**急诊临床信息系统**

概要设计说明书

|  |  |  |
| --- | --- | --- |
| 文件状态：  [√] 草稿  [] 正式发布  [] 正在修改 | 文件标识： | Szyjian-JZ-Doc-001 |
| 当前版本： | 2.0 |
| 作 者： | 王东 |
| 完成日期： | 2021/08/29 |
| 审 批 人： | 苏适 |

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| 2.0 | 王东 |  | 2021/8/26---2021/08/29 | 根据需求规格说明书确定功能模块加之科室功能需求的归纳，总结与提炼 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. 引言 4](#_Toc8642)

[1.1. 编写目的 4](#_Toc18745)

[1.2. 背景 4](#_Toc14204)

[1.3. 基线 4](#_Toc10862)

[1.4. 特殊名词定义 4](#_Toc5464)

[1.5. 参考资料 5](#_Toc21127)

[2. 总体设计 5](#_Toc1440)

[2.1. 概述 5](#_Toc16151)

[2.2. 系统环境描述 6](#_Toc8799)

[2.2.1. 运行环境 6](#_Toc29104)

[2.2.2. 开发环境 7](#_Toc24492)

[2.3. 系统总体结构设计 9](#_Toc27549)

[2.3.1. 系统业务层次图 11](#_Toc18703)

[2.3.2. 业务流程图 11](#_Toc25628)

[2.3.3. 系统架构图 12](#_Toc4922)

[2.3.4. 软件架构图 17](#_Toc30145)

[2.3.5. 逻辑架构图 17](#_Toc27765)

[3. 外部接口 19](#_Toc9137)

[3.1.1. 硬件设备 19](#_Toc2405)

[3.1.2. 软件系统 19](#_Toc24006)

[4. 模块设计 20](#_Toc6425)

[4.1. 患者微服务 20](#_Toc8939)

[4.2. 叫号微服务 21](#_Toc25224)

[4.3. 处方微服务 22](#_Toc6950)

[4.4. 检查微服务 24](#_Toc7901)

[4.5. 检验微服务 25](#_Toc3887)

[4.6. 诊疗微服务 27](#_Toc32583)

[4.7. 病历微服务（同院前病历微服务共用-基于都昌的实现） 29](#_Toc4485)

[4.8. 全景视图微服务 31](#_Toc552)

[5. 数据结构 31](#_Toc3420)

[6. 容灾设计 32](#_Toc6868)

[6.1. 出错情况 32](#_Toc20963)

[6.2. 异常处理 32](#_Toc2116)

[6.3. 补救措施 34](#_Toc9380)

[6.4. 容灾备份 34](#_Toc23147)

[6.5. 系统维护设计 34](#_Toc31765)

[7. 监控设计 35](#_Toc25349)

[8. 用户界面设计 36](#_Toc19354)

[8.1. 急诊叫号 36](#_Toc30239)

[8.1.1. 护士端 36](#_Toc21622)

[8.1.2. 医生端 37](#_Toc16300)

[8.1.3. 患者端 37](#_Toc19339)

[8.2. 急诊诊疗 38](#_Toc24281)

[8.2.1. 诊区 38](#_Toc25050)

[8.2.2. 抢救区、留观区 38](#_Toc15626)

[8.2.3. 历史就诊 39](#_Toc28178)

[8.3. 医生站 39](#_Toc1438)

[8.3.1. 患者主页 39](#_Toc10179)

[8.3.2. 医嘱开立 39](#_Toc3305)

[8.3.3. 电子病历 41](#_Toc10471)

[8.3.4. 手术申请 42](#_Toc10256)

[8.3.5. 会诊管理 42](#_Toc20696)

[8.3.6. 全景视图 43](#_Toc17575)

[8.3.7. 打印中心 43](#_Toc15352)

[8.3.8. 患者360 44](#_Toc17976)

[9. 系统出错处理设计 44](#_Toc23954)

[9.1. 出错信息 44](#_Toc8484)

[9.2. 补救措施 44](#_Toc30139)

[9.3. 系统维护设计 45](#_Toc9602)

# 引言

## 编写目的

本文档用于确定ECIS系统的基本框架，其基本目标是能够针对软件需求分析中提出的一系列软件问题，概要的回答问题如何解决。例如，软件系统将采用什么样的体系构架、需要创建哪些功能模块、 模块之间的关系如何、数据结构如何？软件系统需要什么样的网络环境提供支持、需要采用什 么类型的后台数据库等。本文档适用于开发人员、测试人员、文档撰写人员、营销人员及项目经理。

## 背景

此软件系统的名称：ECIS

此软件系统的需求方：医院

此软件系统的开发方：深圳市尚哲医健科技有限责任公司

此软件系统的用户：医生、护士、科室主任、系统管理员

## 基线

急诊临床信息系统需求分析说明书 v1.0

## 特殊名词定义

|  |  |
| --- | --- |
| HIS | 医院管理信息系统 |
| ECIS | 急诊临床信息系统 |
| LIS | 检验信息系统 |
| EMR | 电子病历 |
| PACS | 医学影像存档与传输系统 |
| ECIS | 急诊临床信息系统 |
| 120 | 120急救系统 |

## 参考资料

* 国家与行业标准

1. 《急诊科建设与管理指南（试行）》（卫医政发〔2009〕50号)
2. 《医院急诊科规范化流程》（WS/T 390--2012）
3. 《电子病历基本规范(试行)》（卫医政发〔2010〕24号）
4. 《电子病历基本架构与数据标准(2018年最新版)》

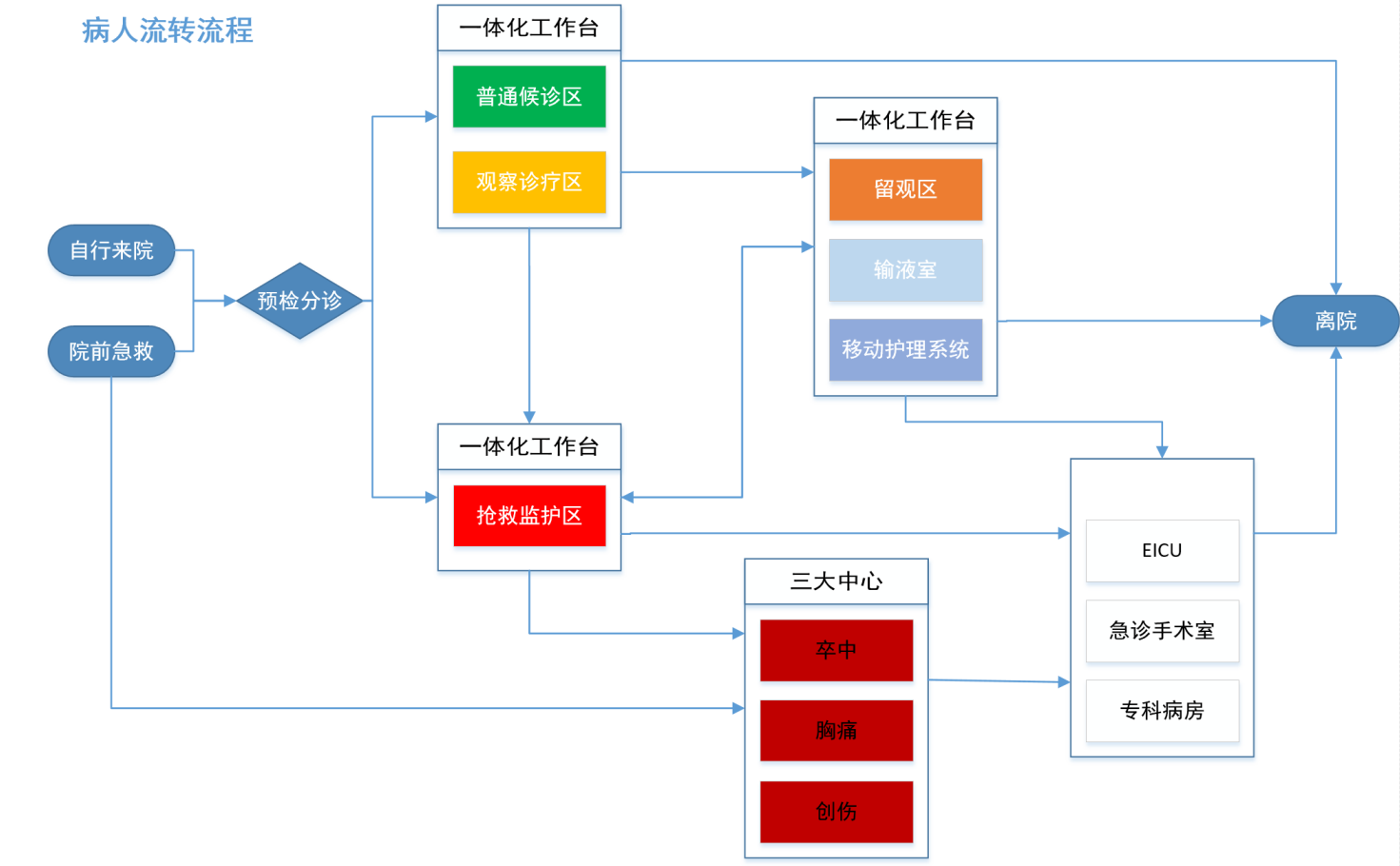
* 技术标准与规范

1. IHE医用信息系统集成标准
2. HL7(医疗信息接口通用标准)
3. XML
4. COM/DCOM

# 总体设计

## 概述

急诊科医疗区的布局包括预检分诊台、诊区、抢救区、留观区、EICU、输液区和支持区。其中诊区包括转科诊室（内科、外科、儿科、妇科、产科、五官科、耳鼻喉科、眼科、口腔科等）以及治疗室、处置室、清创室等。抢救区包括非固定床位，容纳X张病床。留观区包括非固定床位（亚重症监护）和固定床位。支持区包括挂号收费、急诊药房、急诊检验、急诊CT、X光和急诊手术室等部门。根据诊疗需要，急诊病人随时可能会在多个不同的区域之间流转。其中急诊特色的流程比较集中在分诊、抢救和留观这些环节中，目前大多数医院的急诊留观室都采用类似门诊流程进行管理。流转过程如下图所示：



## 系统环境描述

### 运行环境

#### 软件环境

|  |  |  |
| --- | --- | --- |
| **分类** | **名称** | **版本** |
| 操作系统 | Windows系统 | 7及以上 |
| 浏览器 | 谷歌浏览器 | 23及以上 |
| 托盘程序 | Szyjian.Winform.Tray | 1.0 |

部署环境：跨平台部署，支持windows、Linux系统

#### 硬件环境

|  |  |  |
| --- | --- | --- |
| **服务器** | **操作系统** | **推荐配置** |
| Web服务器 | Windows/CentOS | 16G内存，500G硬盘，8核 |
| 应用服务器 | Windows/CentOS | 32G内存，500G硬盘，8核 |
| 数据库服务器 | Windows/CentOS | 16G内存，500G硬盘，8核 |

##### 数据库服务器2台

用以安装急诊系统数据库服务

###### TPCC值估算

约定：

系统同时在线用户数为25人（诊区10个用户，抢救区医生和护士10个用户，留观区5个用户计算，合计25。U1）；

平均每个用户每分钟发出26次业务请求（N1）；

系统发出的业务请求中，更新、查询、统计各占1/3；

平均每次更新业务产生8个事务（T1）；

平均每次查询业务产生15个事务（T2）；

平均每次统计业务产生28个事务（T3）；

一天内忙时的处理量为平均值的2倍；

考虑服务器保留30％的冗余；

服务器需要的处理能力为：

TPC-E=U1\*N1\*（T1+T2+T3）/3\*2/（1-冗余系数）/60

则数据库服务器的处理性能估算为：

TPC-E= 25\*26\*（8+15+28）/3\*2/0.7/60= 526tpsE（每秒处理事务数）

以目前主流国产服务器的测试为基准值，测试结果如下：每颗CPU单核的处理能力为93tpsE。则可计算所需数据库服务器CPU核数及应用服务器CPU核数。

CPU核数 = 526/93 = 6（核）

###### 内存估算

业务服务器内存主要由操作系统占用内存、数据库系统占用内存、并发连接占用内存等几部分组成。

约定：

操作系统占用约3G内存空间；

数据库占用内存5G ；

文件系统占用内存1G；

每个并发连接占用10M；

考虑服务器内存保留30％以上的冗余；

则服务器的内存估算为：

Mem =(3GB + 5GB + 1G + 25\*10M) /(1-30%) = 13.2GB

###### 存储容量估算

系统盘：操作系统50G 数据库30G 内存数据库30G 文件系统20G

一年总共需用系统盘存储容量：50G+30G+30G+20G=130G

系统中存储着急诊业务以及日志等信息。

在已经考虑了数据冗余的前提下，约定：

每个病例附件约为0.5M；

每天病人约500例；

一年总共需用数据盘存储容量约：（10M\*30\*365）/1024=90 G

考虑到业务增长的需求，按照每年数据增长 10% 来计算，5年的数据累积量将是首年数据量的6.1倍（1+1.1+1.1\*1.1+1.1\*1.1\*1.1 + 1.1\*1.1\*1.1= 6.1），所以五年总共需要数据盘的存储容量约：

90G \* 6.1 =549GB

###### 1.1.4. 建议配置

根据以上的性能指标建议数据库服务器标准配置如下：

数据库服务器：CPU八核

内存：16G

存储：1TB SAS（做RAID 1 0）

服务器数量：2台（其中一台为热备服务器）

##### 业务服务器2台

用以安装急诊各个业务微服务10余个。一台为主服务器，另一台为热备服务器。主服务故障后，备用服务器可以自动接管。

###### 1.2.1. TPCC值估算

在同一套系统下，应用服务器访问连接数相当于数据库在线连接数，单系统为25连接/秒，按照经验，每个连接数相当于应用服务器6tpsE的处理能力。

则应用服务器TPC-E =25\*6tpsE\*4 =600tpsE（每秒处理事务数）。

CPU核数 = 600/93= 6.45核

###### 1.2.2. 内存估算

该服务器内存主要由操作系统占用内存、业务系统占用内存、并发连接占用内存等几部分组成。

约定：

操作系统占用约3G内存空间；

单个系统应用占内存约4M；

每个并发连接占用约10 M；

考虑服务器内存保留约30％的冗余；

则服务器的内存估算为：

Mem =(3G +4M\*20 + 15\*10M) /(1-30%) = 15.71 GB

###### 1.2.3. 存储容量估算

应用服务器主要包括操作系统和业务应用系统。

其中操作系统约占约50G；

单个业务系统约占约1G；

共占系统盘空间约为50G + 1G\*20 = 70G

考虑到业务增长的需求，按照每年数据增长 10% 来计算，5年的数据累积量将是首年数据量的6.1倍（1+1.1+1.1\*1.1+1.1\*1.1\*1.1 + 1.1\*1.1\*1.1= 6.1），所以五年总共需要数据盘的存储容量约：70GB \* 6.1 =420.7GB

###### 1.2.4. 建议配置

根据以上的性能指标建议数据库服务器标准配置如下：

数据库服务器：CPU八核

内存：16G

存储：500G SAS（做RAID 1 0）

服务器数量：2台（其中一台作为热备服务器）

### 开发环境

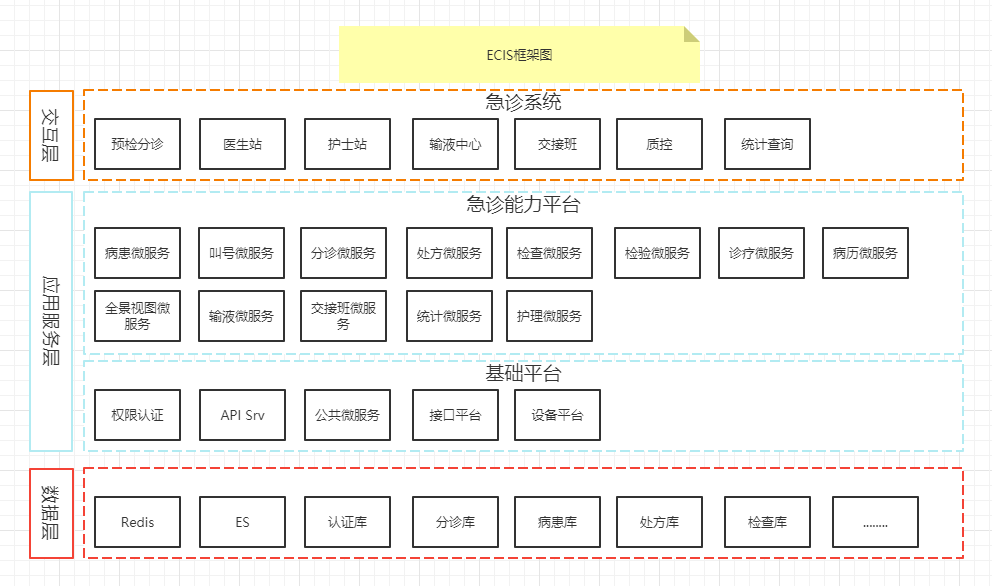
#### 软件环境

|  |  |  |
| --- | --- | --- |
| **分类** | **名称** | **版本** |
| 操作系统 | Windows系统 | 10 |
| 数据库平台 | SQL Server | 2019 |
| 浏览器 | 谷歌 | 23及以上 |
| .NET | .NET Core | 3.1 |
| 前端框架 | VUE | 2.6.1 |
| 微服务架构 | ABP VNext | 3.3 |
| 消息中间件 | RabbitMQ | 3.8.9 |
| 服务注册发现 | Consul | 1.9.0 |
| ORM框架 | EFCore | 3.1 |
| 远程字典服务 | Redis | 0.8.8.384 |
| 网关技术 | API Six | 1.1 |
| 报表打印 | DevExpress | 20.1 |
| 日志组件 | NLog | 4.9.3 |
| 分布式事务 | CAP | 3.1.2 |
| 对象映射器 | Mapster | 7.0.1 |
| 链路追踪 | SkyWalking | 1.1.0 |
| 日志管理 | ELK | 7.10 |

#### 硬件环境

|  |  |  |
| --- | --- | --- |
| **服务器** | **操作系统** | **推荐配置** |
| Web服务器 | CentOS 7 | 8G内存，200G硬盘，8核 |
| 应用服务器 | CentOS 7 | 16G内存，200G硬盘，8核 |
| 数据库服务器 | CentOS7 | 8G内存，200G硬盘，8核 |

## 系统总体结构设计



ECIS V2.0 主要以患者为中心，针对患者就诊过程中的核心业务进行功能实现。患者到达急诊科后经预检分诊进行分类，分类出科别、轻重，分辨出患者的病情（四级），最后决定患者就诊的先后顺序。一二级患者直接进入抢救区抢先就诊，三四级患者根据分诊的先后顺序依次叫号就诊。医生通过急诊诊疗功能页接诊患者，从而下达医嘱、书写病历完成诊疗过程。

内容描述：ECIS V2.0主要包括以下模块：

* **急诊叫号**

1. 护士操作端

功能：用于展示三四级患者的就诊科室及动态展示各个科室已就诊、未就诊、过号的病患队列。针对特殊患者可以优先调整叫号顺序。

1. 医生叫号端

功能：根据叫号队列依次顺叫、重叫患者，未就诊患者给与过号标记。

1. 患者显示端

功能：循环展示各个科室病患姓名、分级信息，对叫号中的患者突出显示并以广播的形式通知患者就诊。

* **急诊诊疗**

1. 诊区
2. 抢救区
3. 留观区
4. 历史就诊

功能：医生、护士接诊患者的入口，诊区以列表的形式展示病患列表，抢救区、留观区以床卡的形式展示病患信息，历史就诊方便快捷查找病患信息。

* **医生工作站**

1. 患者主页

功能：展示患者的全部基本信息，包括档案信息，挂号信息、分诊信息、接诊信息、生命体征信息、评分信息、流转信息。

1. 诊断

功能：可逐条下达诊断。可标记诊断类型。可快捷开立诊断。可查看当前病患历史诊断。

1. 医嘱开立

功能：开立药品、检查、检验、处置、治疗、等项目后自动分方生成打印单据。医嘱套餐可快速开立医嘱。

快速医嘱动态展示常用药品信息等多种快捷操作方式。

1. 电子病历

功能：可用于结构化病历录入，病历模板可快速书写病历，支持急诊病历共享调用，支持多种模式打印策略。支持病历签名。

1. 用血申请

功能：用于抢救病人用血申请，需要对接医院血库系统。

1. 手术申请

功能：用于手术病人手术申请开立，需要对接医院手麻系统。

1. 会诊管理

功能：用于会诊病人申请会诊、记录会诊过程、填写会诊意见，并打印会诊记录单，需要对接医院的短信平台、钉钉、电话等平台。

1. 全景视图

功能：图形化展示患者的诊疗数据。（包括:用药、病历、检查、检验、导管、手术、生命体征数据）

1. 打印中心

功能：打印单据的预览和补打页面。

1. 患者360

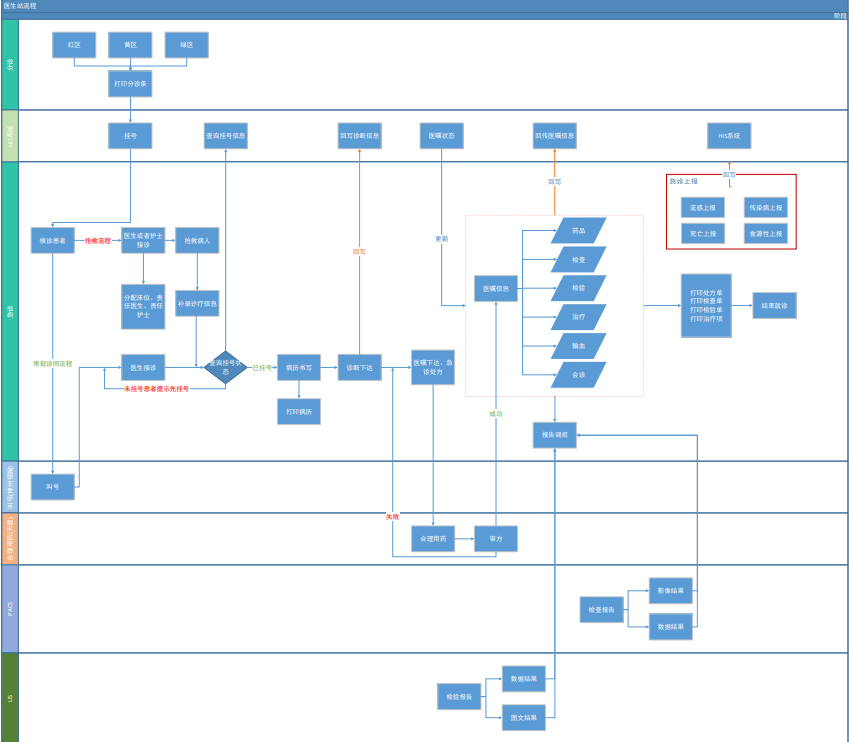
功能：展示急诊临床信息系统历次就诊记录。（包括诊断、医嘱、病历）

* **基础配置**

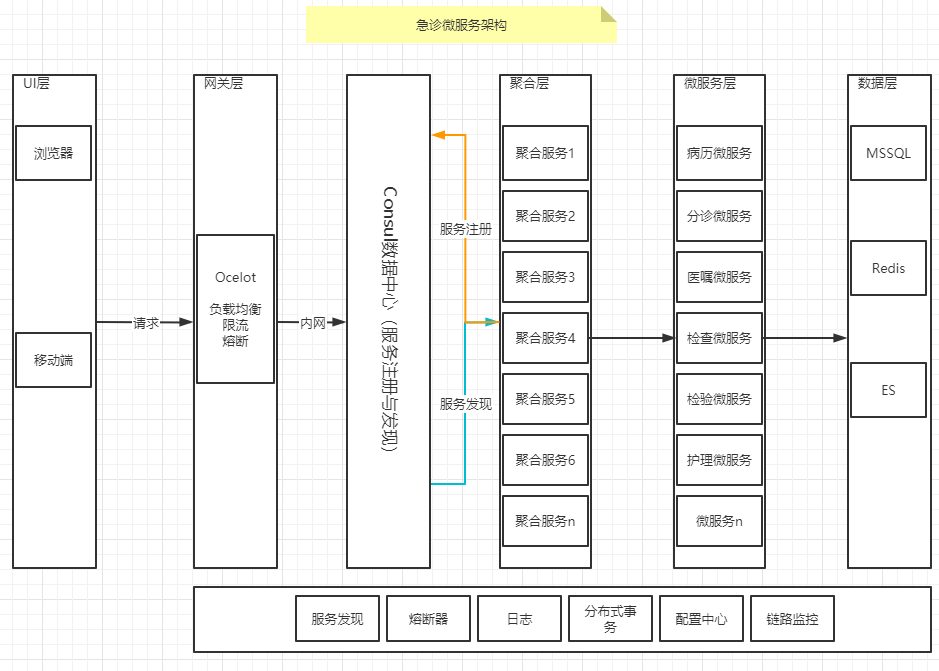
1. 医嘱设置
2. 病历模板
3. 打印模板

功能：维护系统配置

### 业务流程图



### 系统架构图



* 微服务的优势：

1. 每个服务相对较小并容易维护：微服务架构相比单体应用要小的多，开发者理解服务中的逻辑代码更容易。代码库小，打包，启动服务速度也快。
2. 服务可以独立部署：每个服务都可以独立于其他服务进行部署
3. 服务可以独立扩展：服务可以独立扩展，不论是采用X轴扩展的实例克隆，还是Z轴的流量分区方式。此外每个服务都可以部署到适合它们需求的硬件之上
4. 微服务架构可以实现团队的自治：可以根据服务来把开发团队拆分。每个团队都有自己负责的微服务，而不用关心不属于他们负责的服务。
5. 更容易实验和采纳新的技术：最后，微服务可以消除对某个技术栈的长期依赖。因为服务更小，使用更换的编程语言和技术来重写一项服务变得有可能，这也意味着，对一项新技术尝试失败后，可以直接丢弃这部分工作而不至于给整个应用带来失败的风险。
6. 更好的容错性：微服务架构也可以实现更换的故障隔离。例如，某个服务引发的致命错误，不会影响其他服务。其他服务仍然正常运行。
7. 服务可以独立扩容：对于整个架构来说，可以随意选择相关业务进行扩容和负载，通过相关技术工具动态进行随意扩容

* 微服务的劣势：

1. 服务拆分和定义是一项挑战：采用微服务架构首当其冲的问题，就是根本没有一个具体的、良好定义的算法可以完成服务的拆分工作。与软件开发一样，服务的拆分和定义更像一门艺术。更糟糕的是，如果对系统的服务拆分出现了偏差，很有可能会构建出一个分布式的单体应用；一个包含了一大堆互相之间紧耦合的服务，却又必须部署在一起的所谓分布式系统。这将会把单体架构和微服务架构两者的弊端集于一身。
2. 分布式系统带来的各种复杂性、使开发、测试和部署变得更困难：使用微服务架构的另一个问题是开发人员必须处理创建分布式系统的额外复杂性。服务必须是进程间通信。这比简单的方法调用要复杂的多。
3. 当部署跨越多个服务的功能时需要谨慎地协调更多的开发团队：使用微服务架构的另外一项挑战在于当部署跨越多个服务的功能时需要谨慎地协调更多开发团队。必须制定一个发布计划，把服务按照依赖关系进行排序。这跟单体架构下部署多个组件的方式截然不同。
4. 开发者需要思考到底应该在应用的什么阶段使用微服务架构：使用微服务架构的另一个问题是决定在应用程序生命周期的哪个阶段开始使用这种架构。
5. 跨服务数据的问题：在单体应用中，所有的数据都在一个数据库中，而在微服务架构中，每个服务都有自己的数据库，想要获取，操作其他服务的数据，只能通过该服务提供API进行调用，这样就带来一个问题，进程通信的问题，如果涉及到事务，那么还需要使用Saga来管理事务，增加了开发的难度。

* 微服务的拆分原则

1. 1、单一职责、高内聚低耦合
2. 2、微服务粒度适中
3. 3、考虑团队结构
4. 4、以业务模型切入
5. 5、演进式拆分
6. 6、避免环形依赖与双向依赖
7. 7、DDD（可以考虑使用领域驱动设计去进行底层服务的设计,后续会单独分析该设计的相关文章）

#### **微服务集群**

微服务需要一个服务注册发现的数据中心，这里用到了Consul，Consul主要用来注册服务，及服务发现，以及服务的健康检查，我们可以根据需要针对某些业务服务进行自动扩容，添加服务器及扩张服务集群，一台服务挂了，Consul会自动选择可用的服务节点进行连接使用，这样系统稳定性大大增大。

#### **微服务如何保证数据的一致性**

在微服务架构中，每一个服务一个数据库。这样可以保证微服务独立开发，独立演进，独立部署，独立团队。由于一个应用是由一组相互协作的微服务所组成，在分布式环境下由于各个服务访问的数据是相互分离的，服务之间不能靠数据库来保证事务一致性。这就需要在应用层面提供一个协调机制，来保证一组事务执行要么成功，要么失败。



ServiceComb是华为开源的微服务框架，它是华为云微服务框架引擎很重要的一个组成部分。

目前ServiceComb主要有三个项目组成：第一个是ServiceCenter做服务发现的，这是go语言在etcd 基础实现的服务注册中心，另外是java的一个微服务框架，这块是基于vertx的基础上实现了全异步操作接口，框架有比较好的性能。 Saga项目其实就是我们针对前面刚刚提到Saga模式的提供的一个实现。

#### **微服务开源框架**

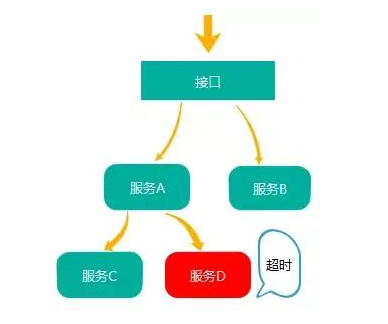
基于ABP VNext 强大的基础设施搭建底层微服务架构，可以方便开发人员快速上手开发，省略重复性的工作。

#### **ORM框架**

EFCore 是微软推出的跨平台ORM框架，相较于EF版本，更加轻量级。

#### **链路追踪**

随着微服务架构的流行，一些微服务架构下的问题也会越来越突出，比如一个请求会涉及多个服务，而服务本身可能也会依赖其他服务，整个请求路径就构成了一个网状的调用链，而在整个调用链中一旦某个节点发生异常，整个调用链的稳定性就会受到影响。



对以上情况， 我们就需要一些可以帮助理解系统行为、用于分析性能问题的工具，以便发生故障的时候，能够快速定位和解决问题，这里用到了SkyWalking ，它是一款优秀的国产 APM 工具，包括了分布式追踪、性能指标分析、应用和服务依赖分析等。

#### **系统日志集成**

庞大的系统中离不开日志系统，排查问题，记录相关敏感信息等都需要一个日志系统，这里选择使用Nlog日志系统，日志写入到ES中，并支持可视化UI进行日志管理，查询，平常遇到问题，直接通过日志管理后台进行排查。

#### **消息队列**

消息队列中间件是分布式系统中重要的组件，主要解决应用耦合，异步消息，流量削锋等问题。实现高性能、高可用、可伸缩和最终一致性架构。使用较多的消息队列有ActiveMQ、RabbitMQ、ZeroMQ、Kafka、MetaMQ、RocketMQ。

#### **任务调度**

这里主要使用的是Quartz.Net 进行作业任务调度，任务调用有什么用处呢？，比如我们需要统计一个数据，但是实时统计需要一大堆的连表查询，并且比较损耗数据库的性能，因此可以选择使用任务调度的方案进行数据统计作业，半夜某个时间点去统计前一天的数据。

#### **Redis**

Redis，可以用来在API网关和数据库层面做一层数据缓存，访问一些不是经常更新的数据，把它缓存起来，每次网络请求过来就可以先通过从分布式缓存中进行数据读取，减少对数据库的查询压力，提高系统的吞吐量。

#### **可视化数据管理及分析（Kibana）**

Kibana 是为 Elasticsearch设计的开源分析和可视化平台。你可以使用 Kibana 来搜索，查看存储在 Elasticsearch 索引中的数据并与之交互。你可以很容易实现高级的数据分析和可视化，以图标的形式展现出来。

#### **DevExpress 报表**

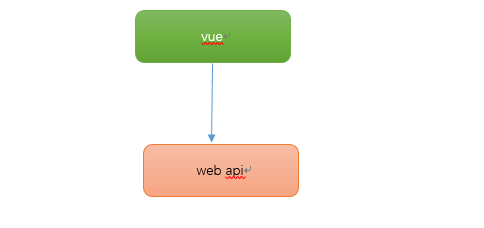
它附带了易于使用的Visual Studio报表设计器和丰富的报表控件集，包括数据透视表、图表，因此您可以构建无与伦比、信息清晰的报表。

### **软件架构图**

* **架构**

服务端采用：DotNet Core+WebApi

前端采用：Vue+Route+ElementUI+Axios



### 逻辑架构图

每个微服务均采用领域驱动设计的经典分层架构。

**Controllers(接口层)**：请求应用层以获取用户所需数据，发送命令给应用层执行某个命令。

**Application(应用层)**：定义软件要完成的任务。对外定义各种应用功能，包含查询或者命令，对外调用领域层完成各种业务逻辑。应用层不包含业务逻辑。

**Domain(领域层):** 表达业务概念，业务状态及业务规则。是业务软件的核心。

**Infrastructure(基础设施层):** 提供层间通信，为领域层实现持久机制。

上述逻辑分层有以下优势：

1. 改变某个层，不会引起对其他层的冲击，或将影响降到最低。
2. 遵循关注点分离原则，讲逻辑功能相似的移到同意层次，增加灵活性与可维护性。
3. 支持并行开发。只要各个层次有清晰的接口，开发组可以基于接口进行测试和开发，提高效率与质量。
4. 各个层次之间是松耦合关系，可以单独更新某个层的组建内容。

图表, 箱线图

描述已自动生成

系统在物理上可以分为三层即客户端、应用服务器、数据库服务器。客户端包含PC、移动客户端。应用程序服务器主要是基于微服务的网络应用，数据库服务器是基于SQL Server 的数据库引擎。

上述物理分层的优势：

1. 数据库服务器与业务服务器分离，使得业务独立部署，为以后的急危重症一体化提供载体。
2. 数据库服务器只需要提供数据检索服务而不需要处理业务服务。使得处理效率更高。

# 外部接口

### 硬件设备

1. **扫码墩**

用于接诊患者时扫码患者打印凭条中的条码

1. **读卡器（身份证、医保卡、就诊卡）**

用于采集身份证信息，医保卡信息，就诊卡号信息

1. **高拍仪**

用于病历拍照归档

1. **监护仪**

用于获取生命体征数据

1. **血糖仪**

用于获取指尖血糖指标

1. **呼吸机**

用于获取呼吸值指标

1. **血气机**

用于获取血气值指标

### 软件系统

1. **集成平台**

用于集成各个系统间的数据传输

1. **HIS系统**

用于无集成平台建设方案下的数据对接（基础字典、业务对接）

1. **LIS系统**

用于检验结果的获取

1. **PACS系统**

用于检查结果的获取

1. **CA电子签章系统**

用于数字化证书认证

1. **合理用药系统**

用于医嘱用药审查、药物说明书查看

1. **用血系统**

用于用血申请

1. **手麻系统**

用于手术申请

1. **短信平台**

用于会诊申请通知

# **模块设计**

## 患者微服务

#### 功能概述

以患者数据为中心的核心业务处理，包括：

患者入科记录

患者经治记录

患者流转记录

患者诊断记录

患者病患时间轴

患者转住院申请记录

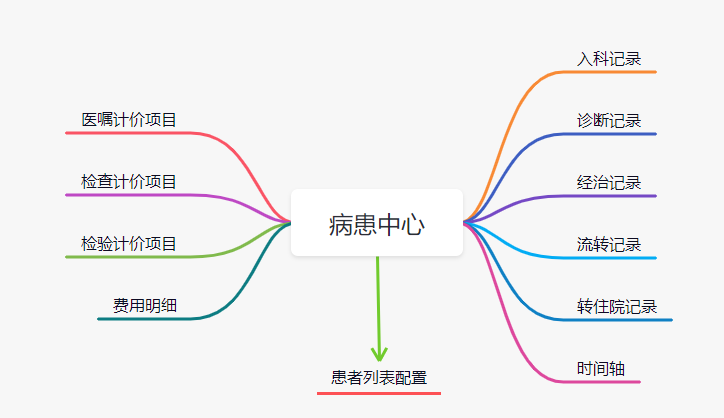
患者医嘱计价项目

患者检查计价项目

患者检验计价项目

患者费用明细表

#### 业务处理流程



**输入**：分诊病患信息

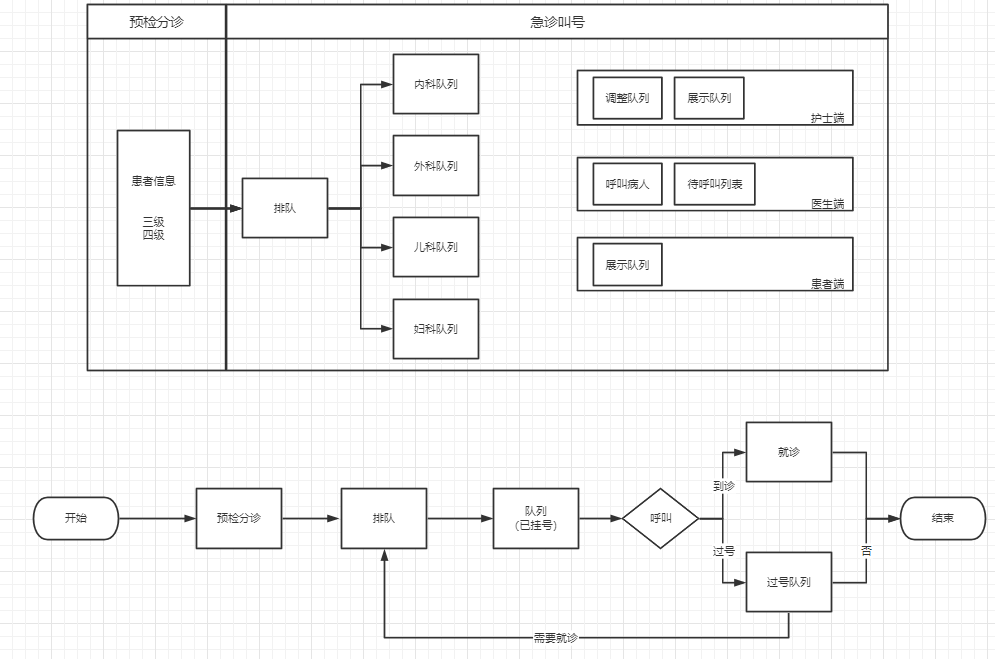
**输出**：病患相关业务表

## 叫号微服务

#### 功能概述

分诊排队叫号模块是指在急诊科家属等候区所使用的智能化呼叫和分诊排队管理系统。医生和护士可以通过该系统有秩序的呼叫就诊患者。

#### 业务处理流程



**输入**：分诊病患信息、病患挂号信息、科室医生信息

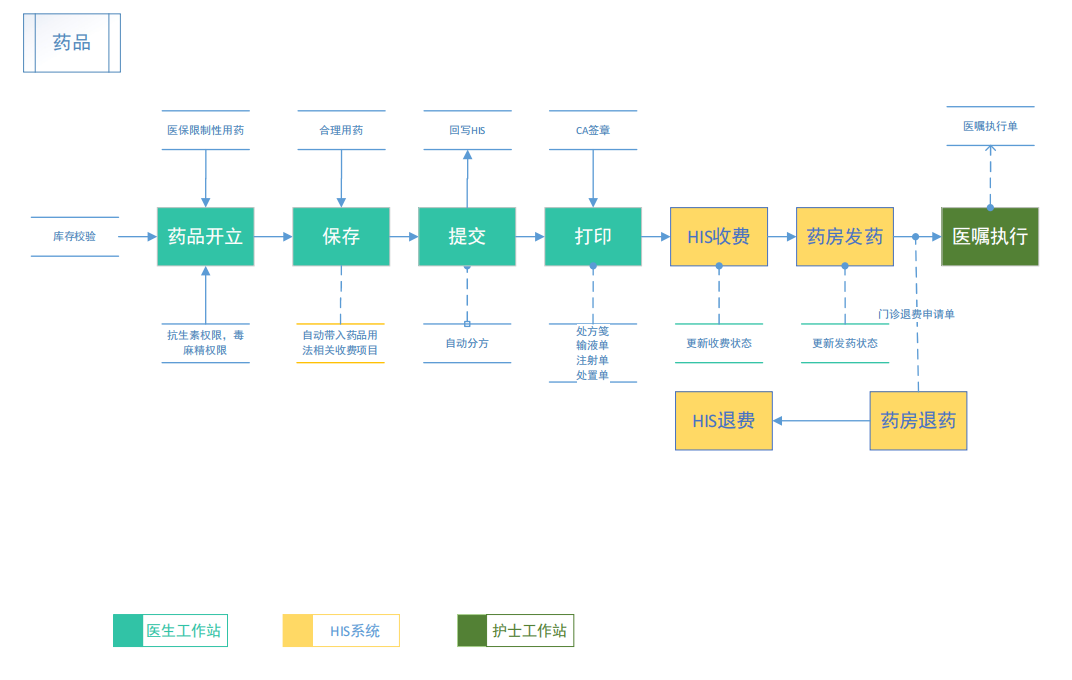
**输出**：科室病患队列信息（已就诊、待就诊、过号）

## **处方微服务**

#### 功能概述

处理药品开立相关核心领域业务，包括药品套餐维护。

#### 业务处理流程



**模块功能**：



**输入**：药品字典、药品频次、药品用法、频次拆分表

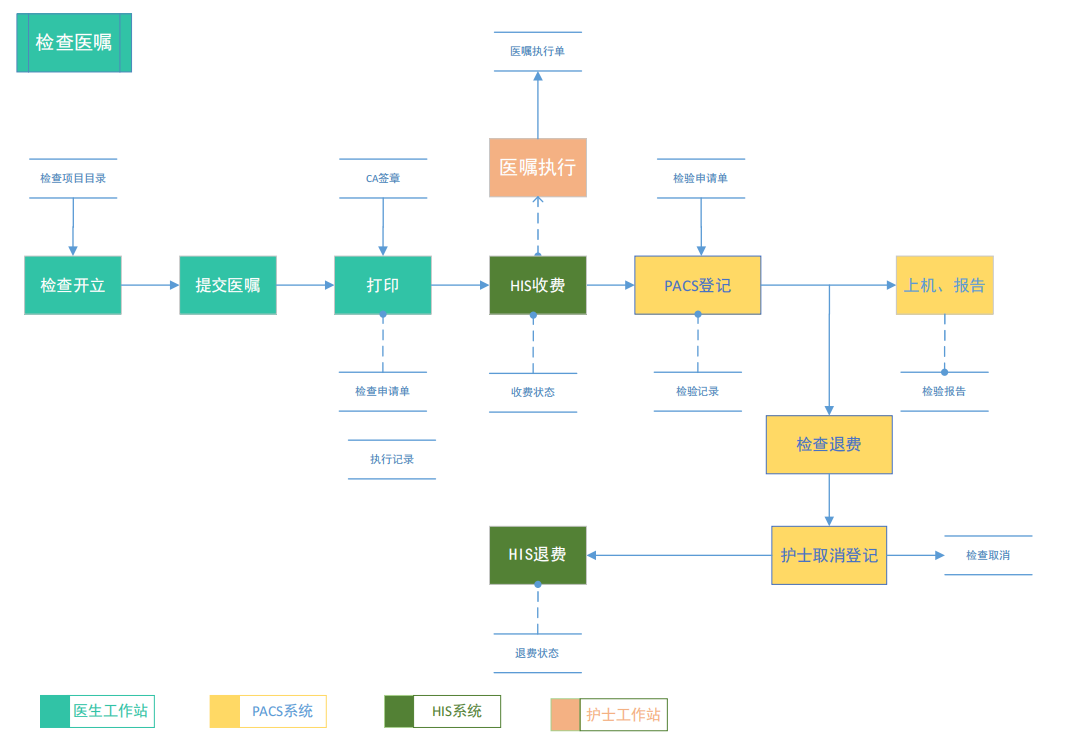
**输出**：药品主表、药品子表、医嘱拆顿表、药品套餐表

## **检查微服务**

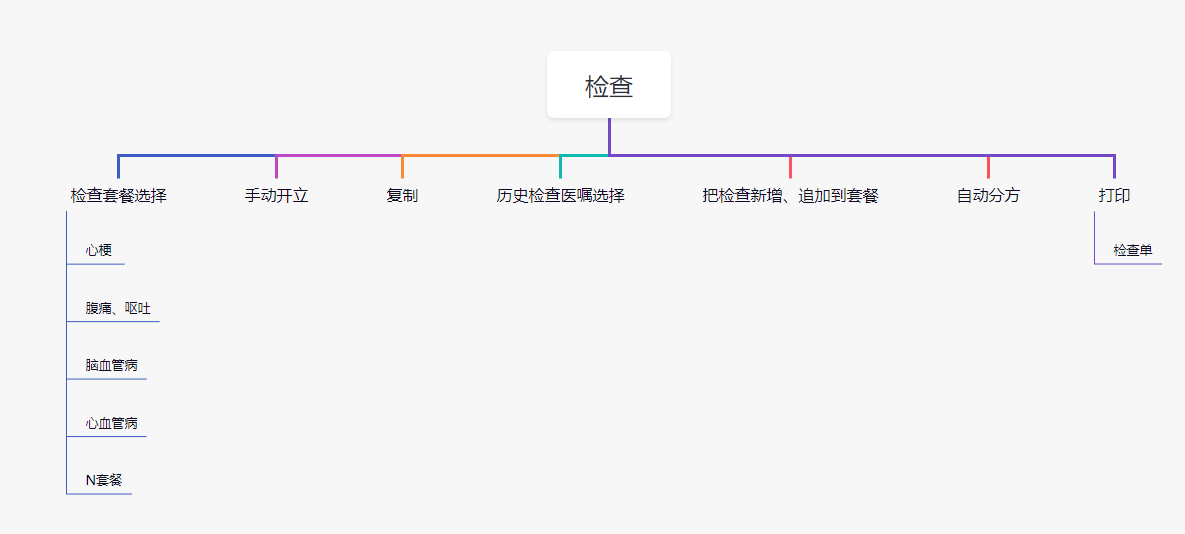
#### 功能概述

处理检查开立相关核心领域业务，包括检查套餐维护。

#### 业务处理流程



**模块功能**：



**输入**：检查项目主表、检查项目子表、检查部位表、检查申请科室表

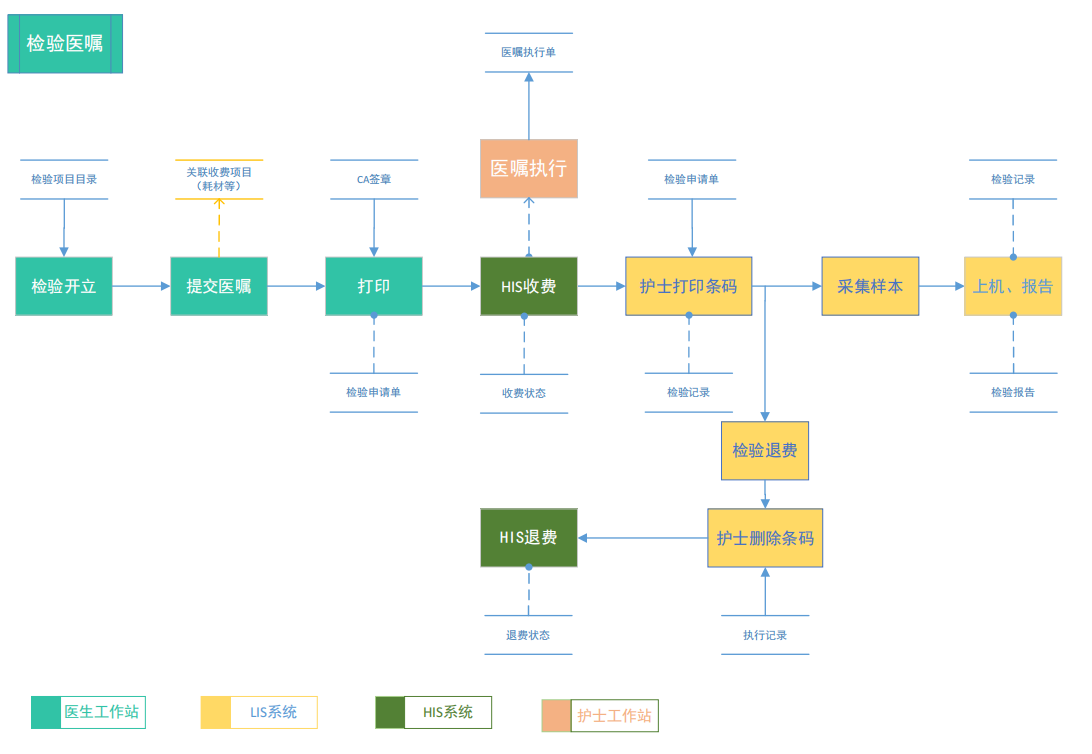
**输出**：检查申请主表、检查申请子表、检查套餐表

## **检验微服务**

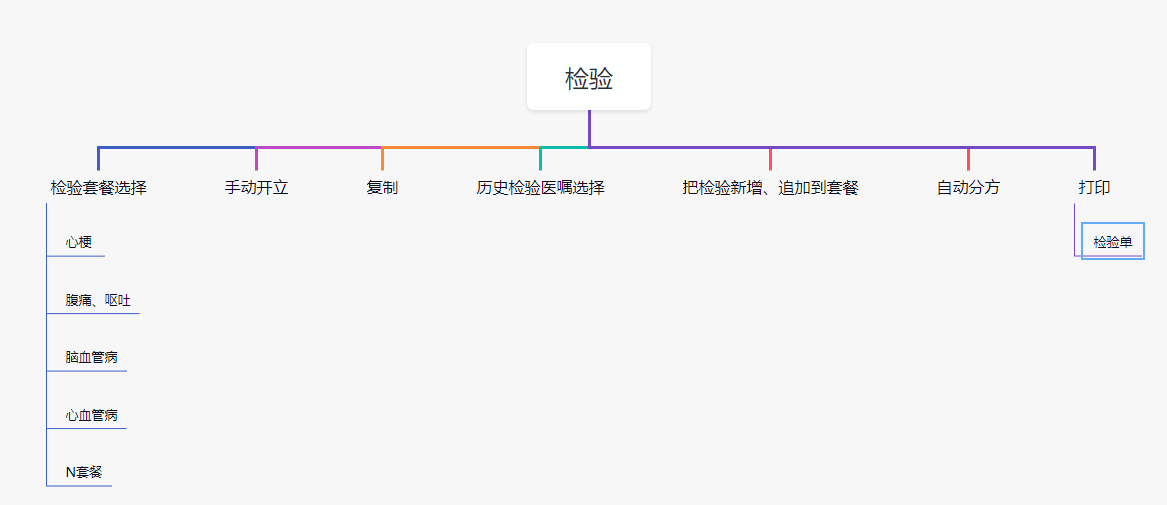
#### 功能概述

处理检验开立相关核心领域业务，包括检验套餐维护。

#### 业务处理流程



**模块功能**：



**输入**：检验项目主表、检验项目子表、检验标本表、检验申请科室表

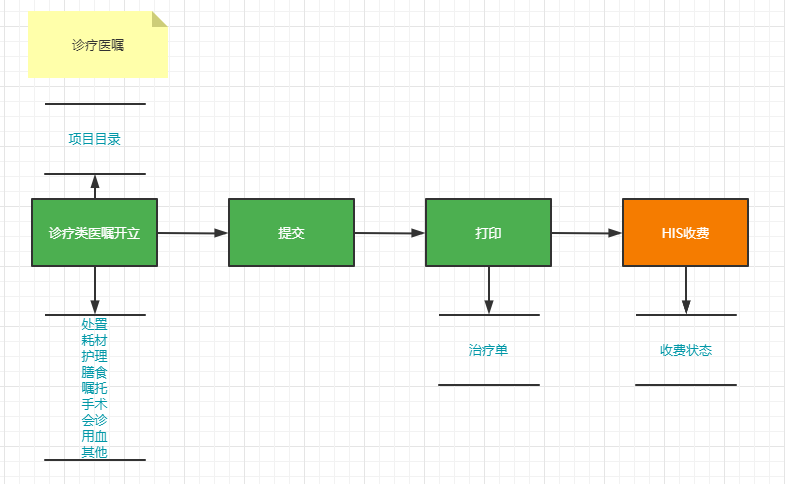
**输出**：检验申请主表、检验申请子表、检验套餐表

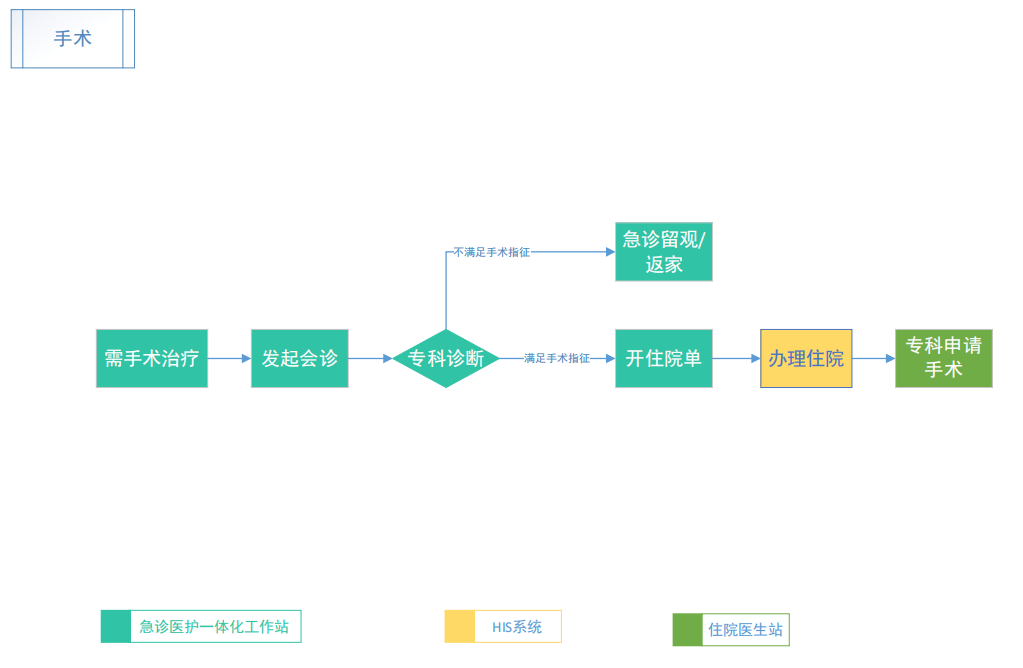
## **诊疗微服务**

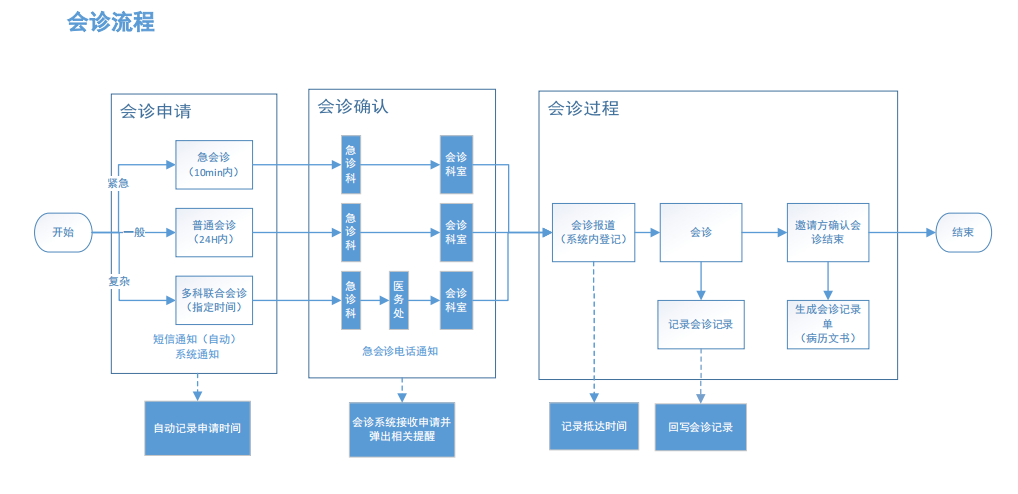
#### 功能概述

处理诊疗开立相关核心领域业务，包括诊疗套餐维护、手术申请、会诊申请、会诊记录、会诊记录单打印。

#### 业务处理流程







**模块功能**：



**输入**：诊疗字典、科室字典、手术字典、人员字典、医嘱数据、病历数据、诊断、报告

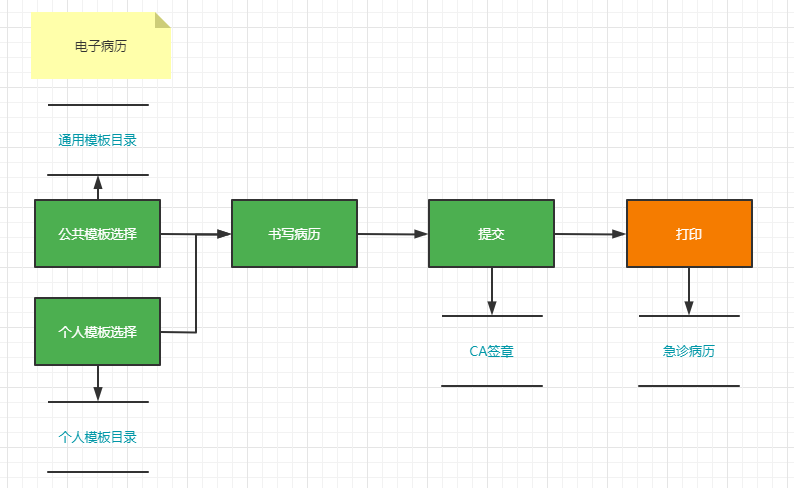
**输出**：诊疗主表、手术申请表、会诊申请主表、会诊申请明细表

## **病历微服务（同院前病历微服务共用-基于都昌的实现）**

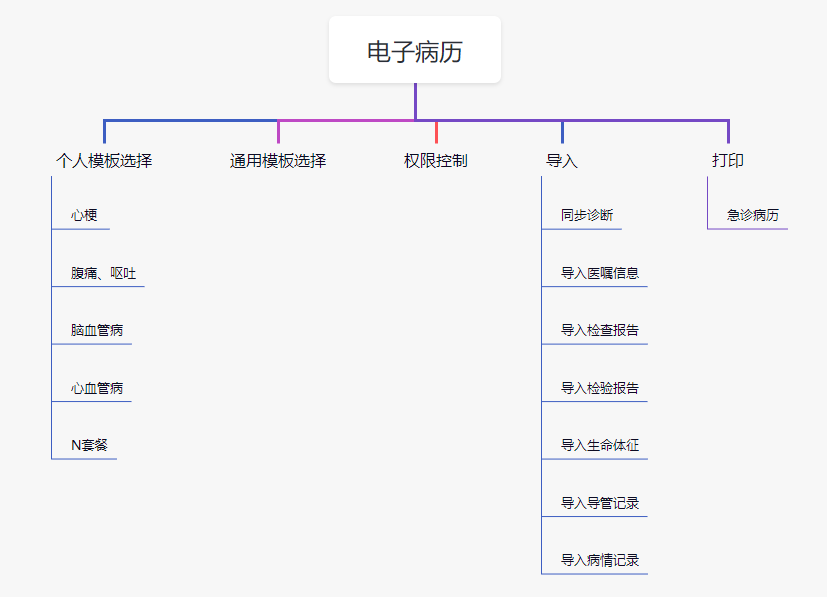
#### 功能概述

可用于结构化病历录入，病历模板可快速书写病历，支持急诊病历共享调用，支持多种模式打印策略。支持病历签名。

#### 业务处理流程



**模块功能**：



**输入**：个人模板、通用模板、医嘱信息、诊断信息、报告信息、导管信息、病情信息

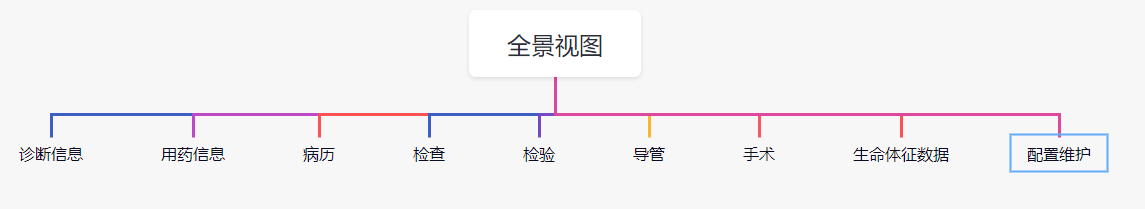
**输出**：急诊病历

## **全景视图微服务**

#### 功能概述

图形化展示患者的诊疗数据。（包括:用药、病历、检查、检验、导管、手术、生命体征数据）

#### 业务处理流程



**输入**：病人数据

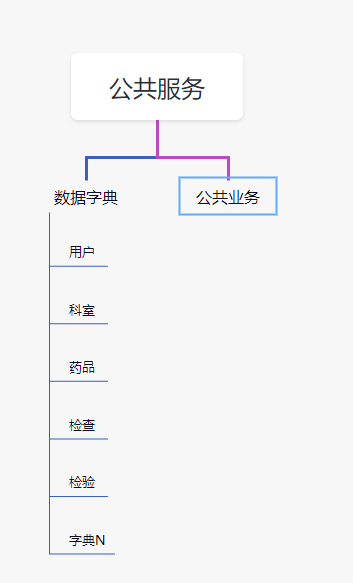
**输出**：病人数据展示

## **公共微服务**

#### 功能概述

公共领域模块，负责（数据字典、公共业务）

#### 业务处理流程



**输入**：病人数据

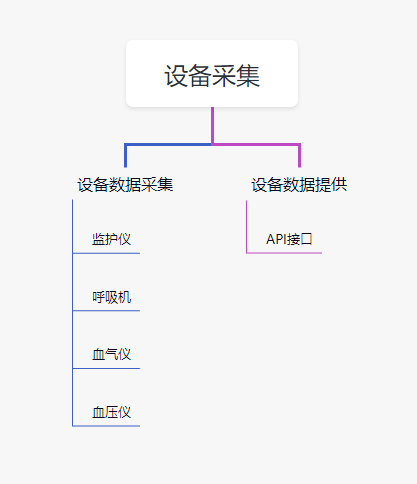
**输出**：病人数据展示

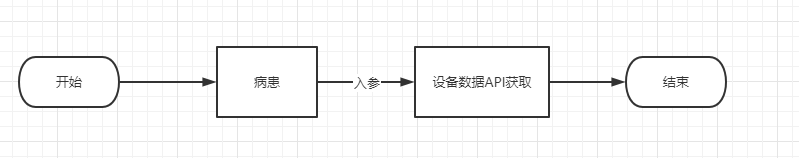
## **设备采集微服务**

#### 功能概述

根据当前病患绑定的设备获取设备数据，呈现给业务系统。

#### 业务处理流程





**输入**：API入参

**输出**：设备数据

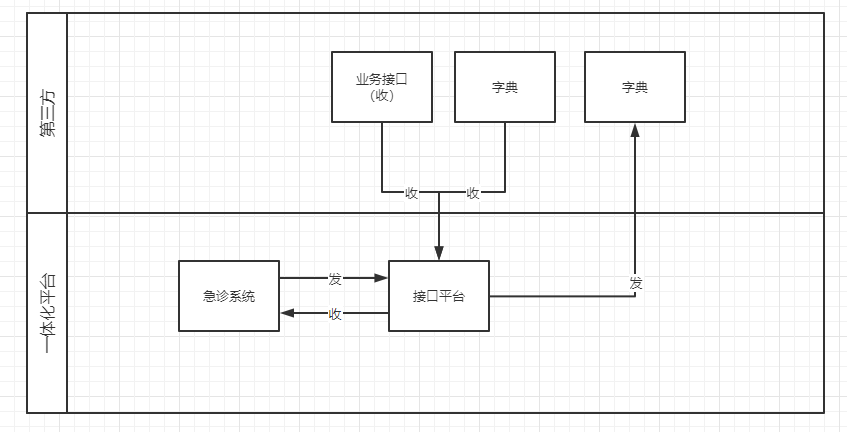
## **接口平台微服务**

#### 功能概述

1.获取医院字典数据，同步业务接口数据（挂号信息、收费信息等）

2.推送急诊业务数据到接口平台（开嘱信息）

#### 业务处理流程



# 数据结构

* **业务数据：MSSQL**

病患微服务：ECISPatient (YiJian.ECIS.Patient )

叫号微服务：ECISCall (YiJian.ECIS.Call )

处方微服务：ECISRecipe (YiJian.ECIS.Recipe )

检查微服务：ECISPacs (YiJian.ECIS.Pacs )

检验微服务：ECISLis (YiJian.ECIS.Lis )

诊疗微服务：ECISTreat (YiJian.ECIS.Treat )

* **日志数据：ES**

搭配ELK使用

* **缓存数据：Redis**

字典数据、需要缓存的业务数据

# 容灾设计

## 出错情况

本设计要考虑的出错情况主要是指软件运行、系统部署时可能出现的异常，包括：

1. 用户不规范的输入导致的程序错误；
2. 系统运行时容器崩溃；
3. 用户环境和开发时环境不一致造成的程序错误；
4. 编码中的逻辑错误或者漏洞导致系统报错；
5. 不包括由于硬件损伤、网络中断等情况的容错处理。

## **异常处理**

本系统将采用如下的一般性设计原则减少程序和用户交互过程中可能出现的错误：

**1. 一致性**

1. 菜单选择、数据显示以及其它功能都应使用一致的格式；
2. 提供有意义的反馈信息；
3. 执行有较大破坏性的动作前要求确认；
4. 在数据录入上进行尽可能严格的合法性检测；
5. 减少在动作间必须记忆的信息数量；
6. 允许用户非恶意错误，系统应保护自己不受致命的破坏；
7. 提供和运行环境相关的帮助机制。

**2. 本系统对于出错信息和警告应该遵循以下原则**

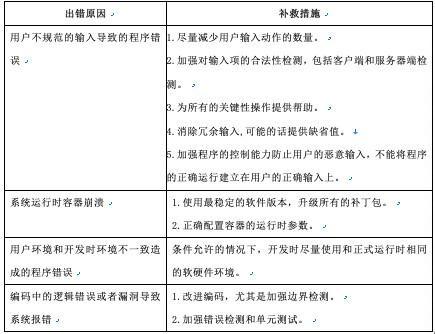
1. 信息以用户可以理解的术语描述；
2. 信息应提供如何从错误中恢复的建设性意见；
3. 信息应指出错误可能导致哪些不良后果，以便用户检查是否出现了这些情况或帮助用户进行改正；
4. 信息应伴随着视觉上的提示，如特殊的图像、颜色或信息闪烁；
5. 信息不能带有判断色彩，即任何情况下不能指责用户。

系统采用统一的异常捕获和处理机制，为了便于团队开发的一致性，统一定义错误代码和友好显示信息。开发过程中根据具体情况可以扩展错误信息，制定更加详细的错误分类和信息显示。

为规范、统一各类系统错误或业务提示信息，统一定义公共信息列表。根据信息的性质与应用范围，将公共信息分类列表如下：



## **补救措施**



## **容灾备份**

利用RAID5数据存储解决方案，实现实时数据备份，保障了系统的安全稳定运行。

## **系统维护设计**

**1. 设备运行维护**

设备运行维护指对网站群网络、服务器硬件设备、网络安全设备的运行管理维护，保证硬件设备正常运行。

设备运行维护由专业的硬件工程师进行，处理日常的设备管理、运行配置等操作。

设备运行维护按照设备操作手册与要求执行。

**2. 应用系统运行维护**

主要负责对网站群所有应用系统软件的运行维护管理工作，一般由网站群系统管理员负责，处理应用系统日常数据维护、运行配置、软件应用异常处理等。

系统管理员将根据日志信息记录对系统进行维护处理。在该系统中，将规范统一系统日志管理，系统日志的信息级别分为一般信息、调试信息、警告信息、严重错误信息。

**3. 系统安全与数据备份恢复维护**

由网站群系统安全管理员负责网络硬件、软件安全管理，同时，按照规范处理日常的数据备份工作，对系统安全异常情况，立即进行恢复应急处理。

日常数据备份内容一般包括数据库数据备份和网站用户上传的文件数据备份。

系统安全管理员严格按照安全管理制度进行日常管理工作。

# **监控设计**

* **Skywalking**:

它是一款优秀的国产 APM 工具，包括了分布式追踪、性能指标分析、应用和服务依赖分析等。

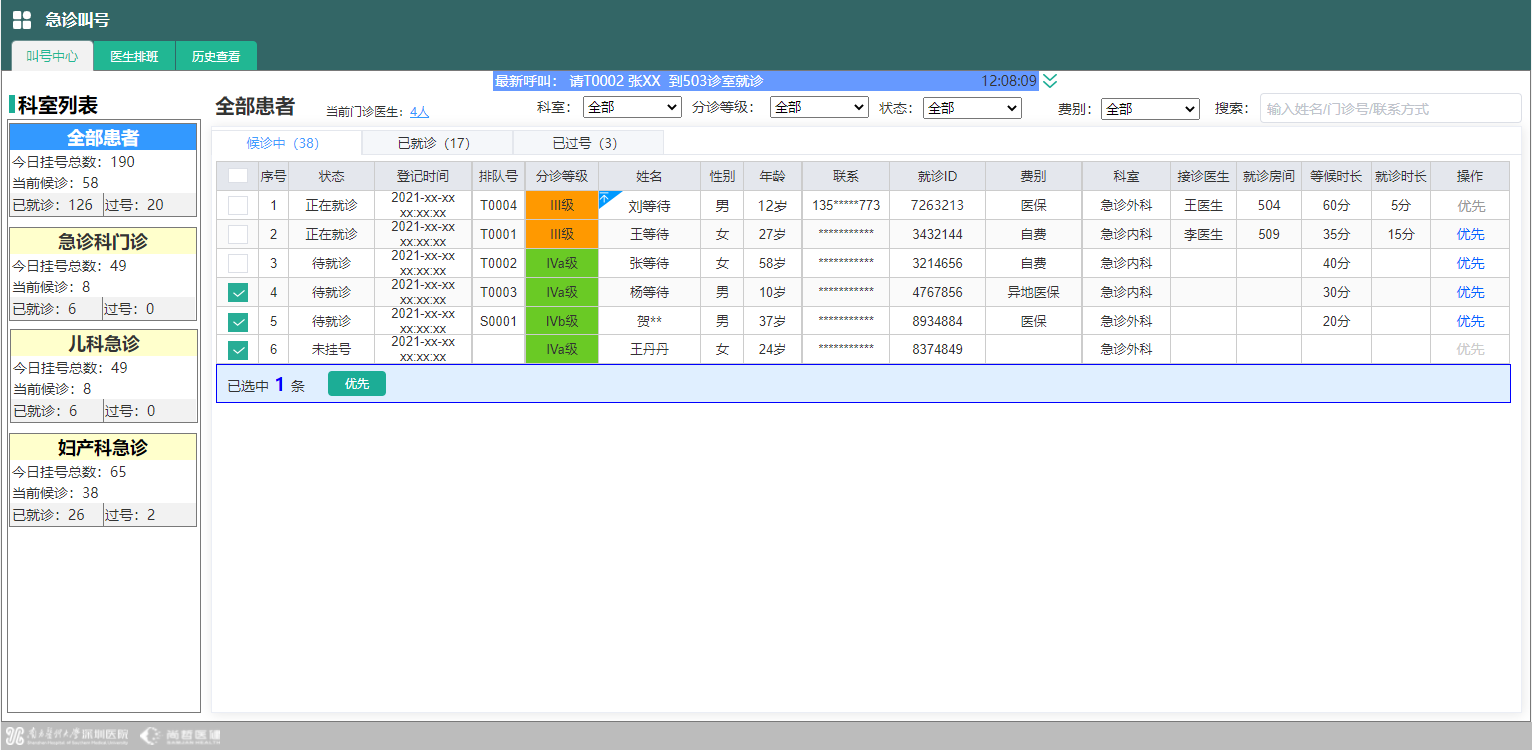
* **Kibana**:

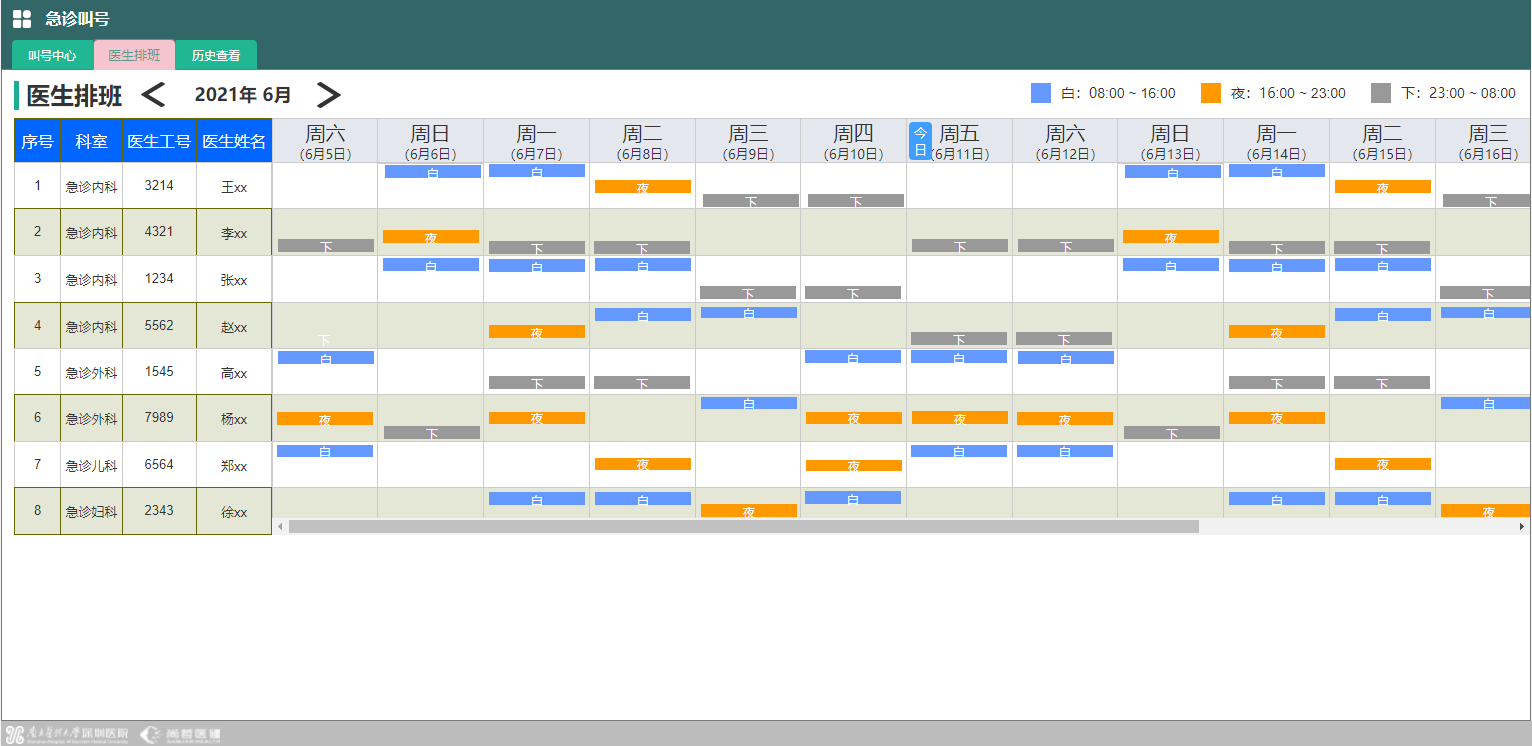
它是为 Elasticsearch设计的开源分析和可视化平台。你可以使用 Kibana 来搜索，查看存储在 Elasticsearch 索引中的数据并与之交互。你可以很容易实现高级的数据分析和可视化，以图标的形式展现出来。

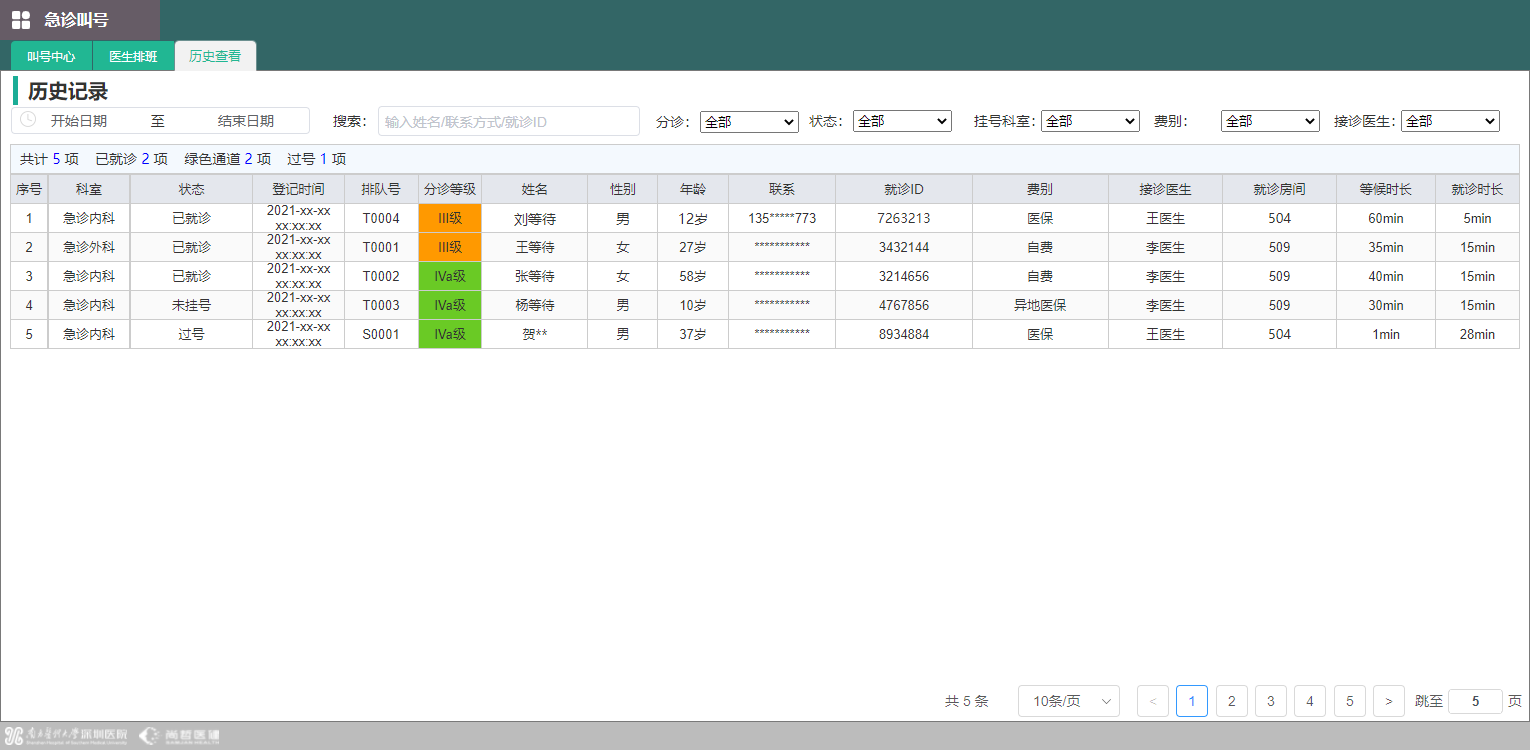
# 用户界面设计

## 急诊叫号

### 护士端







### **医生端**



### **患者端**



## **急诊诊疗**

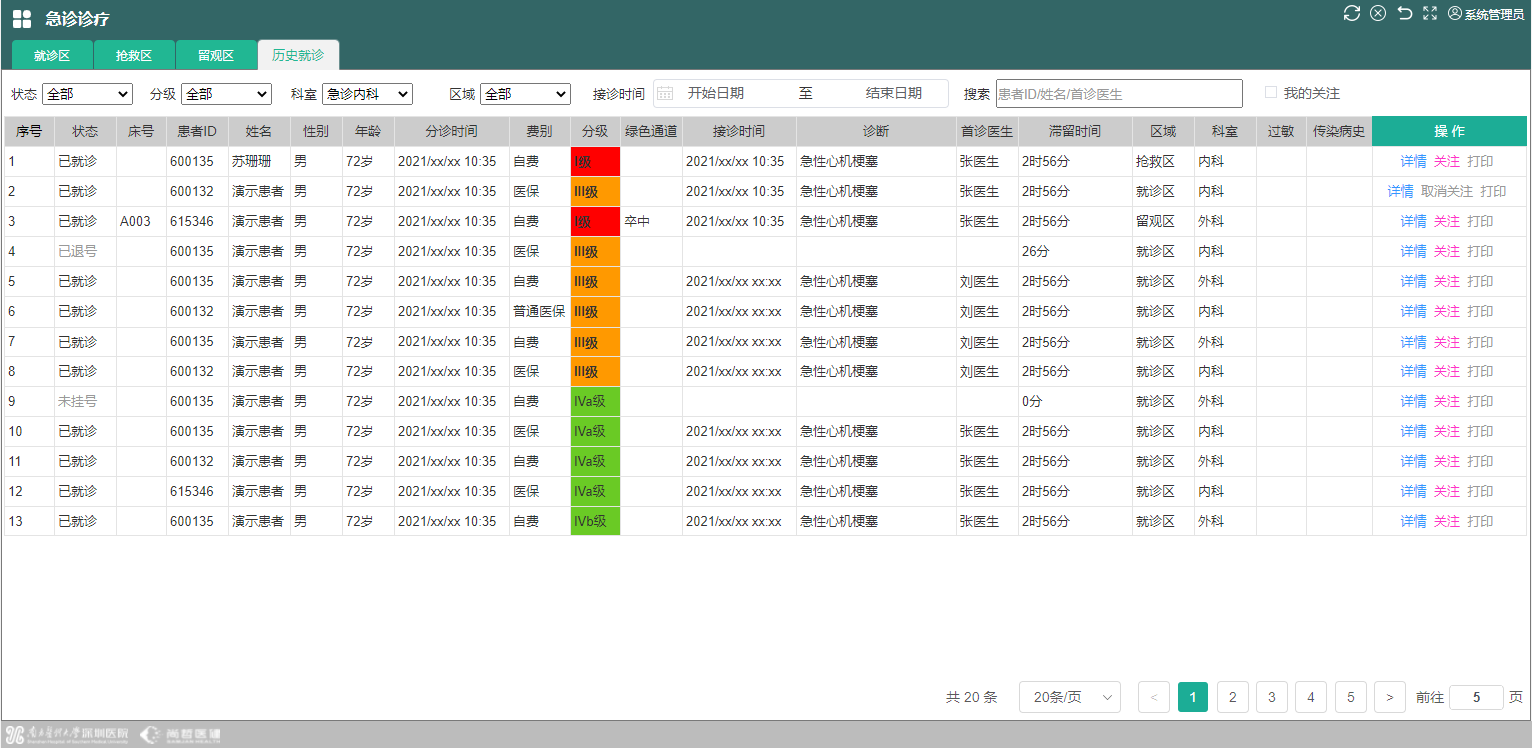
### 诊区



### **抢救区、留观区**



### **历史就诊**



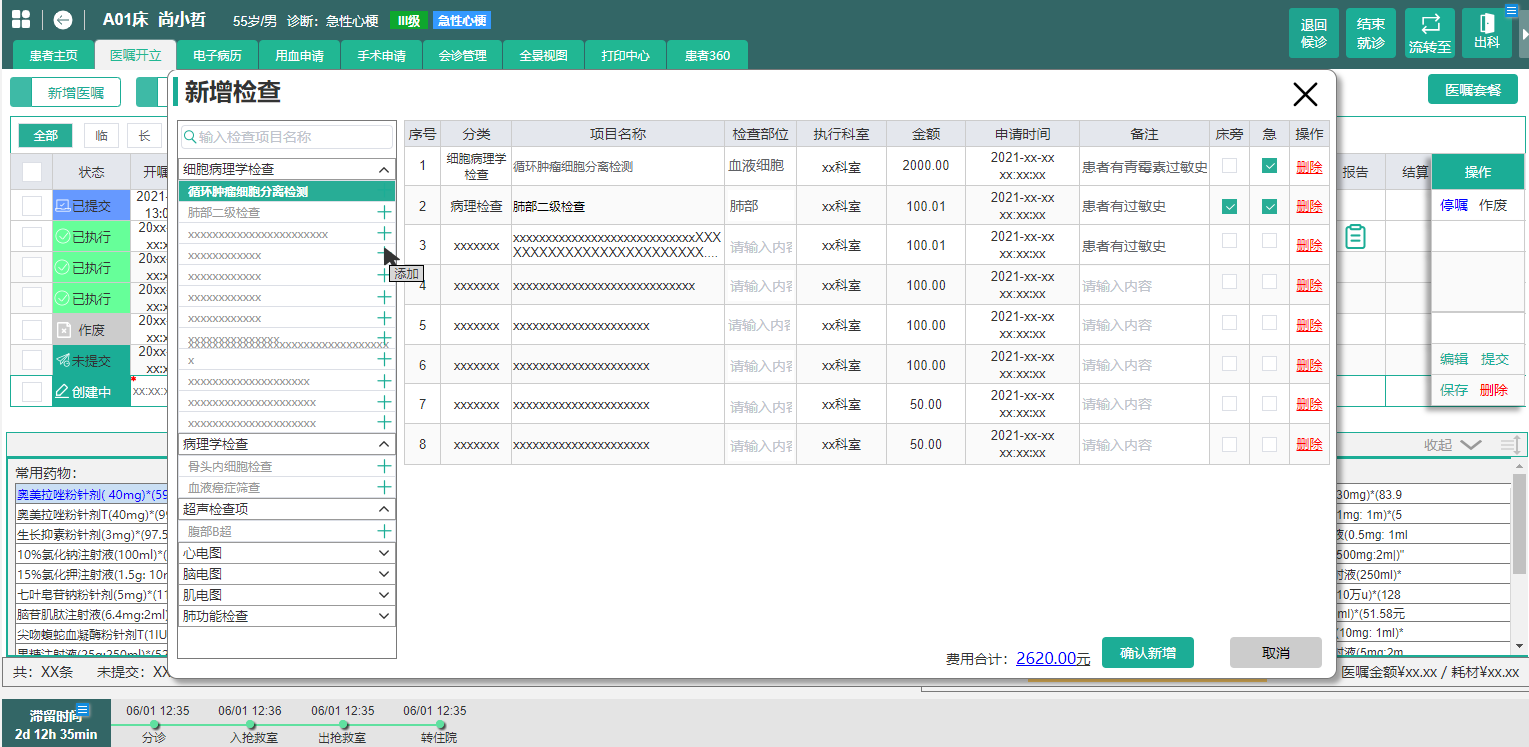
## **医生站**

### 患者主页



### **医嘱开立**







### **电子病历**



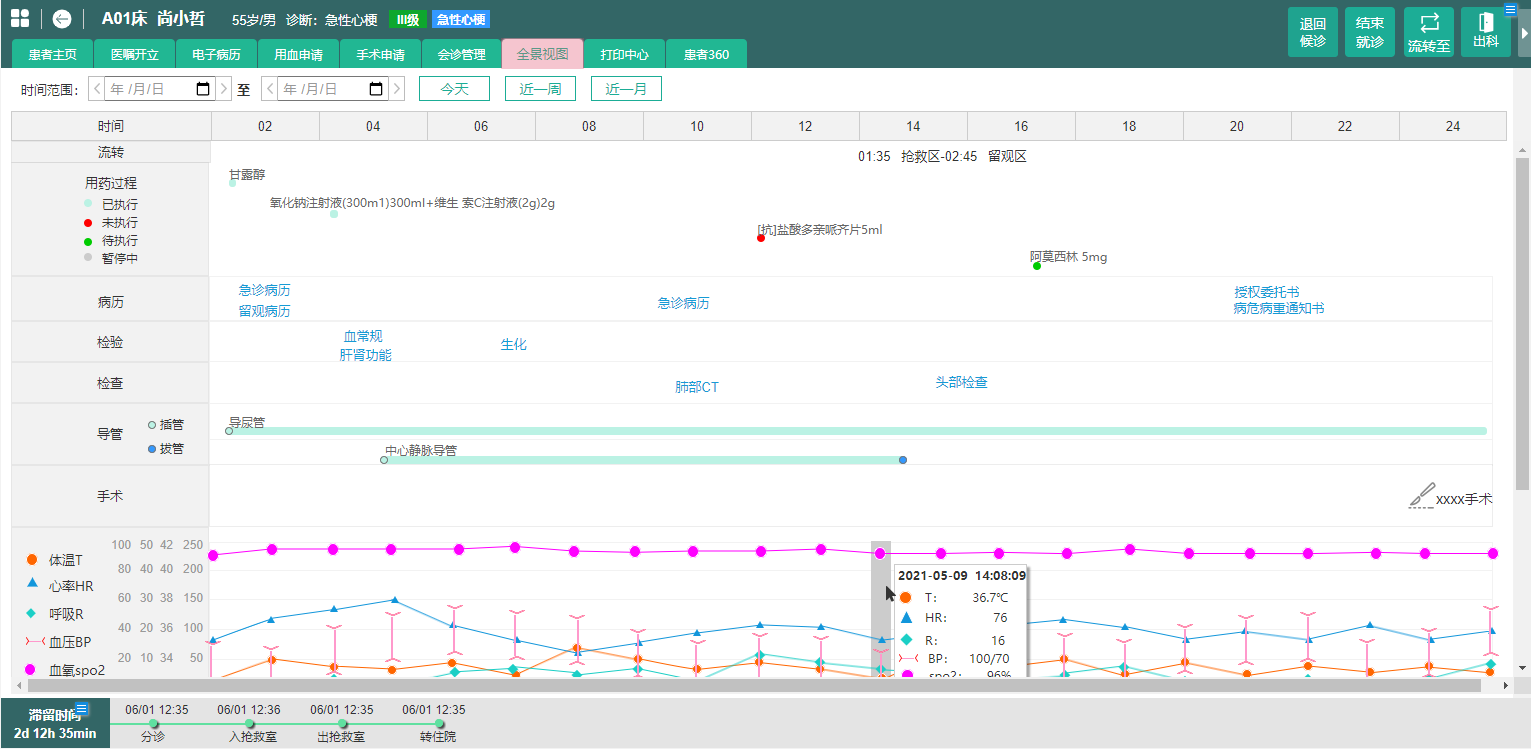
### **手术申请**



### **会诊管理**



### **全景视图**



### **打印中心**



### **患者360**



# 系统出错处理设计

系统出错最严重的就是系统没有保证用户的用户名、密码和资金的安全性，为了从根

本上解决这个问题，首先在编写前台和后台代码时就要注意编写的严密性，不能让不怀好

意的人利用保留的攻击手段侵入系统，盗用、毁坏用户的信息，造成用户的损失。

## 出错信息

a)程序在运行时主要会出现两种错误：

1、由于输入信息，或无法满足要求时产生的错误，称为软错误。

2、由于其他问题，如网络传输超时等，产生的问题，称为硬错误。

b)对于一些关键的操作（如：删除操作），应该提供提示确认机制；

c)对于数据、测试文档，都是要提供相应的保密措施设置；

## 补救措施

a)定期建立数据库备份，一旦服务器数据库被破坏,可以使用最近的一份数据库副本

进行还原。

b)为防止服务器故障,预备另外一台服务器,只要主服务器出现故障,可以迅速启动预

备服务器运行系统。

## 系统维护设计

1)基础数据维护:对于一些基础数据,安排管理员进行维护。

2)数据库备份和恢复:利用SQLServe自身提供的备份和恢复功能实现。

3)系统升级维护:根据用户使用效果调查表,筛选用户提出的功能要求，对于合理的要

求予以采纳,并安排人员对系统进行修改和完善。