javascript, \*/\*; q=0.01

Content-Type: application/json

引用页：

Referer: <http://localhost:9500/Doctor/Doctor/ApplyList>

cookie是：

Cookie:

\_ga=GA1.1.1911654723.1533896508;

ASP.NET\_SessionId=xbtveqzaqccyxc0mmjoq5b0y;

Nop.customer=af22c4c1-ecb7-4254-814f-4e13032ebd24;

errormsg=;

eWorldToken=2895F564057DC4D9BA74293B16B78F2711B90EF92294A029FFD7E07995193A17D504268EB5B6135E836CB1570F350AB7B65817F85078B32F11AE990CFCD44FF2B3D8E228B331570FA14929B5FBAD5A6999AFB03EA9A6F9FE0FEF20DAABD5A4F1ECA5308AB4D4080568C6076BD52BE05EBFACDF9953B8CA84

Request Payload：



这个应该就是影像的资料。

我去数据库中查查看。

响应Response是一段json，然后前端接收，获取证据想要的，然后放到前端的模板页中，去填坑：

{

"data": {

"ServiceId": "b6c02faf-aa2b-4f7d-a8a2-4b7c1be82efe",

"Kind": 111,

"Modality": "CT",

"MedRecNo": null,

"PatientId": "333",

"AccessionNumber": "4444",

"PatientName": null,

"ExamineDocID": null,

"ExamineDoc": null,

"ExamineDate": null,

"ReportDocID": null,

"ReportCustomerGuid": "cdc5f939-fc42-4297-a4d5-05788a4a31e0",

"ReportCustomerName": "王专家",

"RequestCustomerGuid": "af22c4c1-ecb7-4254-814f-4e13032ebd24",

"RequestCustomerName": "洪启晨",

"RequestHospitalCode": "17330",

"RequestHospitalName": "平湖市第一人民医院",

"ServiceCenterId": 39,

"ServiceCenterName": "非公医疗机构服务中心",

"Grade": 85,

"Level": "良",

"GradeReason": "摄影体位 (不标准),扣15分;",

"GradeDate": "2018-10-16 14:49:38",

"Memo": null,

"Id": 119

},

"code": 0,

"msg": "操作成功"

}

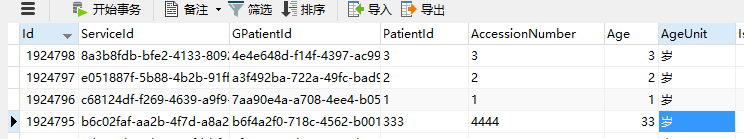
数据库中找到这条记录了：

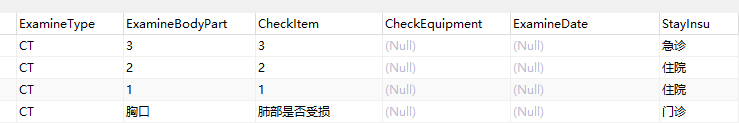
（1）

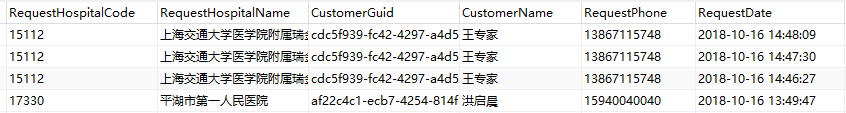
ImageRequest中：



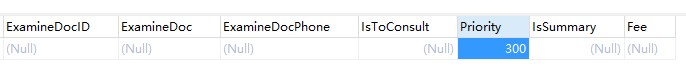
都是第四条：











priority：优先级的问题。

（2）

接下来在Patient表中找到了hua这位同学：



就填了些必填的内容，其余均为null。





这里发现了缺陷，年龄居然不是在病人Patient表中，而是在上面的ImageRequest中。





（3）

在评分表里：

DiagnosisGrade









因为涉及多个表的数据，所以查询的时候肯定要连表，把这些数据都查询出来再以json的形式给前台。

（4）终于他娘地找到诊断结果了。

在ImageRequestReply表中：



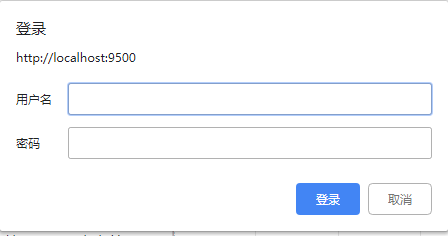








又断开了，提示登录



但发现是登录不上的。

3.

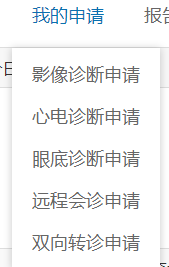
在http://localhost:9500/Quality/GetDiagnosisGrade请求之后紧接着请求，

<http://localhost:9500/Doctor/Doctor/GetDiagnosisWriteDetail>

应该是内部ajax继续调用获取地址，

仍旧是post请求，请求继续携带这个参数。





==========================================

不知道该放哪里了，喻永红说的东西，就塞这里好了

继续说起专家的工作界面：



这里会涉及专家的权限问题，这些权限，都是机构后台给配置好的。给了什么权限，就有什么权限，就能做什么动作。



==========================================

发现的问题（上面也都有描述，详细的也在本文档的上部）：

1.

又要重新登录了，所以这里的cookie要改，他每次使用网页，用到用户校验的地方，就要重新写一次，然后给保存在客户端那边。

放在我们这边的Redis上是最好的，每次过来都查一次。

反正目前，这个记录登录用户的问题，可以完善一些，后期我再测试。

当然也有可能是别人登陆这个账号了。

2.