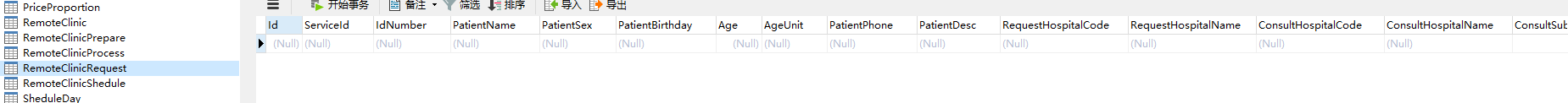
流程探索——门诊

1.

1.1 患者预约申请表



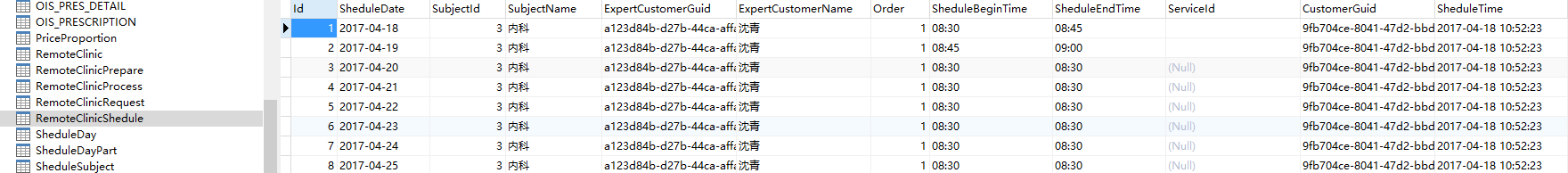
但是表里并没有数据



1.2 远程门诊预约队列

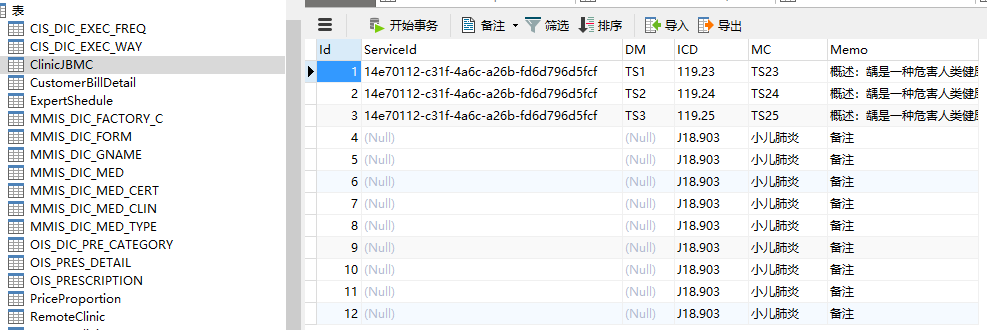
建立专家和申请人（应该是医生）之间的一对一关系





1.3 远程门诊疾病名称





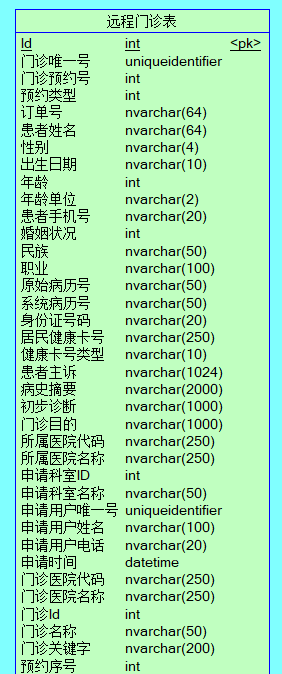
数据不多，应该是用的不多

1.4 门诊表

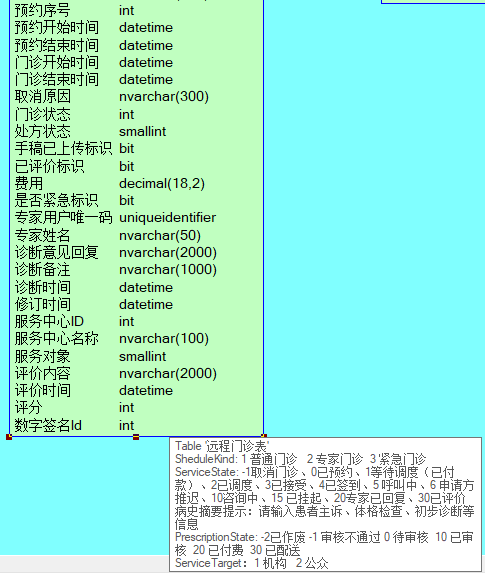
这张表就建立起了患者，普通门诊医生，远程门诊医生，三者之间的关系，三者的guid都确定，外加该条门诊的guid，也就确定了三者之间当前这唯一一条数据。

这种就像是一张汇总表，所需要的数据都有了，再也不用去连三张表查询两个医生，患者，三者间所有的各自数据了。

结果通过接口发现，这里涉及对于是否是专家的判断，如果是，那么当前用户guid对应的就是专家的guid，不然就是对应的customerGuid。

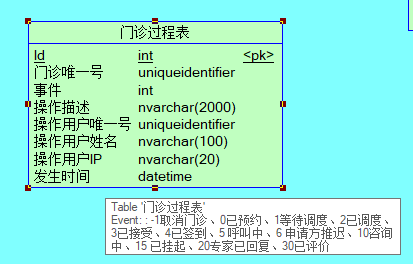






1.5 门诊过程表

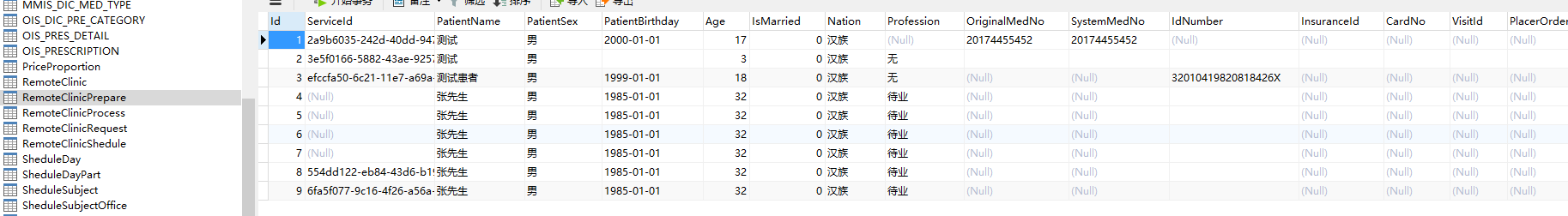
门诊唯一号可以精确到哪一条门诊申请，然后再跟操作的专家guid对应上。形成精确的一对一关系。





1.6 远程门诊申请接口缓存表

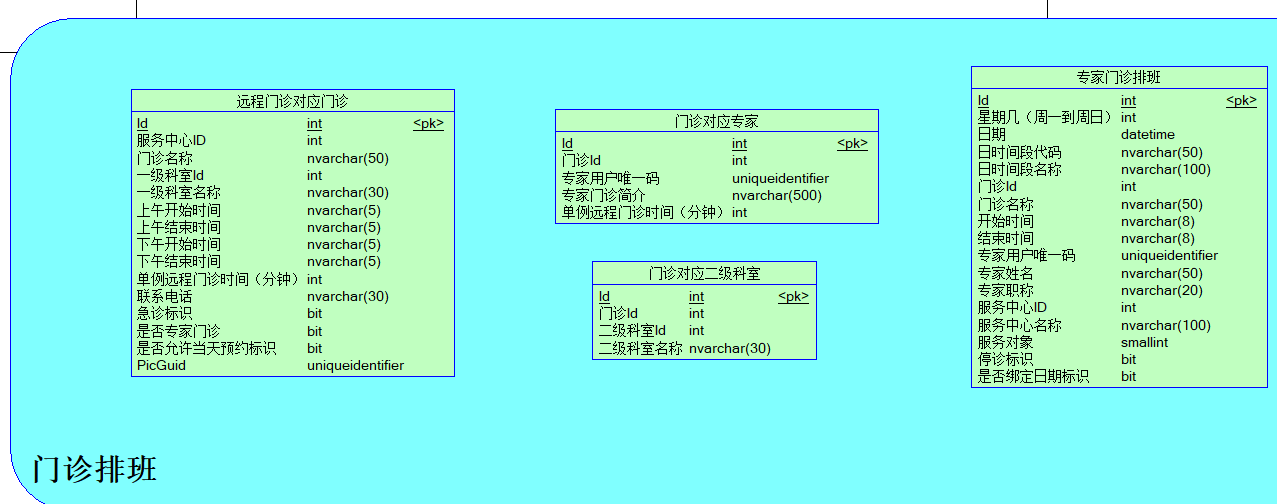




没上面数据，应该也是没怎么在使用。

2.

门诊排版：

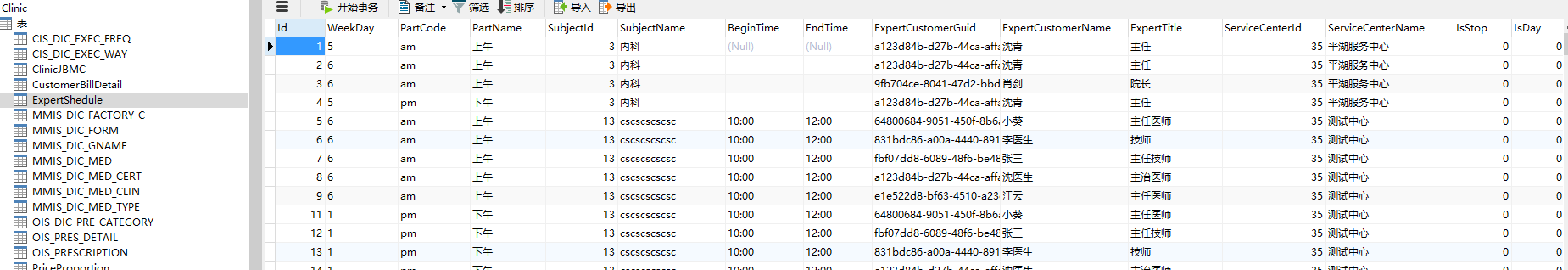


2.1 专家门诊排班

这里就是设定了专家具体在其可提供服务的哪家服务中心中，哪个门诊，确定了排班。这样该服务中心下的签约机构，就都可以申请该专家来提供服务了。

也就是提供了 会员专家——服务机构——当前的签约机构（类似卫生院） 三者间的可绑定关系。

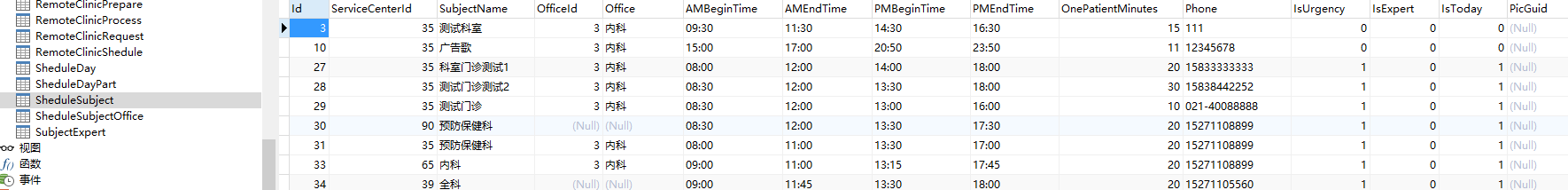




2.2 远程门诊对应门诊

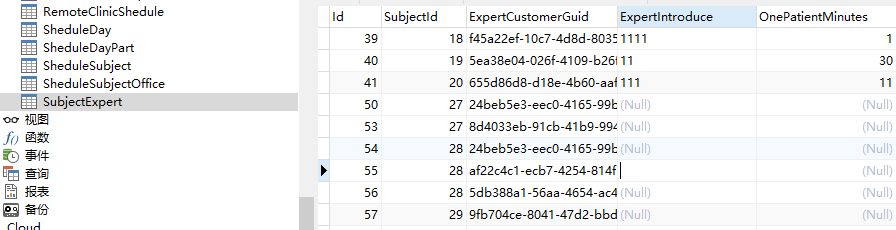
感觉上面的门诊表一张，就搞定了所有的对应关系。



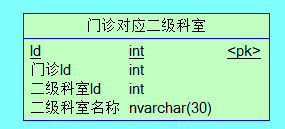


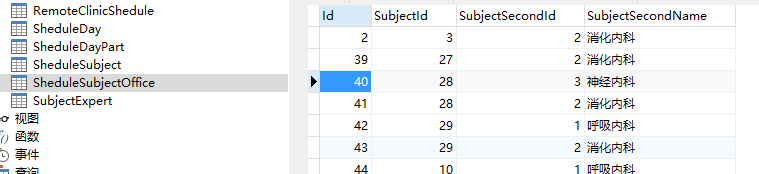
2.3 门诊对应专家





2.4 门诊对应二级科室





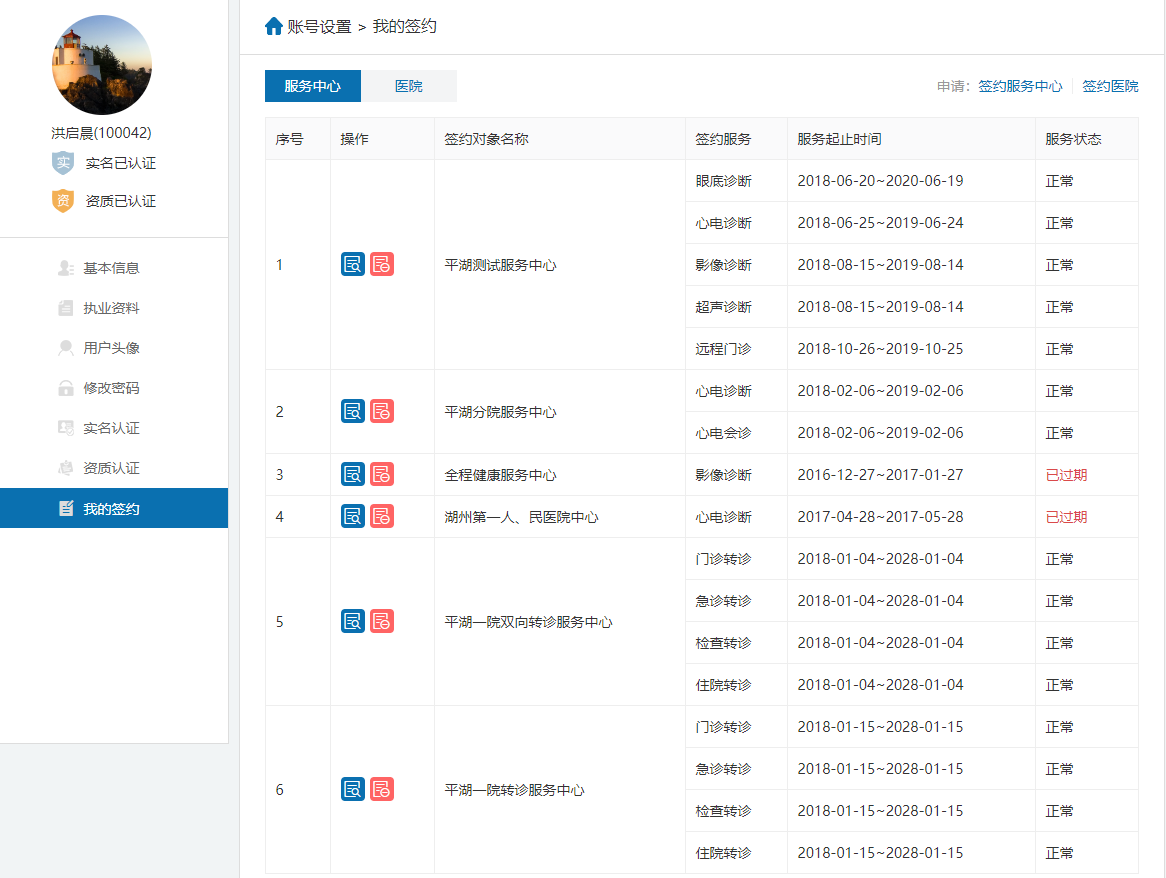
3.

4.

登陆查看：

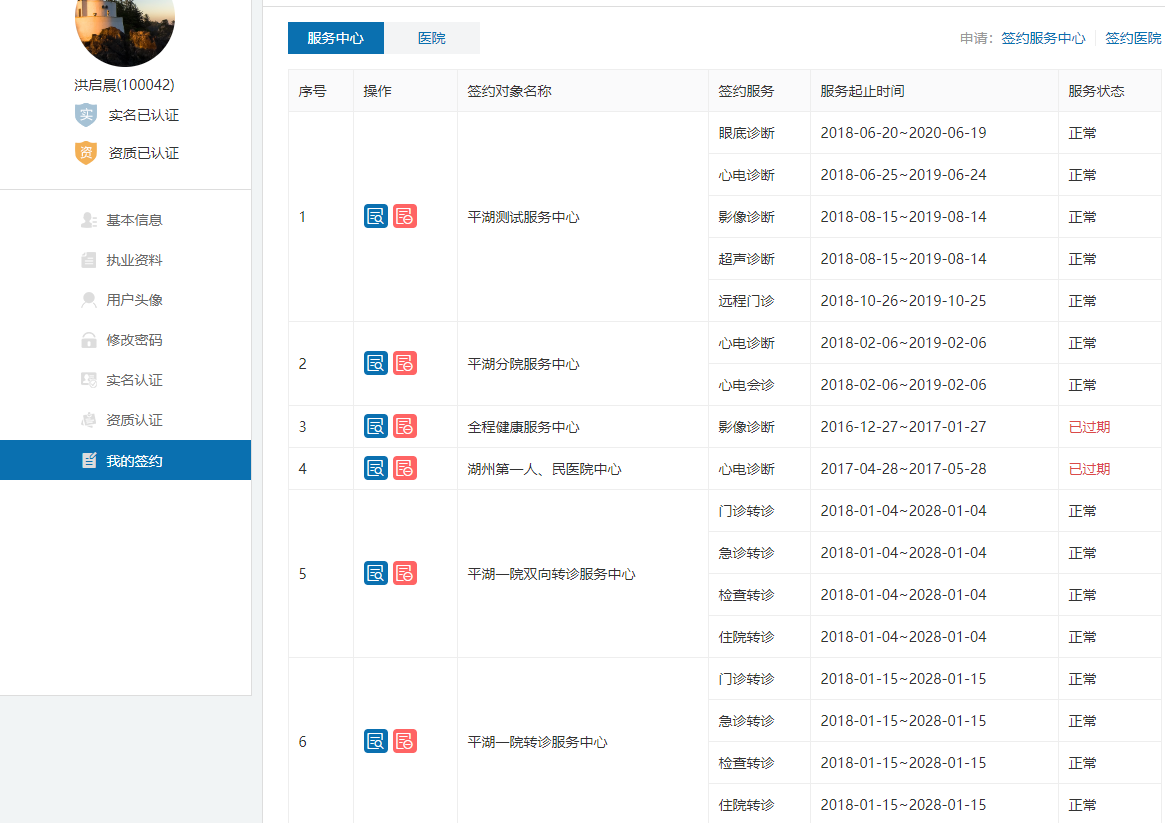
1.

登陆之后，可查看和修改各种个人信息，还看查看签约信息。



点击医院和签约医院无反应，应该没做。

点击签约服务中心



这边调的不是我们这边的接口：

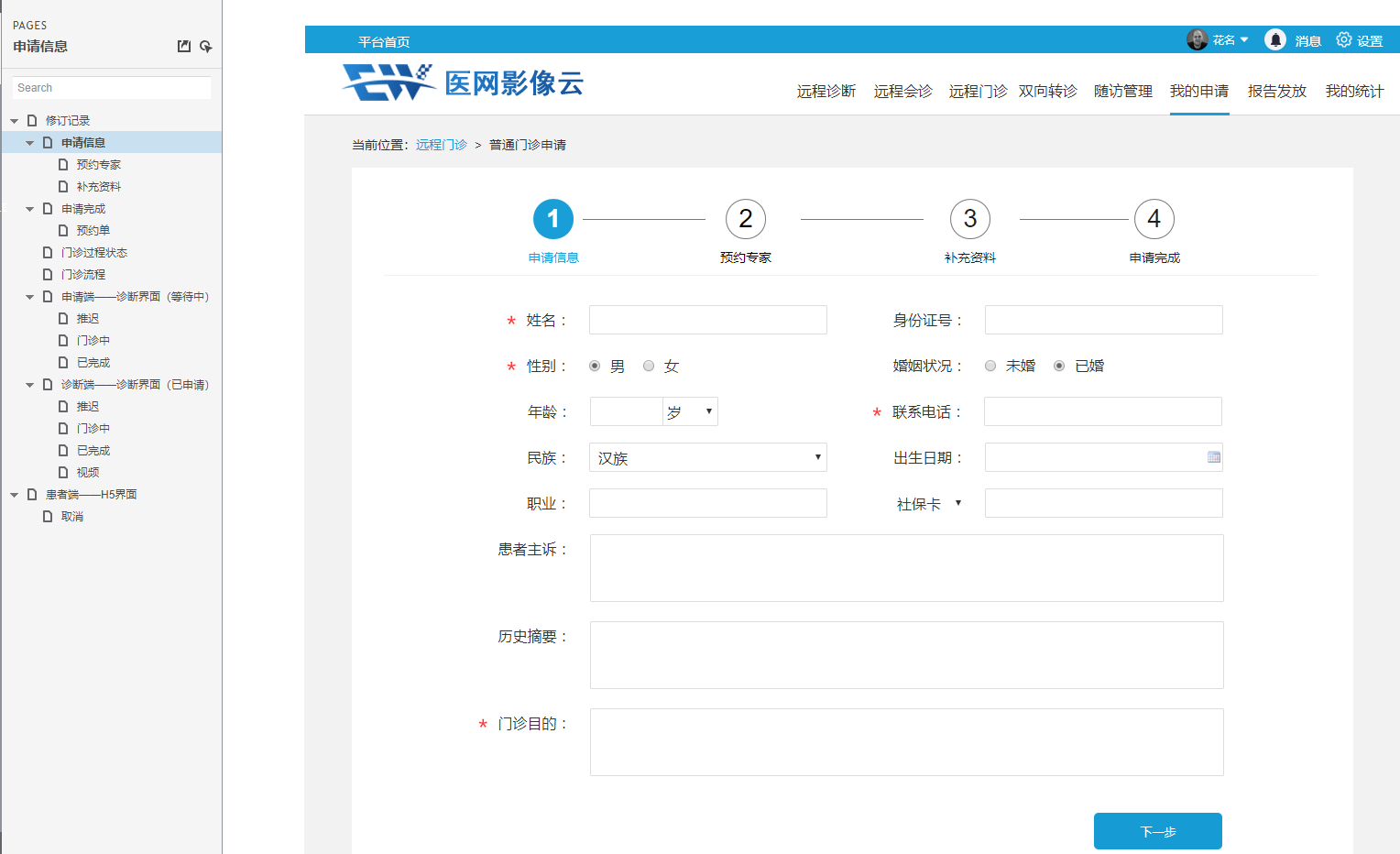
http://218.108.111.166:9100/UCenter/service/ServiceCenterSignList

2.

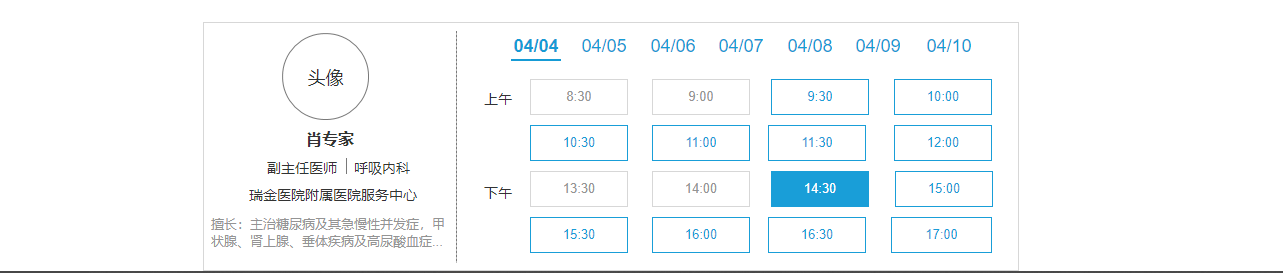
3.

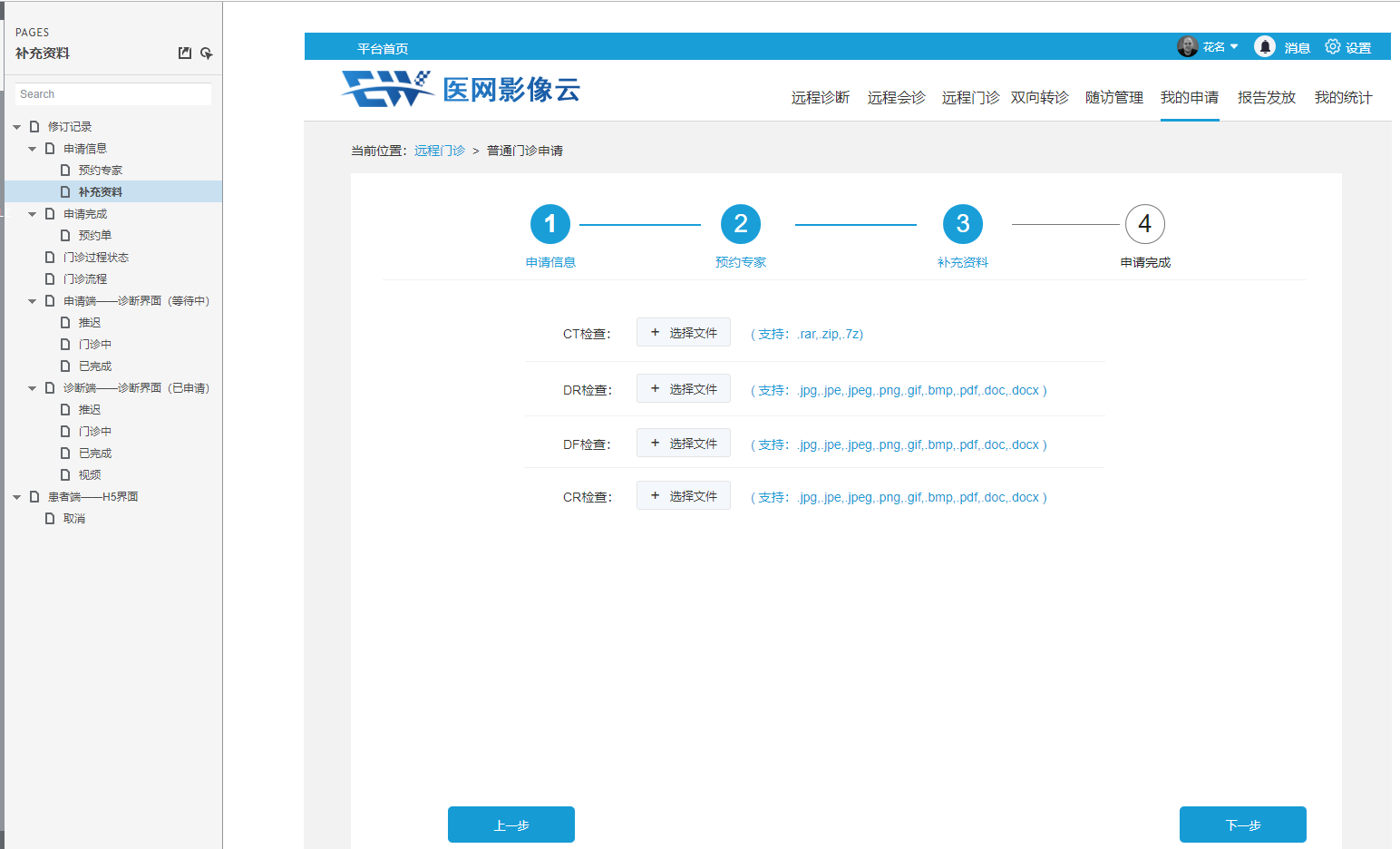
4.

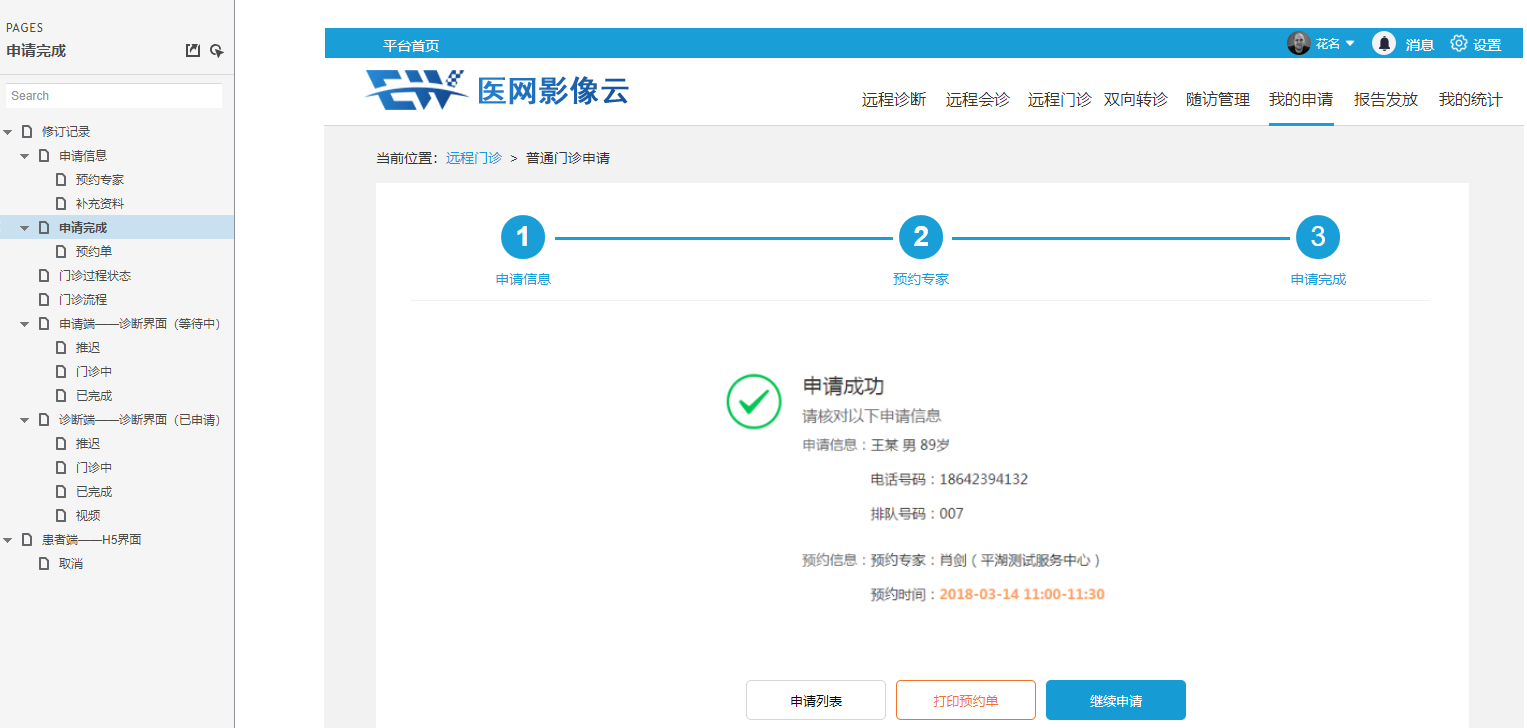




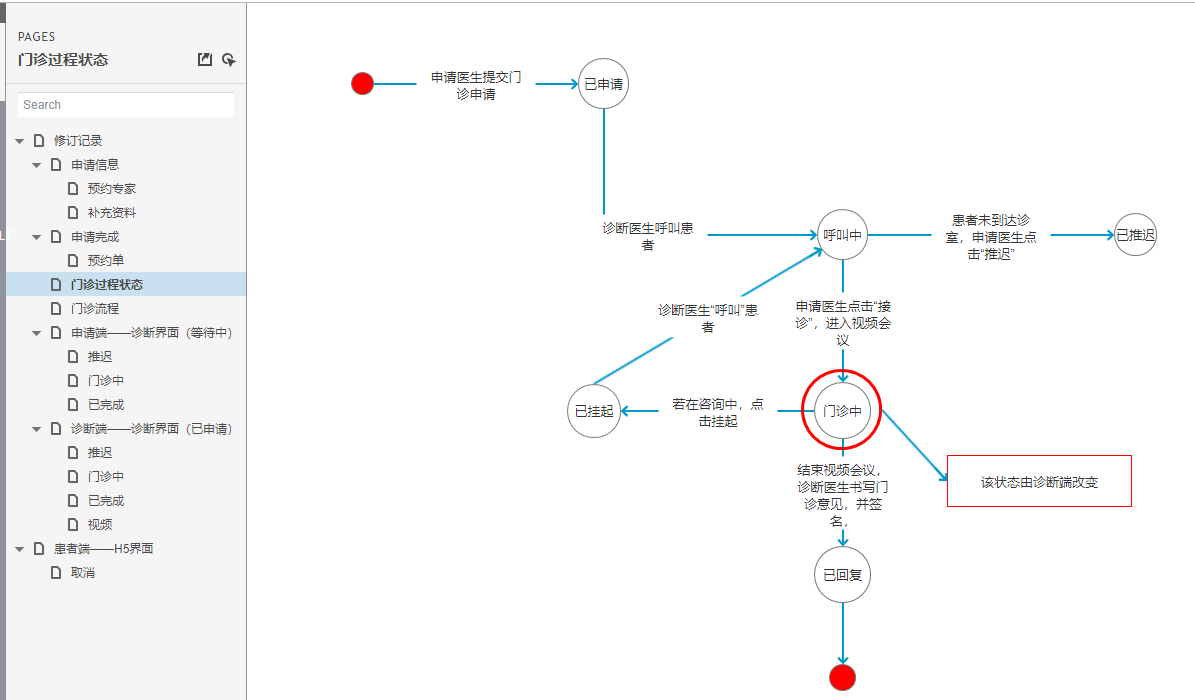


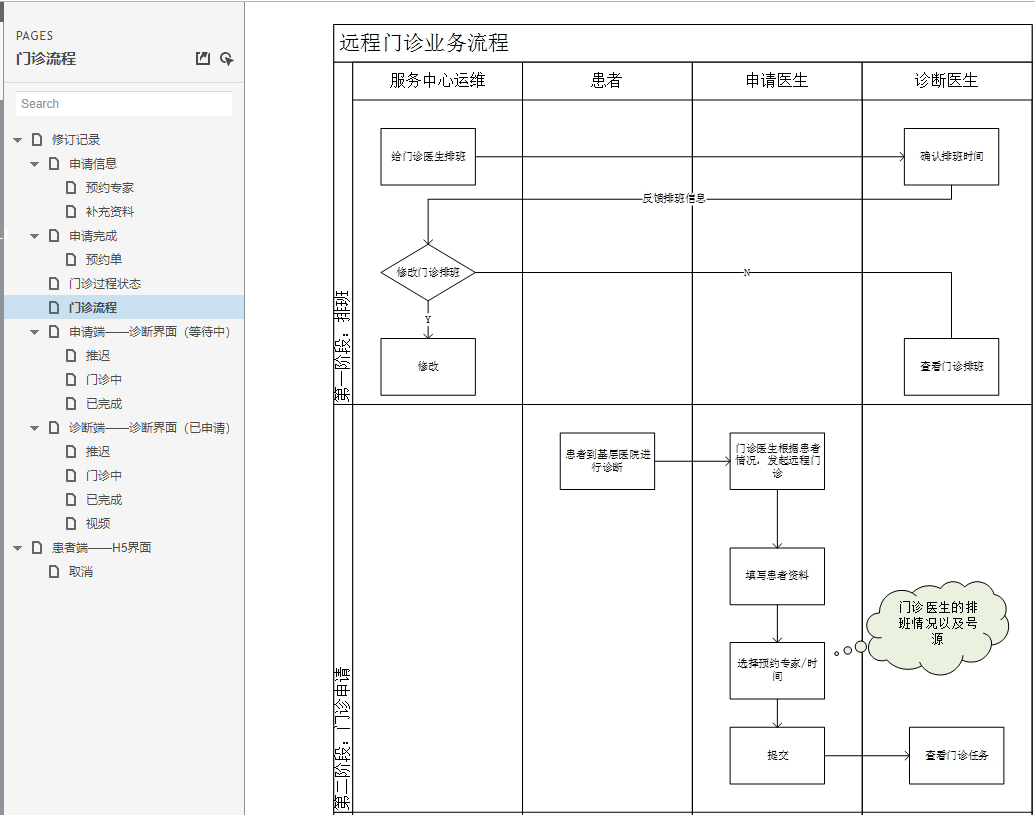


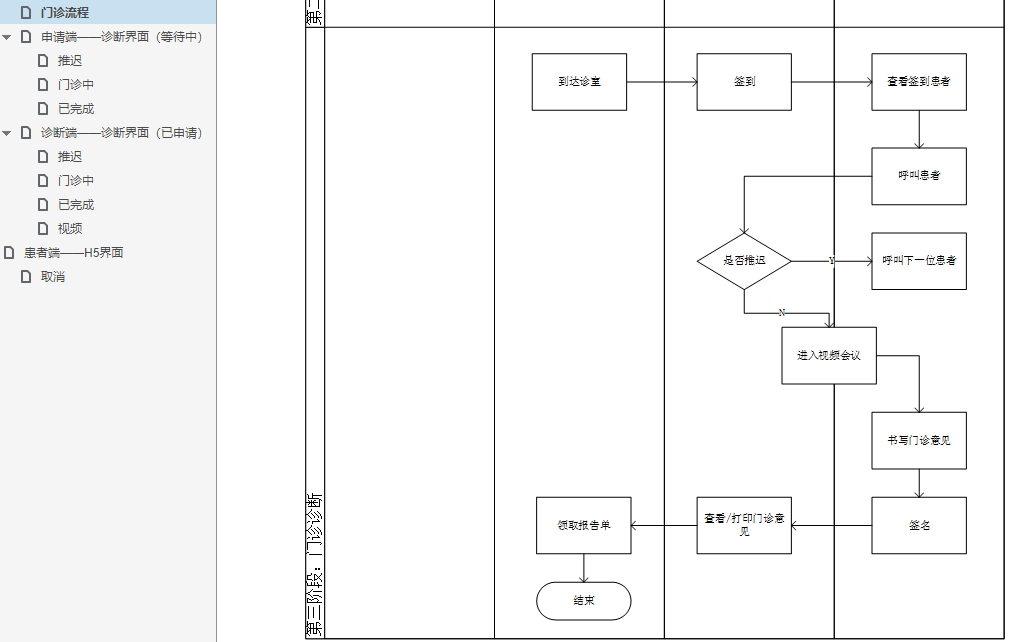


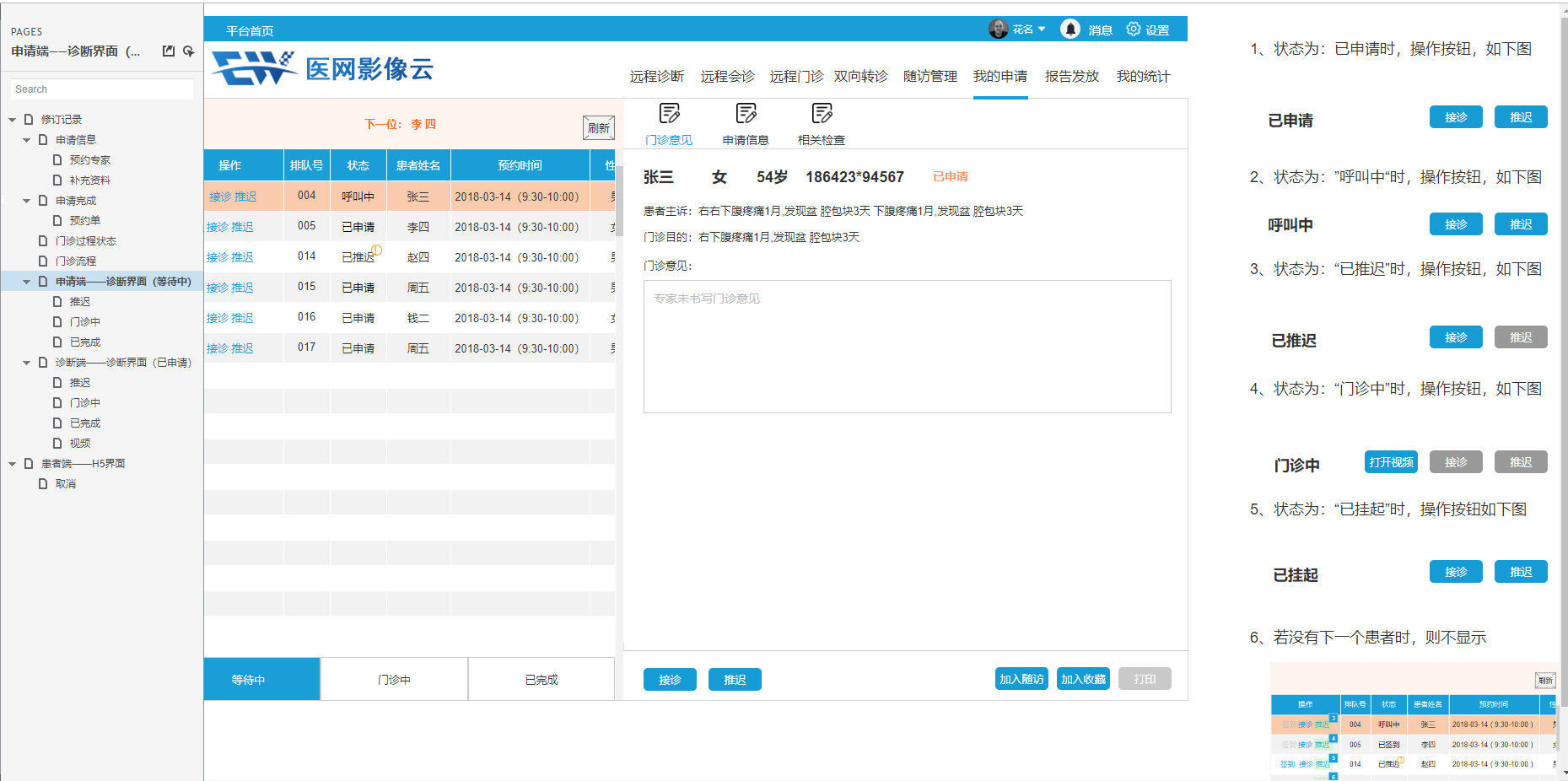


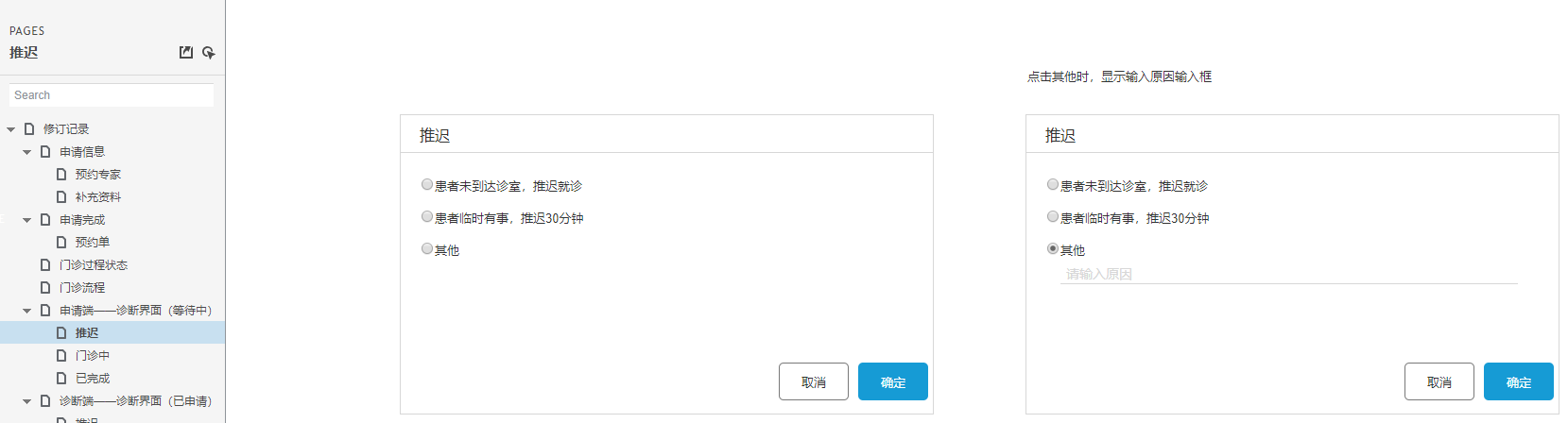






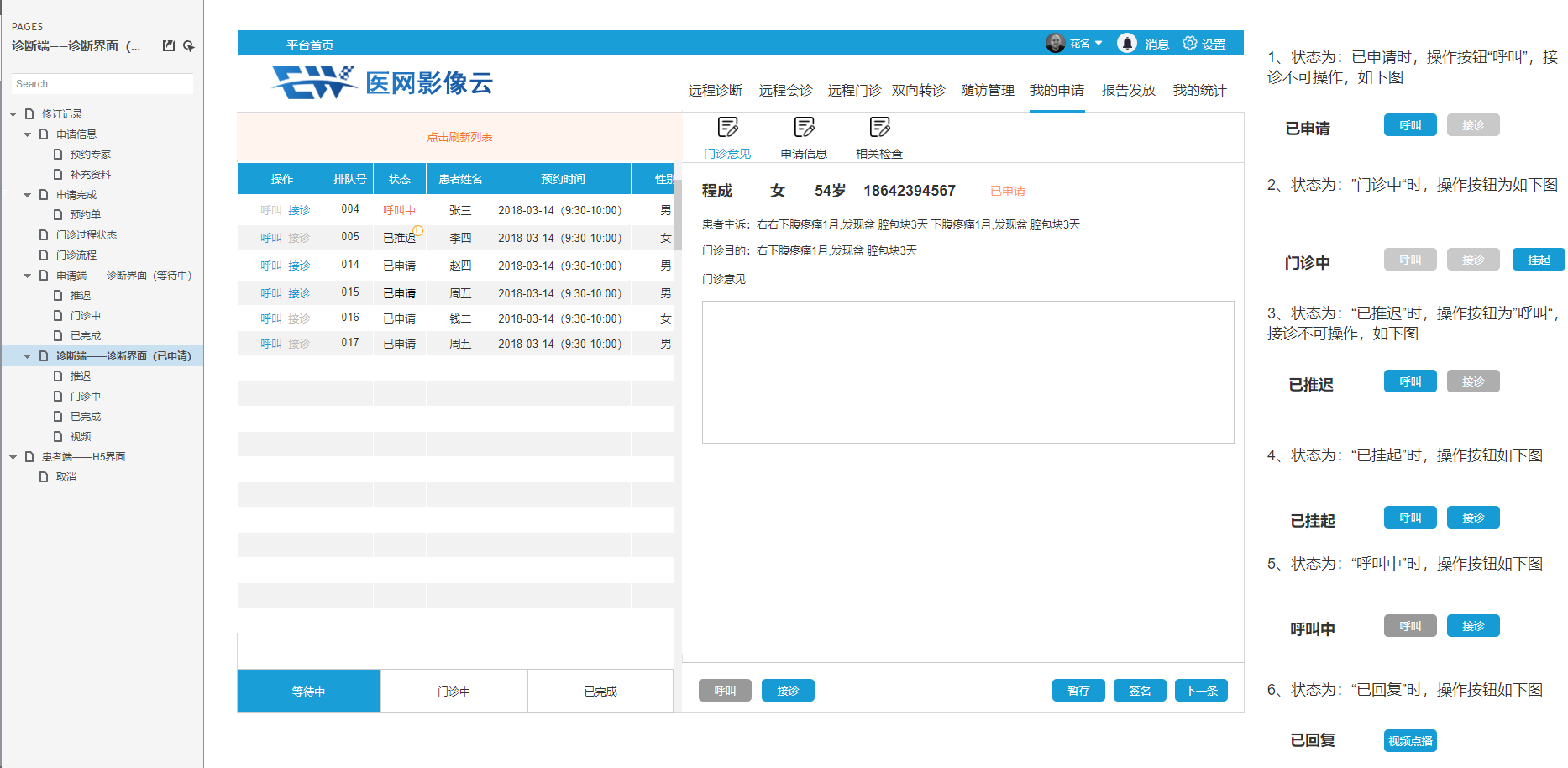


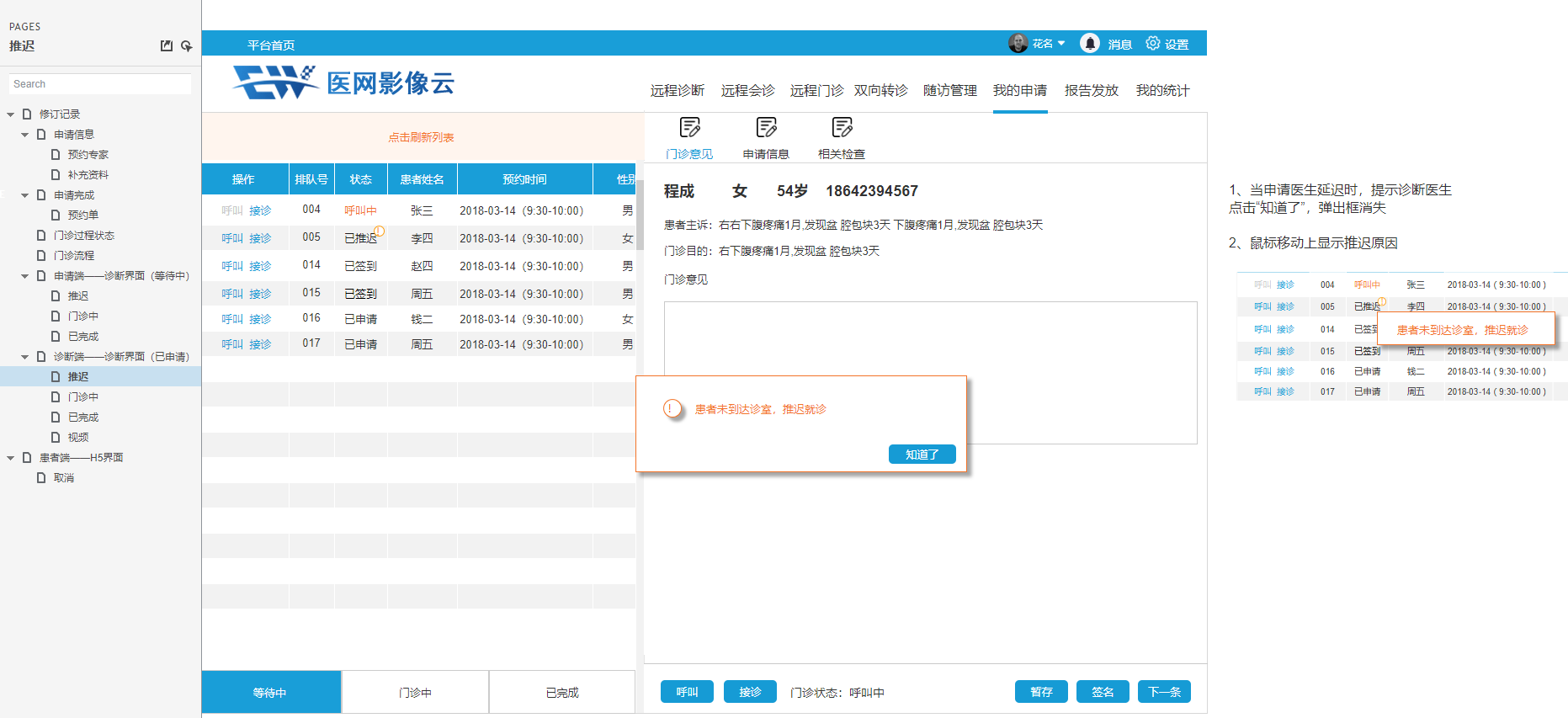


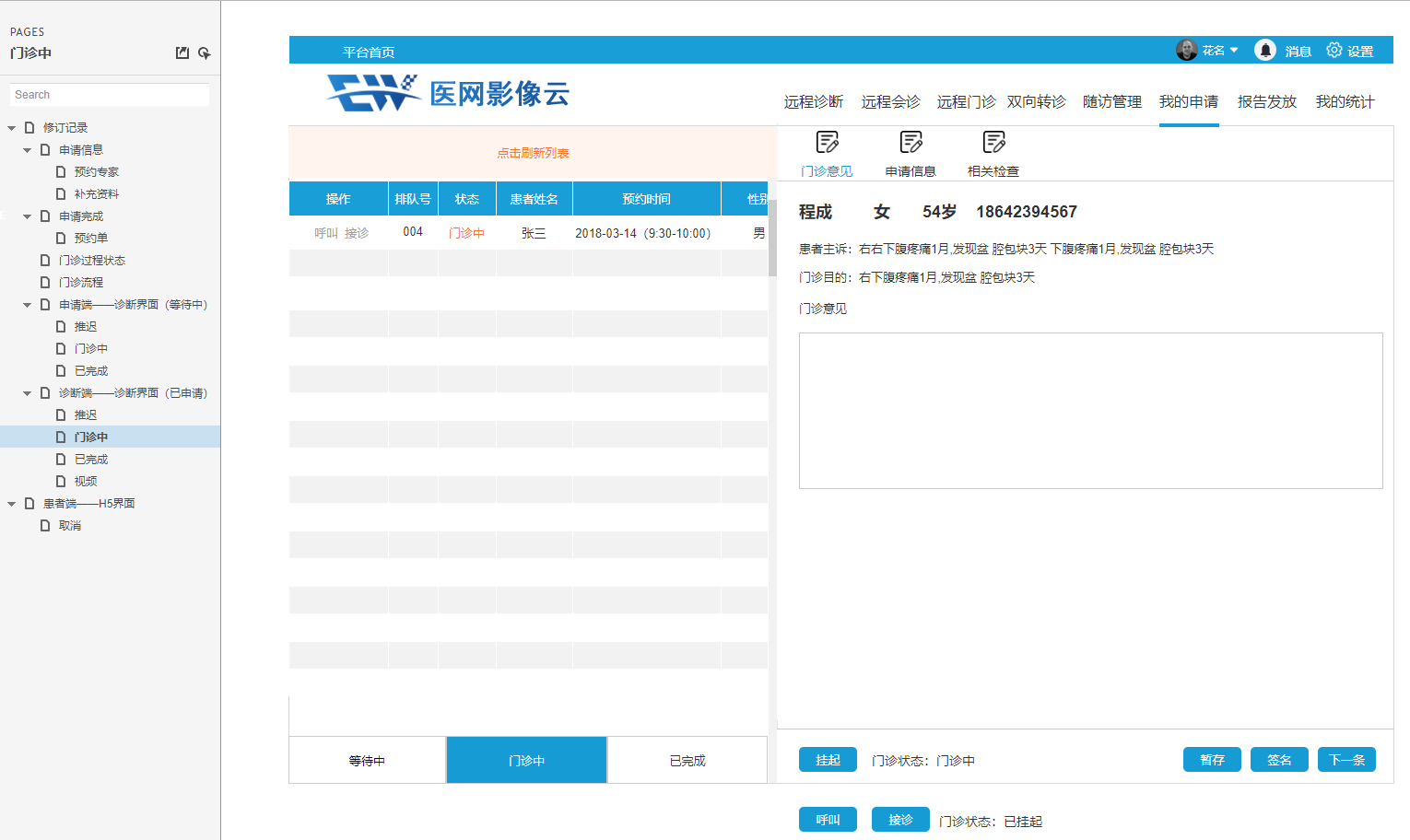


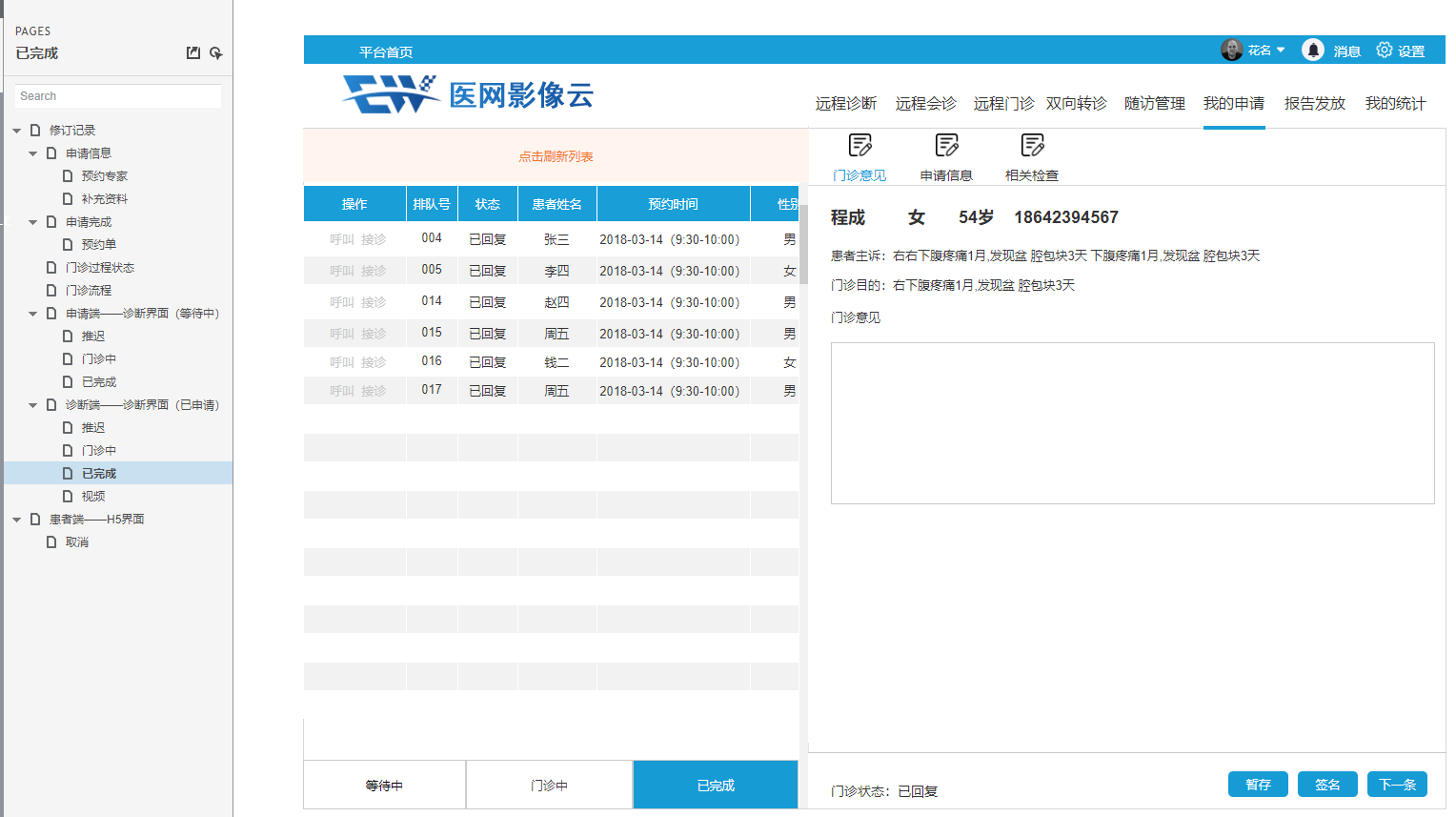


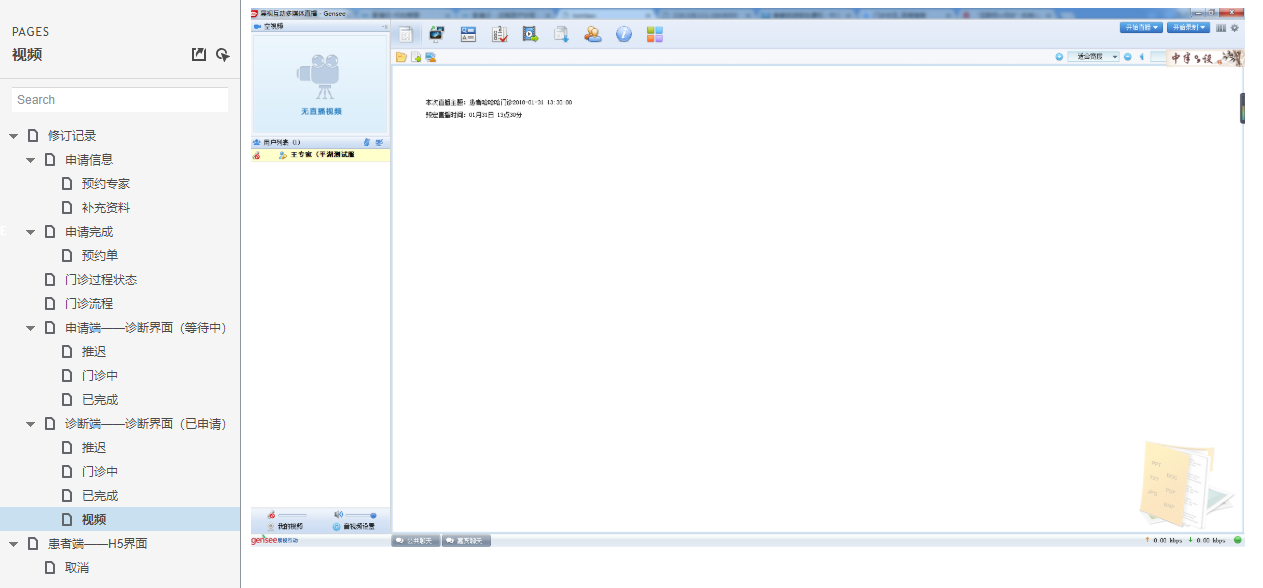












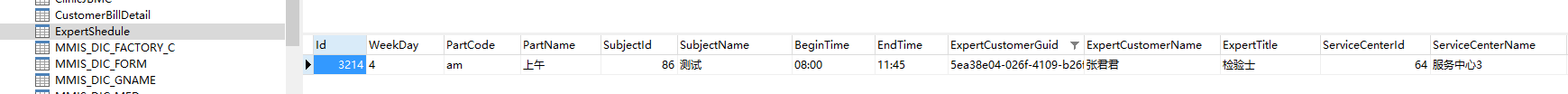




正式开始走流程：

1.

登陆洪启晨的账号，没有排班消息，但登陆张君是有排班消息的，而且在专家排班表中也找到了这条数据。



进入远程门诊页面：

<http://192.168.1.202:9500/Doctor/Clinic/ExpertIndex>



对应

/// <summary>

/// 门诊专家首页

/// </summary>

/// <returns></returns>

public ActionResult ExpertIndex()

{

return View();

}

（1）



<http://localhost:9500/Doctor/Clinic/GetExpertClinicList>

入参：

{"BeginSheduleTime":"2018-11-08","EndSheduleTime":"2018-11-08","BeginRequestTime":null,"EndRequestTime":null,"PatientName":null,"SheduleKind":[],"ServiceState":["-1","0","1","2","3","5","10","15","20","30"],"PageIndex":0,"PageSize":20,"SortInfo":{"SortField":"SheduleKind","SortType":"Desc"}}

其中的SheduleKind为空，当点击门诊类型后，就发生变化，再次请求后台



SheduleKind: [2, 3]

返回的结果：



一条，其实也就是查询 "ExpertCustomerGuid": "af22c4c1-ecb7-4254-814f-4e13032ebd24",也就是接收方专家的guid是当前登陆方的guid，所有数据就可以了。再根据所选择的数据进行筛选以及排序下。

对应：

{

"data": {

"PageInfo": {

"PageIndex": 0,

"PageSize": 20,

"TotalPages": 1,

"TotalCount": 1,

"HasPreviousPage": false,

"HasNextPage": false

},

"DataList": [

{

"Id": 0,

"ServiceId": "9f07ffed-6b80-45c9-8c5c-25e3b41cbfaa",

"RequestId": null,

"SheduleKind": 1,

"PatientName": "10-26-1",

"PatientSex": "男",

"PatientBirthday": null,

"Age": 1,

"AgeUnit": "岁",

"IsMarried": 0,

"Nation": null,

"Profession": null,

"OriginalMedNo": null,

"SystemMedNo": null,

"IdNumber": null,

"HealthCardNumber": null,

"HealthCardType": null,

"PatientDesc": null,

"MedSummary": null,

"Purpose": "1",

"RequestHospitalCode": null,

"RequestHospitalName": "上海交通大学医学院附属瑞金医院",

"RequestOfficeId": null,

"RequestOfficeName": null,

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

"CustomerName": "王专家",

"RequestPhone": null,

"PatientPhone": "15271007788",

"RequestDate": "2018-10-26 11:31:54",

"ConsultHospitalCode": null,

"ConsultHospitalName": null,

"ConsultSubjectId": 59,

"ConsultSubjectName": "cs",

"KeyWord": null,

"Order": 231,

"SheduleBeginTime": "2018-10-26 11:50:00",

"SheduleEndTime": "2018-10-26 11:51:00",

"ConsultStartDate": "2018-10-26 11:36:33",

"ConsultEndDate": null,

"CancelReason": null,

"ServiceState": 10,

"IsScript": null,

"IsEvaluate": null,

"IsUrgency": null,

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

"ExpertCustomerName": "洪启晨",

"Diagnosis": null,

"DiagnosisMemo": null,

"DiagnosisTime": null,

"ReviseTime": null,

"ServiceCenterId": 35,

"ServiceCenterName": "平湖测试服务中心",

"AdviceContent": null,

"AdviceDate": null,

"Grade": null,

"IsCall": null,

"IsStop": null,

"FirstDiagnosis": null,

"ExpertPhone": null,

"DocSetJson": null

}

]

},

"code": 0,

"msg": "成功"

}

UI层：

#region 专家门诊任务

[HttpPost]

public ActionResult GetExpertClinicList(RemoteClinicSearhModel form) //, string serviceState, string serviceCenterId

{

#region 参数检查

var result = new BaseApiModel<ApiPageListModel<RemoteClinicModel>>();

if (form.ServiceCenterId == null || form.ServiceCenterId.Count > 0)

{

if (CustomerIdentity.OpenService.Count ==0)

{

result.code = ResultStatusCode.LACK\_PARA;

result.msg = "您未签约任何服务中心";

return Json(result);

}

从登陆的用户信息中查找出所有的，用户所开通的服务，并且对应到所有的服务中心Id。

form.ServiceCenterId = CustomerIdentity.OpenService.Select(s => s.ServiceCenterId).ToList();

}

#endregion

form.PageIndex = form.PageIndex <= 0 ? 0 : form.PageIndex;

form.PageSize = form.PageSize > 30 || form.PageSize <= 0 ? 30 : form.PageSize;

form.CustomerGuid = CustomerIdentity.CustomerGuid;

result = \_clinicClient.Invoke(s => s.GetExpertRemoteClinicList(form));

return Json(result);

}

BLL层的处理：

/// <summary>

/// 获取门诊专家远程门诊任务列表

/// </summary>

/// <param name="search">搜索条件</param>

/// <returns></returns>

public BaseApiModel<ApiPageListModel<RemoteClinicModel>> GetExpertRemoteClinicList(RemoteClinicSearhModel search)

{

var result = new BaseApiModel<ApiPageListModel<RemoteClinicModel>> { code = ResultStatusCode.OK, msg = "成功" };

try

{

#region 参数检查

if (!search.CustomerGuid.HasValue)

{

result.code = ResultStatusCode.LACK\_PARA;

result.msg = "门诊专家唯一号不能为空";

return result;

}

#endregion

这里设置为是专家列表，也就是当前登陆的用户Guid对应的是远程诊断表中的专家ExpertGuid。

search.IsExpert = true;

if (search.BeginSheduleTime.HasValue && search.EndSheduleTime.HasValue)

{

把这些所有的时间段制定都放在bll层，之前我放到dal层，就报错，说ef中不能处理这种事件表达式。对啊，能在bll层处理掉的逻辑，以及数值运算，都不要放到dal层中，因为会放到数据库中去执行逻辑判断和数值计算。

search.EndSheduleTime = search.EndSheduleTime.Value.AddDays(1);

}

if (search.EndRequestTime.HasValue && search.EndRequestTime != DateTime.MinValue)

{

search.EndRequestTime = search.EndRequestTime.Value.AddDays(1);

}

根据前端的查询入参，返回符合条件的门诊列表集合。查询的方式依旧在dal层根据各个参数是否存在而继续添加一个where。

我还是又想改造成lambda查询的冲动，当然要在bll层做参数是否存在的判断，如果存在就全部扔到lambda表达式里面去。

但是改造成lambda之后，当查询的参数变了，不会的，查询的参数来自于最原生的该集合，所以所有参数都是齐全的，只不过在bll层，参数判断中要修改参数的判定，我试试看。

var clinics = \_remoteClinicService.GetRemoteClinicList(search);

result.data = new ApiPageListModel<RemoteClinicModel>

{

DataList = clinics,

PageInfo = new PageInfo

{

HasNextPage = clinics.PageIndex + 1 < clinics.TotalPages,

HasPreviousPage = clinics.PageIndex > 0,

PageIndex = clinics.PageIndex,

PageSize = clinics.PageSize,

TotalCount = clinics.TotalCount,

TotalPages = clinics.TotalPages

}

};

}

catch (Exception ex)

{

SystemLogHelper.LogInsert(new AddLogParam { LogLevelId = (int)LogLevel.Error, ShortMessage = "协同-[远程门诊]-获取门诊专家远程门诊任务列表", FullMessage = $"传入参数：{JsonConvert.SerializeObject(search)}，详细错误：{ex}" });

result.code = ResultStatusCode.DEFAULT\_ERROR;

result.msg = "获取门诊信息异常";

}

return result;

}

DAL层的按条件搜索：

/// <summary>

/// 获取门诊申请列表

/// </summary>

/// <param name="search">搜索条件</param>

/// <returns></returns>

public PagedList<RemoteClinicModel> GetRemoteClinicList(RemoteClinicSearhModel search)

{

var query = from s in \_remoteClinicRepository.Table

select new RemoteClinicModel()

{

Age = s.Age,

AgeUnit = s.AgeUnit,

IsMarried = s.IsMarried,

CancelReason = s.CancelReason,

ConsultEndDate = s.ConsultEndDate,

ConsultStartDate = s.ConsultStartDate,

ConsultSubjectId = s.ConsultSubjectId,

ConsultSubjectName = s.ConsultSubjectName,

CustomerGuid = s.CustomerGuid,

CustomerName = s.CustomerName,

DiagnosisTime = s.DiagnosisTime,

ExpertCustomerGuid = s.ExpertCustomerGuid,

ExpertCustomerName = s.ExpertCustomerName,

IsEvaluate = s.IsEvaluate,

//IsStop = s.IsStop,

IsUrgency = s.IsUrgency,

PatientName = s.PatientName,

PatientSex = s.PatientSex,

RequestDate = s.RequestDate,

RequestHospitalName = s.RequestHospitalName,

RequestId = s.RequestId,

SheduleKind = s.SheduleKind,

ServiceCenterName = s.ServiceCenterName,

ServiceCenterId = s.ServiceCenterId,

RequestOfficeName = s.RequestOfficeName,

ServiceState = s.ServiceState,

ServiceId = s.ServiceId,

SheduleBeginTime = s.SheduleBeginTime,

SheduleEndTime = s.SheduleEndTime,

Purpose = s.Purpose,

PatientPhone = s.PatientPhone,

Order = s.Order,

FirstDiagnosis = s.FirstDiagnosis

};

#region 更多搜索条件

if (search.ServiceId.HasValue)

{

query = query.Where(s => s.ServiceId == search.ServiceId);

}

if (search.CustomerGuid.HasValue)

{

这地方，感觉是和我之间写的接口那样，把当前用户是接收方或者请求方的情况都考虑进去了，不过它这里只有一种情况会最后存在。我那边是两个都得有。

其实在想，如果涉及请求方和接收方的对调，可以参考飞行棋的取反去操作，只有就对应上了。

playNumber = 1 – playNumber

比方我创建字典Dictionary<int,Guid>,其中dic[1]是requestGuid，而dic[1]是ResponseGuid，只有就可以通过playNumber = 1 – playNumber，去实现用户切换了。

/// 是否专家任务列表 True=是，False=否

if (search.IsExpert == true)

query = query.Where(s => s.ExpertCustomerGuid == search.CustomerGuid);

else

query = query.Where(s => s.CustomerGuid == search.CustomerGuid);

}

//if (search.IsCall.HasValue)

//{

// query = query.Where(s => s.IsCall == search.IsCall);

//}

//if (search.IsStop.HasValue)

//{

// query = query.Where(s => s.IsStop == search.IsStop);

//}

if (search.SheduleKind != null && search.SheduleKind.Count != 0)

{

query = query.Where(s => search.SheduleKind.Contains(s.SheduleKind.Value));

}

if (search.BeginSheduleTime.HasValue && search.EndSheduleTime.HasValue)

{

query = query.Where(s => s.SheduleBeginTime >= search.BeginSheduleTime && s.SheduleEndTime <= search.EndSheduleTime);

}

if (search.BeginRequestTime.HasValue && search.EndRequestTime.HasValue)

{

query = query.Where(s => s.RequestDate >= search.BeginRequestTime && s.RequestDate <= search.EndRequestTime);

}

if (search.ServiceState != null && search.ServiceState.Count > 0)

{

query = query.Where(s => search.ServiceState.Contains(s.ServiceState.Value));

}

if (search.ServiceCenterId != null && search.ServiceCenterId.Count > 0)

{

query = query.Where(s => search.ServiceCenterId.Contains(s.ServiceCenterId.Value));

}

if (!string.IsNullOrWhiteSpace(search.PatientName))

{

query = query.Where(s => s.PatientName.Contains(search.PatientName));

}

#endregion

if (null == search.SortInfo || (string.IsNullOrWhiteSpace(search.SortInfo.SortField) || string.IsNullOrWhiteSpace(search.SortInfo.SortType)))

{

query = query.OrderBy(s => s.SheduleKind).ThenBy(s => s.SheduleBeginTime);

}

else

{

query = LinqHelper.DataSorting(query, search.SortInfo.SortField, search.SortInfo.SortType);

}

PagedList<RemoteClinicModel> page = new PagedList<RemoteClinicModel>(query, search.PageIndex, search.PageSize);

return page;

}

这里登陆用户对应的是接收方，而如果要查当前医生的申请，那只要IsExpert字段不是true就可以了。

如下：

/// 获取申请医生远程门诊申请列表

/// </summary>

/// <param name="search">搜索条件</param>

/// <returns></returns>

public BaseApiModel<ApiPageListModel<RemoteClinicModel>> GetDoctorRemoteClinicList(RemoteClinicSearhModel search)

{

……

search.IsExpert = null;

这里设置为null了

if (search.BeginSheduleTime.HasValue && search.EndSheduleTime.HasValue)

{

search.EndSheduleTime = search.EndSheduleTime.Value.AddDays(1);

}

if (search.EndRequestTime.HasValue && search.EndRequestTime.Value != DateTime.MinValue)

{

search.EndRequestTime = search.EndRequestTime.Value.AddDays(1);

}

var clinics = \_remoteClinicService.GetRemoteClinicList(search);

Dal层：

if (search.CustomerGuid.HasValue)

{

因为上面设置了search.IsExpert==null，所以这里就是走下面这条路，获取了当前用户为请求方，也就是对应我做的朋友圈里面的RequestCustomerGuid。

if (search.IsExpert == true)

query = query.Where(s => s.ExpertCustomerGuid == search.CustomerGuid);

else

query = query.Where(s => s.CustomerGuid == search.CustomerGuid);

}

（2）



<http://localhost:9500/Doctor/Clinic/GetStatistics>

（3）



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

入参，无

{}

响应的是一堆的站内信消息。

……

……

{

"Kind": null,

"ServiceId": "a0076003-1625-475c-9fe9-ba98d54d96fe",

"Content": "专家[洪启晨]在[2018/4/24 18:40:45]对您提交的 心电诊断 报告[患者姓名：测试，性别：男]进行了[领取]操作，请知悉",

"PushState": true

},

{

"Kind": null,

"ServiceId": "a0076003-1625-475c-9fe9-ba98d54d96fe",

"Content": "专家洪启晨在2018/4/24 16:52:50对您提交的心电诊断 报告[患者姓名：测试，性别：男]进行了领取操作，请知悉",

"PushState": true

}

]

},

"code": 0,

"msg": "获取站内消息成功"

}

ca处理：

#region 站内消息管理

/// <summary>

/// 站内消息管理

/// </summary>

/// <returns></returns>

[HttpPost]

public async Task<ActionResult> GetUserMessageList()

{

var user = CustomerIdentity;

dynamic apiResult = null;

走的是异步操作

await Task.Factory.StartNew(() =>

{

apiResult =

\_commonClient.Invoke(m => m.GetUserMessageList(user.CustomerGuid));

});

return Json(apiResult);

}

#endregion

BLL层：

去redis中获取数据

/// <summary>

/// 获取站内消息列表

/// </summary>

/// <param name="customerGuid"></param>

/// <returns></returns>

public BaseApiModel<ApiListModel<MessageEntity>> GetUserMessageList(Guid? customerGuid)

{

return RadisHelper.Get(customerGuid);

}

另一个BLL层：

var responseData = StackExchangeRedisHelper.Get($"{RadisName}{customerGuid}");

（4）



<http://localhost:9500/Doctor/Collection/GetCollectionCount>

入参：无。

{}

后台响应：

{

"data": 9,

"code": 0,

"msg": "成功"

}

反馈在页面上



CA处理：

增加远程医疗协同的字段

private readonly int collectionSystem = (int)CollectionSystemEnum.远程医疗协同;

/// <summary>

/// 查询收藏总数

/// </summary>

/// <returns></returns>

[HttpPost]

public async Task<JsonResult> GetCollectionCount()

{

dynamic ret = null;

var user = CustomerIdentity;

又是异步任务，增加服务器吞吐量？

await Task.Factory.StartNew(() =>

{

ret = \_crmClient.Invoke(s => s.GetCollectionCount(user.CustomerGuid, collectionSystem));

});

return Json(ret);

}

DAL层，查询远程系统的收藏量：

/// <summary>

/// 查询用户收藏总数

/// </summary>

/// <param name="customerGuid">用户唯一号</param>

/// <param name="CollectionSystem">系统来源</param>

/// <returns></returns>

public int GetCollectionCount(Guid customerGuid, int? collectionSystem)

{

var query = from s in \_myCollectionService.Table

where s.CustomerGuid == customerGuid

select s;

if (collectionSystem != null)

{

query = query.Where(s => s.CollectionSystem == collectionSystem);

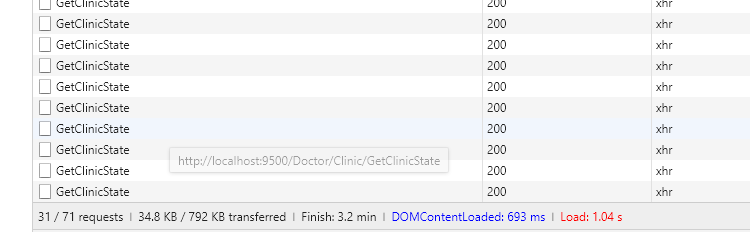
}

return query.Count();

}

(5)

一直在轮询这个请求，应该是前端设置了定时的ajax访问任务：



<http://localhost:9500/Doctor/Clinic/GetClinicState>

入参：

{"ServiceIds":["9f07ffed-6b80-45c9-8c5c-25e3b41cbfaa"],"Type":2}

ca处理：

/// <summary>

/// 获取门诊状态 -新

/// </summary>

/// <param name="form">参数</param>

/// <returns></returns>

[HttpPost]

public ActionResult GetClinicState(ClinicRefreshSearchModel form)

{

彪哥是在这里做参数校验的，理论上确实应该校验，如果不符合，直接返回错误数据，就不用进一步地去wcf中调方法了，但如果这样，这里就要new一个baseApi这种返回结果集合了，但是在下一步的bll层又要重新做身份校验，对应的参数是否存在，而如果身份校验放在ui层，也就是控制器中，如果是面向服务开发，这边又不可能直接拿到数据，所以综合起来，还是全部放到bll层去处理参数和身份。这样baseApi这种统一的返回参数，也只需要new一次。

#region 参数检查

if (form.Type == 1 || form.Type == 2)

form.CustomerGuid = CustomerIdentity.CustomerGuid;

#endregion

var result = \_clinicClient.Invoke(s => s.GetClinicState(form));

return Json(result);

}

DAL层：

彪哥这里的处理和我之前的，不传GroupId，返回所有结果的想法很类似，就是查出所有好友的数据。

public List<RemoteClinic> GetClinicStateList(Guid? customerGuid, DateTime? beginTime, DateTime? endTime, List<Guid> serviceIds, bool isExpert = true)

{

var query = from s in \_remoteClinicRepository.Table

select new

{

s.CustomerGuid,

s.ExpertCustomerGuid,

s.SheduleBeginTime,

s.ServiceId,

s.ServiceState

};

这种所有的根据参数来筛选结果，我在想，是否可以传一个lambda条件

if (customerGuid.HasValue)

if (isExpert)

{

query = query.Where(s => s.ExpertCustomerGuid == customerGuid);

}

else

{

query = query.Where(s => s.CustomerGuid == customerGuid);

}

if (serviceIds != null && serviceIds.Count > 0)

{

query = query.Where(s => serviceIds.Contains(s.ServiceId.Value));

}

if (beginTime.HasValue && endTime.HasValue)

{

query = query.Where(s => s.SheduleBeginTime >= beginTime && s.SheduleBeginTime <= endTime);

}

var list = query.ToList();

return list.Select(s => new RemoteClinic()

{

ServiceState = s.ServiceState,

ServiceId = s.ServiceId

}).ToList();

}

在BLL层的处理：

DateTime? beginTime, endTime;

if (model.Type == 1 || model.Type == 2)

{

beginTime = DateTime.Parse(DateTime.Now.ToString("yyyy-MM-dd"));

endTime = beginTime.Value.AddDays(1);

}

else

{

beginTime = null;

endTime = null;

}

先获取所有的远程门诊集合列表，而且这边限制了日期，也就是当天的所有门诊集合

我渐渐发现的问题，DAL层返回的全是ToList之后的内存集合，而不是Iqueryalbe的延时集合，增加性能，这里是一个优化点。

var clinics = \_remoteClinicService.GetClinicStateList(model.CustomerGuid, beginTime, endTime, model.ServiceIds, model.Type == 2);

if (clinics.Count > 0)

{

这里的s.ServiceId.GetValueOrDefault()，测试过了，如果是null，则返回的是000-000那种，Guid.Empty默认值。

这里的门诊列表设置未6，申请方推迟，所以轮询的是为了获取所有的推迟的唯一门诊号guid集合

ServiceState: -1取消门诊、0已预约、1等待调度（已付款）、2已调度、3已接受、4已签到、5 呼叫中、6 申请方推迟、10咨询中、15 已挂起、20专家已回复、30已评价

var serviceIds = clinics.Where(s => s.ServiceState == 6).Select(s => s.ServiceId.GetValueOrDefault()).ToList();

List<RemoteClinicProcess> processList = null;

如果存在有要推迟的用户

if (serviceIds.Count > 0)

{

获取门诊过程列表。这里限定了event字段的值为6

也就是所有推迟的门诊集合

列举：Event: : -1取消门诊、0已预约、1等待调度、2已调度、3已接受、4已签到、5 呼叫中、6 申请方推迟、10咨询中、15 已挂起、20专家已回复、30已评价

processList = \_processService.GetList(serviceIds, new List<int>() {

6

});

}

最终返回的是刷新后的门诊状态结果数据集。

数据的整合都是放在BLL层，这是我要学习的地方。

result.data = new ApiListModel<ClinicRefreshResultModel>

{

DataList = clinics.Select(s => new ClinicRefreshResultModel()

{

ServiceId = s.ServiceId.GetValueOrDefault(),

ServiceState = s.ServiceState.GetValueOrDefault(),

Process = processList?.Where(p => p.ServiceId == s.ServiceId).OrderByDescending(p => p.Id).FirstOrDefault()?.MapTo<RemoteClinicProcessModel>()

}).ToList()

};

}

返回结果一直是null的数据

{

"data": null,

"code": 0,

"msg": "成功"

}

2.

2.1点击查看门诊信息



页面弹窗，div变block：



<http://192.168.1.202:9500/Doctor/Clinic/GetRemoteClinic>

入参：

{"serviceId":"9f07ffed-6b80-45c9-8c5c-25e3b41cbfaa"}

应该是门诊唯一号

返回消息一大坨，反正不管前台要不要，只管往前台发就是了。

{

"data": {

"Id": 361,

"ServiceId": "9f07ffed-6b80-45c9-8c5c-25e3b41cbfaa",

"RequestId": null,

"SheduleKind": 1,

"PatientName": "10-26-1",

"PatientSex": "男",

"PatientBirthday": null,

"Age": 1,

"AgeUnit": "岁",

"IsMarried": 0,

"Nation": null,

"Profession": null,

"OriginalMedNo": null,

"SystemMedNo": null,

"IdNumber": null,

"HealthCardNumber": null,

"HealthCardType": null,

"PatientDesc": null,

"MedSummary": null,

"Purpose": "1",

"RequestHospitalCode": "15112",

"RequestHospitalName": "上海交通大学医学院附属瑞金医院",

"RequestOfficeId": null,

"RequestOfficeName": null,

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

"CustomerName": "王专家",

"RequestPhone": "13867115748",

"PatientPhone": "15271007788",

"RequestDate": "2018-10-26 11:31:54",

"ConsultHospitalCode": null,

"ConsultHospitalName": null,

"ConsultSubjectId": 59,

"ConsultSubjectName": "cs",

"KeyWord": null,

"Order": 231,

"SheduleBeginTime": "2018-10-26 11:50:00",

"SheduleEndTime": "2018-10-26 11:51:00",

"ConsultStartDate": "2018-10-26 11:36:33",

"ConsultEndDate": null,

"CancelReason": null,

"ServiceState": 10,

"IsScript": null,

"IsEvaluate": null,

"IsUrgency": null,

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

"ExpertCustomerName": "洪启晨",

"Diagnosis": null,

"DiagnosisMemo": null,

"DiagnosisTime": null,

"ReviseTime": null,

"ServiceCenterId": 35,

"ServiceCenterName": "平湖测试服务中心",

"AdviceContent": null,

"AdviceDate": null,

"Grade": null,

"IsCall": null,

"IsStop": null,

"FirstDiagnosis": null,

"ExpertPhone": null,

"DocSetJson": null

},

"code": 0,

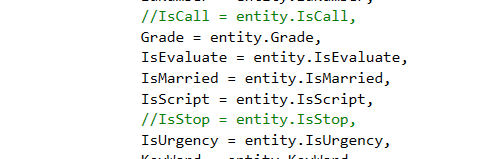
"msg": "获取门诊信息成功"

}

有个DTO的转换

result.data = ClincModelExtend.RemoteClinicToModel(clinic);

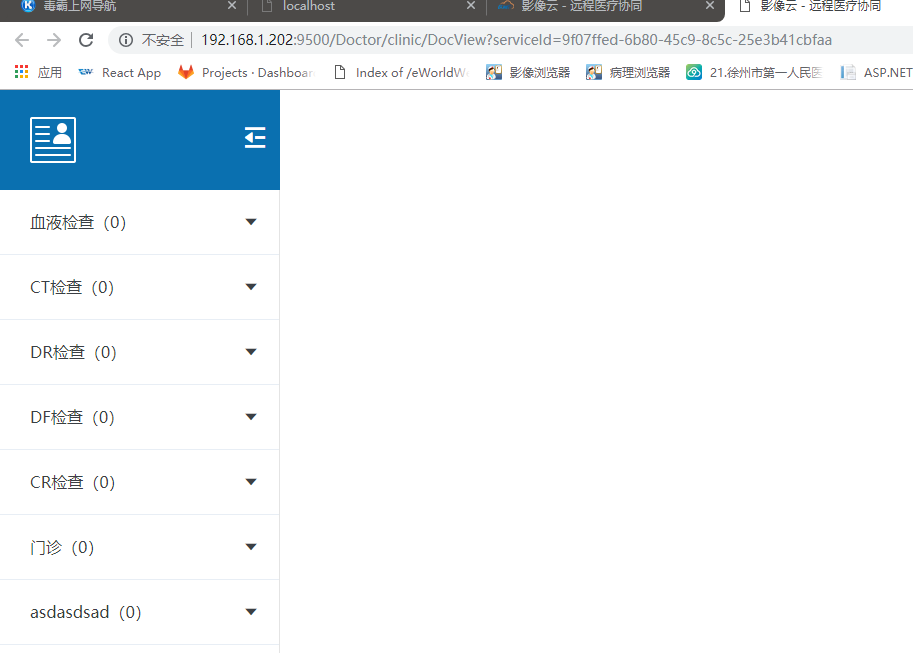
其实也就少了两个参数



（2）当继续点击资料浏览



跳到这个界面



<http://192.168.1.202:9500/Doctor/clinic/DocView?serviceId=9f07ffed-6b80-45c9-8c5c-25e3b41cbfaa>

CA：

/// <summary>

/// 资料浏览

/// </summary>

/// <returns></returns>

public ActionResult DocView()

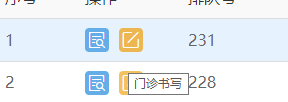
{

return View();

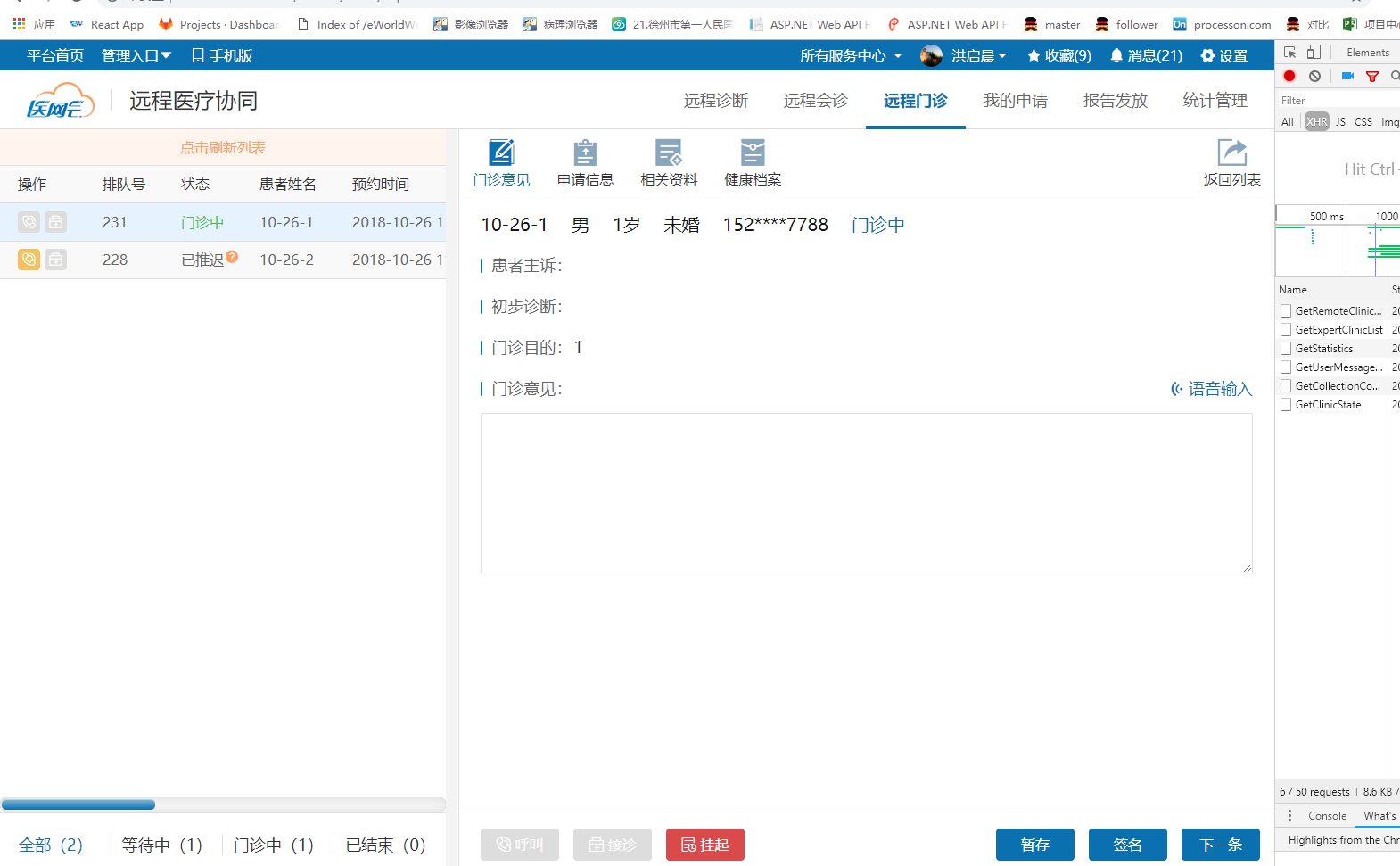
}

3.

当点击了门诊书写



跳入这个界面：



地址：

<http://192.168.1.202:9500/Doctor/Clinic/ExpertWrite?serviceId=9f07ffed-6b80-45c9-8c5c-25e3b41cbfaa>

对应：

/// <summary>

/// 专家书写界面

/// </summary>

/// <returns></returns>

public ActionResult ExpertWrite()

{

return View();

}

（1）



<http://192.168.1.202:9500/Doctor/Clinic/GetRemoteClinicDetail>

入参，{"serviceId":"9f07ffed-6b80-45c9-8c5c-25e3b41cbfaa"}

Response返回的信息：

{

"data": {

"Clinic": {

"Id": 361,

"ServiceId": "9f07ffed-6b80-45c9-8c5c-25e3b41cbfaa",

"RequestId": null,

"SheduleKind": 1,

"PatientName": "10-26-1",

"PatientSex": "男",

"PatientBirthday": null,

"Age": 1,

"AgeUnit": "岁",

"IsMarried": 0,

"Nation": null,

"Profession": null,

"OriginalMedNo": null,

"SystemMedNo": null,

"IdNumber": null,

"HealthCardNumber": null,

"HealthCardType": null,

"PatientDesc": null,

"MedSummary": null,

"Purpose": "1",

"RequestHospitalCode": "15112",

"RequestHospitalName": "上海交通大学医学院附属瑞金医院",

"RequestOfficeId": null,

"RequestOfficeName": null,

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

"CustomerName": "王专家",

"RequestPhone": "13867115748",

"PatientPhone": "15271007788",

"RequestDate": "2018-10-26 11:31:54",

"ConsultHospitalCode": null,

"ConsultHospitalName": null,

"ConsultSubjectId": 59,

"ConsultSubjectName": "cs",

"KeyWord": null,

"Order": 231,

"SheduleBeginTime": "2018-10-26 11:50:00",

"SheduleEndTime": "2018-10-26 11:51:00",

"ConsultStartDate": "2018-10-26 11:36:33",

"ConsultEndDate": null,

"CancelReason": null,

"ServiceState": 10,

"IsScript": null,

"IsEvaluate": null,

"IsUrgency": null,

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

"ExpertCustomerName": "洪启晨",

"Diagnosis": null,

"DiagnosisMemo": null,

"DiagnosisTime": null,

"ReviseTime": null,

"ServiceCenterId": 35,

"ServiceCenterName": "平湖测试服务中心",

"AdviceContent": null,

"AdviceDate": null,

"Grade": null,

"IsCall": null,

"IsStop": null,

"FirstDiagnosis": null,

"ExpertPhone": null,

"DocSetJson": null

},

"JBMC": [],

"SignValue": null,

"VideoInfo": {

"VideoType": "3",

"Url": [

"http://localhost:9500"

]

},

"VideoUser": {

"Current": {

"UserId": null,

"Password": null,

"Name": "洪启晨",

"Organization": "平湖市第一人民医院",

"DepartMent": "医学影像科",

"Title": "主任医师"

},

"Other": {

"UserId": null,

"Password": null,

"Name": "王专家",

"Organization": "上海交通大学医学院附属瑞金医院",

"DepartMent": "传染科",

"Title": "主任医师"

}

}

},

"code": 0,

"msg": "获取门诊信息成功"

}

CA处理：

获取门诊详情：

public ActionResult GetRemoteClinicDetail(Guid serviceId)

{

#region 参数检查

var result = new BaseApiModel<RemoteClinicDetailModel>();

if (serviceId.Equals(Guid.Empty))

{

result.code = ResultStatusCode.LACK\_PARA;

result.msg = "门诊唯一号不能为空";

return Json(result);

}

#endregion

result = \_clinicClient.Invoke(s => s.GetRemoteClinicDetail(serviceId));

视频会议信息未展开，没有看下去

if (result.data?.Clinic != null)

{

#region 视频会议信息

result.data.VideoInfo = new ClinicVideoInfo();

result.data.VideoInfo.VideoType = ConfigAppCommon.MeetSoftType == "1" ? "2" : "0";

if (ConfigAppCommon.EnabledTraining.ToLower().Equals("false") && result.data.VideoInfo.VideoType == "0")

{

result.data.VideoInfo.VideoType = "1";

}

if (\_medicineSettings.IsJuphoonVideo)

{

result.data.VideoInfo.VideoType = "3";

}

if (result.data.VideoInfo.VideoType.Equals("3"))

{

#region 菊风视频

if (result.data.Clinic.ServiceState >= 0 && result.data.Clinic.ServiceState < 20)

{

result.data.VideoInfo.Url = new[] {

\_medicineSettings.JuphoonServerUrl

};

var otherCustomerGuid = CustomerIdentity.CustomerGuid == result.data.Clinic.CustomerGuid ?

result.data.Clinic.ExpertCustomerGuid : result.data.Clinic.CustomerGuid;

var currentCustomer = \_crmClient.Invoke(s => s.GetDoctorMemberInformation(CustomerIdentity.CustomerGuid));

var otherCustomer = \_crmClient.Invoke(s => s.GetDoctorMemberInformation(otherCustomerGuid));

result.data.VideoUser = new ClinicVideoUser()

{

Current = new ClinicVideoUserInfo()

{

Title = currentCustomer.data.Title,

Organization = currentCustomer.data.HospitalName,

DepartMent = currentCustomer.data.Office,

Name = currentCustomer.data.CustomerName,

UserId = currentCustomer.data.UserId

},

Other = new ClinicVideoUserInfo()

{

Title = otherCustomer.data.Title,

Organization = otherCustomer.data.HospitalName,

DepartMent = otherCustomer.data.Office,

Name = otherCustomer.data.CustomerName,

UserId = otherCustomer.data.UserId

}

};

}

#endregion

}

else

{

#region 展示互动/软铸视频会议

var meeting = \_commonClient.Invoke(s => s.GetMeetingByGlobalIdAndType(result.data.Clinic.ServiceId.ToString(), (int)RelationType.门诊));

if (meeting != null)

{

if (result.data.Clinic.ServiceState >= 0 && result.data.Clinic.ServiceState < 20)

{

var joinUrl = BaseCommon.GetMeetingJoinUrl(new MeetJoinRequestModel()

{

CustomerGuid = CustomerIdentity.CustomerGuid,

CustomerName = CustomerIdentity.CustomerName,

IsSpeaker = CustomerIdentity.CustomerGuid.Equals(result.data.Clinic.ExpertCustomerGuid),

OrganizationName = CustomerIdentity.CustomerGuid.Equals(result.data.Clinic.ExpertCustomerGuid) ? result.data.Clinic.ServiceCenterName : CustomerIdentity.HospitalName,

OrganizerToken = meeting.OrganizerToken,

PanelistJoinUrl = meeting.PanelistJoinUrl,

PanelistToken = meeting.PanelistToken,

SubjectName = null,

MId = meeting.Mid,

});

result.data.VideoInfo.Url = new[] {

joinUrl

};

}

else if (result.data.Clinic.ServiceState >= 20)

{

var \_records = BaseCommon.GetMeetingRecordList(new MeetVideoRecordRequestModel()

{

AttendeeToken = meeting.AttendeeToken,

CustomerName = CustomerIdentity.CustomerName,

MId = meeting.Mid

});

result.data.VideoInfo.Url = \_records?.Select(s => s.Url).ToArray();

}

}

#endregion

}

#endregion

}

return Json(result);

}

由于门诊申请详情数据由多部分组成，所以分开赋值。

/// <summary>

/// 门诊申请详情数据模型

/// </summary>

public class RemoteClinicDetailModel

{

/// <summary>

/// 门诊申请信息

/// </summary>

public RemoteClinicModel Clinic { get; set; }

/// <summary>

/// 门诊疾病编码列表

/// </summary>

public IList<ClinicJBMCModel> JBMC { get; set; }

/// <summary>

/// CA签名字符串

/// </summary>

public string SignValue { get; set; }

/// <summary>

/// 视频会议信息

/// </summary>

public ClinicVideoInfo VideoInfo { get; set; }

/// <summary>

/// 视频用户，仅视频类型为菊风时有效

/// </summary>

public ClinicVideoUser VideoUser { get; set; }

}

BLL层中分别去获取：

/// <summary>

/// 获取门诊详情信息

/// </summary>

/// <param name="serviceId">门诊唯一号</param>

/// <returns></returns>

public BaseApiModel<RemoteClinicDetailModel> GetRemoteClinicDetail(Guid serviceId)

{

var result = new BaseApiModel<RemoteClinicDetailModel> { code = ResultStatusCode.OK, msg = "成功", data = new RemoteClinicDetailModel() };

try

{

var clinic = \_remoteClinicService.GetRemoteClinic(serviceId);

if (null == clinic)

{

result.code = ResultStatusCode.DATA\_DONOT\_EXIST;

result.msg = "门诊信息不存在";

return result;

}

将远程门诊的信息赋值给Clinic属性

result.data.Clinic = ClincModelExtend.RemoteClinicToModel(clinic);

if (clinic.DigitalSignatureId.HasValue)

{

获取门诊签名信息

var sign = \_remoteClinicService.GetSignValue(clinic.DigitalSignatureId.Value);

result.data.SignValue = sign?.SignValue;

}

签名表中的数据。

通过门诊唯一号获取门诊疾病编码列表，再给赋值给data中的JBMC属性。

result.data.JBMC = \_remoteClinicService.GetClinicJBMCList(serviceId).Select(s => new ClinicJBMCModel()

{

DM = s.DM,

ICD = s.ICD,

MC = s.MC,

Memo = s.Memo,

ServiceId = s.ServiceId

}).ToList();

result.msg = "获取门诊信息成功";

return result;

}

catch (Exception ex)

{

SystemLogHelper.LogInsert(new AddLogParam { LogLevelId = (int)LogLevel.Error, ShortMessage = "协同-[远程门诊]-获取门诊详情信息", FullMessage = $"传入参数：serviceId={serviceId}，详细错误：{ex}" });

result.code = ResultStatusCode.DEFAULT\_ERROR;

result.msg = "获取门诊信息异常";

}

return result;

}

扩展：

a.获取数字签名，主键Id就等于远程门诊中的数字Id，应该是这边创建了，然后写入远程门诊表中，类似我之前的把提问表id写入到提问对象中，放在事务中去操作。

这样就可以通过远程门诊表中的id，再去数字签名表中查找详细数据。

/// <summary>

/// 获取门诊签名信息

/// </summary>

/// <param name="signatureId">签名Id</param>

/// <returns></returns>

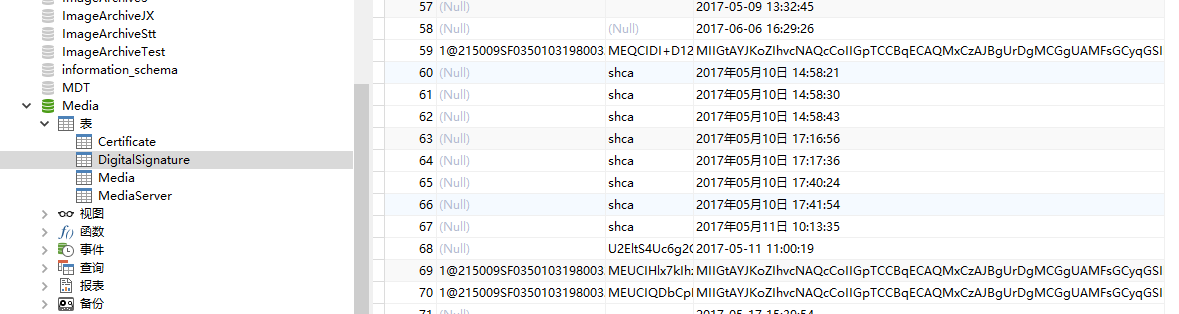
public DigitalSignature GetSignValue(int signatureId)

{

return (from s in \_digitalSignatureRepository.Table where s.Id == signatureId select s).FirstOrDefault();

}

表中的数据：



（2）



获取门诊统计信息

<http://192.168.1.202:9500/Doctor/Clinic/GetStatistics>

入参：

{"BeginRequestTime":null,"EndRequestTime":null,"BeginSheduleTime":"2018-10-08","EndSheduleTime":"2018-11-08","IsExpert":true,"ServiceCenterId":null,"ServiceState":[0,1,2,3,4,5,6,10,15,20,30],"SheduleKind":null}

返回信息：

{

"data": {

"DataList": [

{

"ServiceState": 10,

"Count": 2

}

]

},

"code": 0,

"msg": "成功"

}

DAL层：

/// <summary>

/// 查询门诊列表统计数据(分组依据 状态)

/// </summary>

/// <param name="search"></param>

/// <returns></returns>

public List<ClinicStatisticsModel> GetStatistics(RemoteClinicSearhModel search)

{

var query = from s in \_remoteClinicRepository.Table

select s;

#region 更多搜索条件

if (search.ServiceId.HasValue)

{

query = query.Where(s => s.ServiceId == search.ServiceId);

}

if (search.CustomerGuid.HasValue)

{

if (search.IsExpert == true)

query = query.Where(s => s.ExpertCustomerGuid == search.CustomerGuid);

else

query = query.Where(s => s.CustomerGuid == search.CustomerGuid);

}

if (search.SheduleKind != null && search.SheduleKind.Count != 0)

{

query = query.Where(s => search.SheduleKind.Contains(s.SheduleKind.Value));

}

if (search.BeginSheduleTime.HasValue && search.EndSheduleTime.HasValue)

{

query = query.Where(s => s.SheduleBeginTime >= search.BeginSheduleTime && s.SheduleBeginTime <= search.EndSheduleTime);

}

if (search.BeginRequestTime.HasValue && search.EndRequestTime.HasValue)

{

query = query.Where(s => s.RequestDate >= search.BeginRequestTime && s.RequestDate <= search.EndRequestTime);

}

if (search.ServiceState != null && search.ServiceState.Count > 0)

{

query = query.Where(s => search.ServiceState.Contains(s.ServiceState.Value));

}

if (search.ServiceCenterId != null && search.ServiceCenterId.Count > 0)

{

query = query.Where(s => search.ServiceCenterId.Contains(s.ServiceCenterId.Value));

}

if (!string.IsNullOrWhiteSpace(search.PatientName))

{

query = query.Where(s => s.PatientName.Contains(search.PatientName));

}

#endregion

根据所有的search条件获取门诊列表之后，根据门诊状态进行分组，输出所有的分组值，以及对应的所有列表信息条数

ServiceState: -1取消门诊、0已预约、1等待调度（已付款）、2已调度、3已接受、4已签到、5 呼叫中、6 申请方推迟、10咨询中、15 已挂起、20专家已回复、30已评价

var \_group = query.GroupBy(s => s.ServiceState).Select(s => new ClinicStatisticsModel()

{

ServiceState = s.Key,

Count = s.Count()

}).ToList();

return \_group;

}

分组信息体现在这：



（3）



仍旧是首页的获取专家列表

体现在左侧：



（4）

（5）

4.

5.

6.