|  |
| --- |
| eds |
| 系统架构说明书 |
| EDS |
|  |
| **赫杰辉** |
| **2016/1/8** |

|  |
| --- |
| Version：0.1 |

# 简介

本文档包含系统架构相关说明。

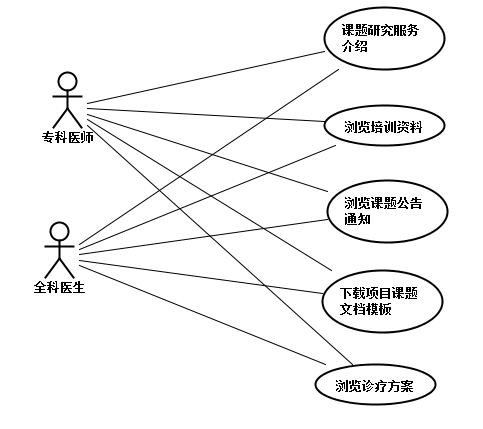
# 功能需求

功能需求由用例图说明。具体用例请参考用例说明文档

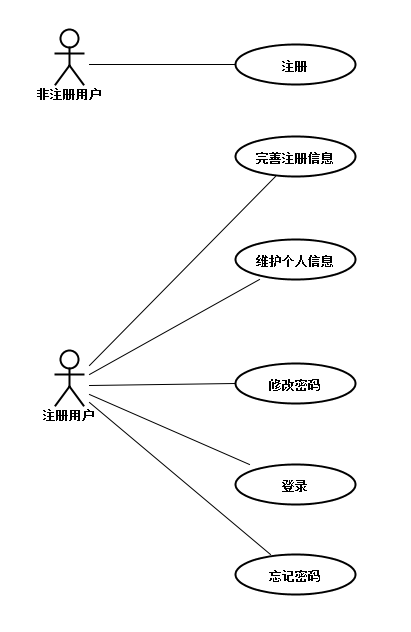
## 工作站



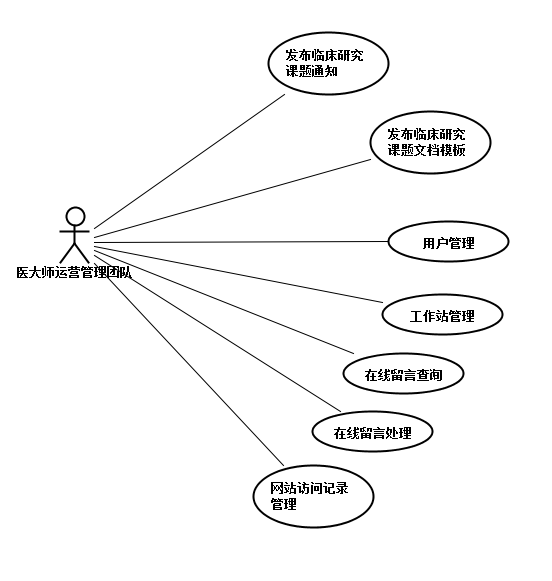
## 医师培训



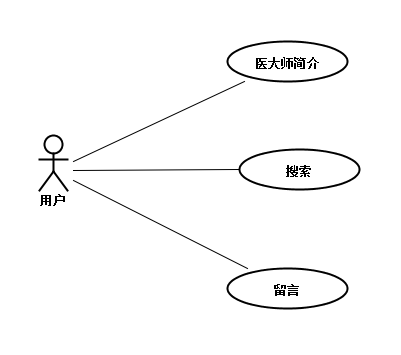
## 用户管理



## 系统管理



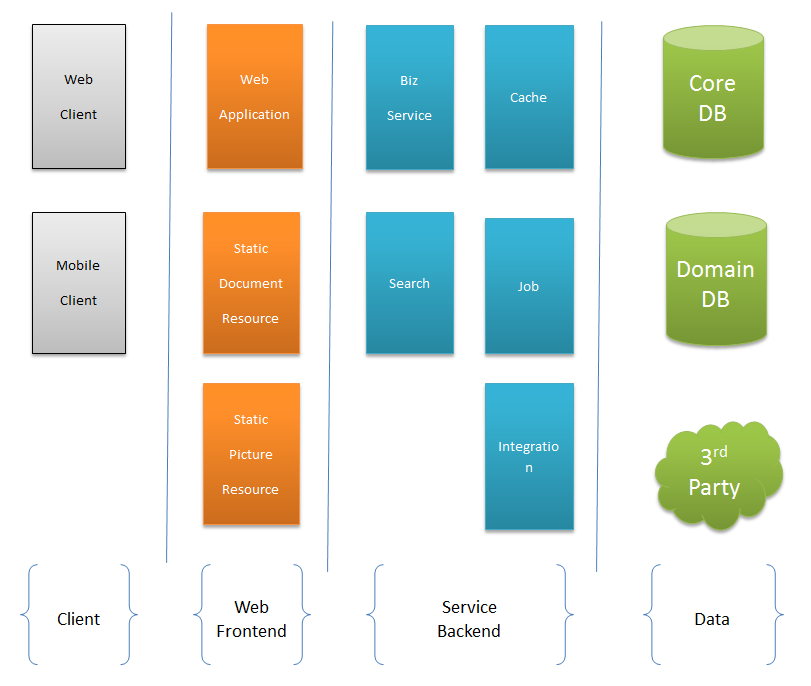
## 其他



# 非功能需求

|  |  |
| --- | --- |
| 非功能需求点 | 描述 |
|  |  |
|  |  |
|  |  |
|  |  |

# 逻辑架构图



### 层次说明

#### Client

系统的客户端。包括：

1. 浏览器
2. 移动端用户

#### Web Frontend

系统对外提供服务的入口。包括：

1. Web Application。提供网站内容访问的服务器。生成动态网页或提供RESTFUL接口，实现use case基本的功能
2. Static Document Resource。提供静态文档访问的服务器。包括访问频率高的网页或内容片段，培训资料，文档模板等
3. Static Picture Resource。提供静态图片访问的服务器

#### Service Backend

系统逻辑层。包括：

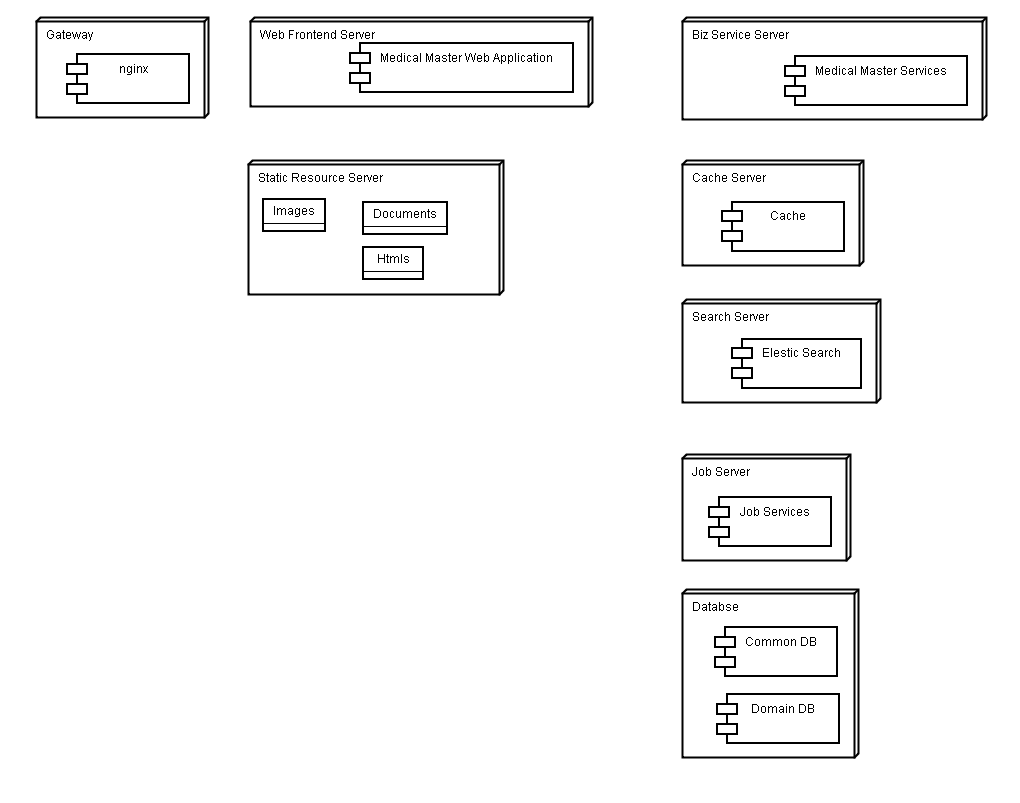
1. Biz Service。系统的逻辑实现。提供针对每个具体领域模型的操作。
2. Search。搜索服务器。提供文档扫描，索引和搜索功能
3. Cache。缓存服务器。提供热点数据的缓存服务，加快网站访问速度，减轻数据库压力
4. Job。批处理服务器。对定时或可异步处理的工作提供支持。用于统计报表或生成静态化网页等
5. Integration。提供第三方数据访问接口。方便系统获取其他系统的数据

#### Data

数据层。包括：

1. Core DB。存放系统核心功能用到的数据
2. Domain DB。按实际情况和系统发展添加的特定领域数据库。配合核心功能，但缺少也不影响。例如病例等用于浏览但和系统运行无关的数据
3. 3rd Party。外部数据源。

# 部署架构图



# 开发工具

各个开发工具选取的原则是优先考虑最新的稳定版本。下面列出主要的开发工具

1. IDE：Eclipse Mars
2. JDK：JDK 1.8
3. JS：jQuery/Bootstrap
4. System design
   1. x-series
5. Dependent module
   1. x-series
   2. Ctrip Dal
   3. Jerssy
   4. Log: log4j

# 运行环境

1. Server OS: Linux
2. JVM: 1.8
3. Database: Mysql 5.7
4. Cache: Redis
5. Configure Service: Zookeeper
6. Search and centralized logging: Elastic Search
7. Monitoring: Zabix

# Restful service开发规范

Rest相当于领域模型的web接口。领域模型本身也可以直接通过api做进程内访问

1. 定义资源
   1. 不允许用@singleton
2. 定义操作
   1. Get：查询
      1. 缺省为得到列表，如果数量庞大需定义条件，例如分页信息或其他过滤信息。行内容为简略信息，包括主键，名字，或少量的顶层分类信息
      2. 按照reource/{id}
   2. Post：创建新对象
   3. Put：更新对象，
      1. 按照reource/{id}
   4. Delete：删除操作
      1. 按照reource/{id}
3. 定义参数对象
   1. 注意大小写要匹配
   2. 对字段多的请求定义为值对象
   3. 值对象必须要有一个缺省构造参数
   4. 通过标记QueryParam, FormParam, PathParam来影射参数
4. 定义返回对象
   1. 注意事项同上
5. 定义manager
   1. 针对特定领域模型的操作集合
6. 定义操作
7. 参数少于5个可以不定义值对象，多余5个需要定义值对象[TBD]
8. 定义DAO
9. 定义Entity

监控

1. 定义唯一id

# 项目说明

## Common

Common包含了web和service都可能用到的通用的和专有的请求和返回类，通用的xunit实现，还有帮助类。

注意

由于目前功能还不完善，所有的request的字段对应的类型都是string，后期可能去掉此类限制。返回类无此限制。

通用的请求与响应

1. 所有类型都是Context的实现
2. CommonResponse。通用的返回值，可以把额外的信息定义在基于map的result里面
3. RequestByIdContext。通用的仅包含id参数的请求。例如传user id，workstaion id等。无需重新定义
4. Status。通用的状态返回值

领域相关的专用请求与响应

1. User
2. Workstation
3. …

通用的xunit实现

1. FieldLocator。返回指定字段值作为key的locator

帮助类

1. ParseHelper。通用的类型转换工具类。为了避免从web传来的数据为空的情况下转换失败。DAL缺省会过滤掉为空的字段，如果希望update空子段，需要提供额外的hints。具体可联系我
2. ExceptionWraper。把Exception包装为RuntimeException

DAO

1. 包括所有领域模型的Entity和DAO
2. 如果有新的需要请联系我添加，后期安装好携程的code gen后可自行添加修改

## Service

Service由resource，control与manager组成

Resource

1. 在com.medicalmaster.resource包下面
2. 每个domain对应一个resource。例如UserResource, WorkstationResource
3. Resource的方法定义接受的请求类型和返回类型
4. Resource负责实例化resource对应的xunit Converter。把请求类型发给convert来convert为返回类型
5. 所有的逻辑由convert实现，resource仅仅起到service层对外声明输入输出的作用

Control

1. Resource对应的control按照domain名字划分组织在com.medicalmaster.resource.control包下面
2. Web层的action一般一一对应到service层的control
3. Control负责理解request并调用相应的manager来实现功能，并把manager的返回至包装为web层需要的respone类型
4. Control同时处理需要多个manager协作实现的功能，协作的工作在control里面处理

## Web

Web层包含系统入口的servlet， 显示的jsp，处理的action。主要的处理过程是MainServlet把http请求包装为WebContext来调用main处理器处理。

MainServlet

1. 系统入口，负责包装http请求为Context，交由main处理器处理
2. 无需修改

通用Control

1. com.medicalmaster.web.control维护web主流程用到的所有xunit组件
2. 主要由赫杰辉维护，有通用的需求提到他这边
3. WebContext为web层使用的context，保证了http request，response等和一些通用的方法
4. AccessCheckValidator，AccessValidator。判断当前用户是否有权限做事情。未完成
5. BuildResponseAction。所有的web请求最终由该类调用显示jsp。
6. UploadFileProcessor。文件上传处理器，目前没上线

Domain control

1. 在com.medicalmaster.web.control下面，按照domain名字组织。例如user， workstation
2. 包含每个需要变更领域模型的请求
3. 实现processor接口
4. 调用ResourceProxy实现真的resource的get/post/put/delete
5. 处理返回结果，设置request或session，并且指定显示的页面

帮助类

1. ResourceProxy。
   1. 把前端传入的参数转化为请求对象
      1. 依据request类里面定义的Path， Query， Form 等Param标记生成对应请求service层resource的URL。
   2. 支持Get，Post，Put，Delete
   3. 如果功能不够可联系赫杰辉添加

# 开发流程

开发指的是完成一个具体的功能，可能包括从JSP到DB的一整套流程。

Web层工作

1. 定义流程
   1. 打开main.xunit，添加新的processor到main processor里面
   2. 添加新的branch 模块在user和workstaion下面
   3. 修改main里面新添加的processor的引用指向新的模块
   4. 定义模块里面的子流程
2. 实现调用接口
   1. 生成子流程对应的web层的action
   2. 在common层里面定义一个action需要的request， response。如果可能的话请复用通用的请求和相应
   3. 调用ResourceProxy完成相应请求
   4. 单个请求需要显示和处理的数据通过WebContext的setRequestAttr设置。Session相关的调用setSessionAttr设置。设置的值可以在后继的处理器或JSP里面使用
3. 实现显示页面
   1. 所有的调用需要在结束时调用WebContext里面的setTargetPage设置要显示的返回页面
   2. 返回页面用JSP实现
   3. 所有的页面都是嵌入在index.jsp里面，如果没提供，缺省是根目录下的home.jsp
   4. 返回页面在需要包含相对路径，例如“user/home”但无需提供.JSP的后缀

JSP开发

1. 公用的放在根目录，resource和domain相关的放在对应的目录下面
2. 表单的请求都用post，表单的字段要和request的字段对应。
3. 表单需要带隐藏字段module和action来在main里面定位到对应的processor
4. 通过request.getAttribute和session.getAttribute来得到需要的信息
5. 目前如果用户登录，会在session里面存一个user对象

Service层工作

1. 定义新的resource到com.medicalmaster.resource
   1. 添加所有需要的方法以映射到get/post/put/delete
2. 定义流程
   1. 仿造user.xunit新建对应的xunit文件
   2. 添加branch unit
   3. 修改locator，赋予和user.xunit里面的locator一样的取值
   4. 对每个resource里面的方法添加一个converter在branch unit里面
3. 实现converter
   1. 在resoucre包的control包下面建立对应该resoucre名字的control包
   2. 对每个resource的convert节点实现converter接口
   3. 双击convert节点，选择converter的实现来实现绑定
   4. Convert负责接收request
   5. Convert负责把request里面的数据转换为manager可以接受的类型
   6. 调用manager实现具体的事情
   7. Convert不可以抛异常，异常需要捕捉，并通过ExceptionWrapper来包装和再次抛出
   8. 如果操作需要涉及多个manager，Convert负责协调调用。保证manager的职责最小最单一
4. 定义manager
   1. 新建对应该resource的子包在com.medicalmaster.domain下面
   2. 添加针对该resource的manager
   3. Manager里面定义对外提供的方法
      1. 注意方法需要定义自己的对象，而不能重用common里面的request和response。
      2. 这样可以保证manager不依赖其他对象，相对独立，方便集成其他客户端和独立提供服务
   4. Manager通过DAO实现对数据库的修改
   5. Manager可以抛出应用级别的异常

命名规范

1. Module的名字。小写的名词，尽量用一个单词代表，例如user，workstation，research，diagnistic，etc
2. Action的名字，小写动词，例如invite，activate，update之类的
3. Processor的名字是动名词+**Action**结尾。例如UpdateUserInfoAction
4. Converter的名字为动名词。例如UpdateInfo