**健康管理平台架构设计**

1. **上下文**



* 1. **用户**

普通用户

会员：星级

VIP

普通

医生：

专家

医生

系统管理员：

组织内管理员

admin

nova客服人员

* 1. **外部系统**

手机APP：

设备终端：

三方系统：如580

1. **系统功能概览**
2. **质量属性**

性能：在在线用户数为60万，并发连接数为1万的情况下，用户请求的响应时间小于3秒。

用户使用的具体功能是什么？

数据量

基础设施：3台负载均衡服务器，6台Tomcat服务器，2台MySQL服务器，4台分布式服务服务器，2台分布式文件服务器，1台缓存服务器，1台静态页面服务器。

*吴强*

可用性：可接受99.99%.

升级不停服务。

安全性：

根据客户要求可以使用https协议。局部使用：登录和付款等。https性能慢。非集群版https实现？

对用户使用基于角色的访问控制来保证安全。

数据安全。细化

对外部系统(包括公司其他系统)使用API 密钥的方式来保证安全。细化

网络安全：横向攻击，策略：内部系统需安全认证？

web安全

灾难恢复：

吴强讨论。

系统监控：

需要。

审计：

需记录医生的客户管理、设备管理操作。

需记录管理员的组织管理、设备管理、预警指示设置操作。

修改前的数据需要保存吗？

灵活性：

组织级配置指标

扩展性：

需要

服务平台：提供基础服务。

合规性：

产品经理

国际化：用户界面支持中文和英文。界面配成中文。

数据库的数据（如：下拉框的选项）目前不必支持。

浏览器兼容性：

由于面向互联网，目前主流浏览器都需要。具体包括：IE8以上，Firefox，Chrome，360浏览器，QQ浏览器，搜狗浏览器。

1. **约束**

* 预算
* 时间
* 技术

JavaEE；

使用HTTP REST协议兼容已有的对外部系统的接口；

部署到Linux服务器上。

Oracle JDK7

CentOS 7

springMvc

spring

具体项目用的应用服务器都不同。

* 人员

人数：人?

技术水平：3年以上经验

1. **原则**
   1. **开发原则**

* 遵守统一的代码规范
* 代码中执行SQL语句使用预处理方式：PreparedStatement，目的是防止SQL注入攻击。
* 跨站点攻击js
* 单元测试

服务层方法需做。

* 接口规范

debug版，正式版。

接口测试程序：人机界面

* 日志

向前端返回错误信息时，使用错误码和简明的提示信息，不需返回完整的错误信息。在日志中需打出错误码和完整的错误信息备查。

* 构建

分两种版本：集群版和非集群版。

非集群版系统由2个Tomcat服务器（web后端服务器和接口服务器）、缓存服务器Redis和Oracle服务器组成。因此需要把集群版的所有服务的配置文件（除provider配置文件和MyBatis mapper文件外）合并，连同集群版的前端资源文件和所有服务的jar文件及其依赖的jar文件打到web应用的war文件中，web.xml中需配置加载合并后的spring应用上下文配置文件，不需配置consumer配置文件。代码上，要把对文件的存取做成两种实现：分布式文件方式和本地文件方式，非集群版使用本地文件方式，在Spring应用上下文中配置使用的方式。

服务的jar文件打包方法：选择输出JAR file，而不是Runnable JAR file。只打包sr/main/java，由于根路径上的资源文件已手工合并到war中了，就不用打包它们了。打包时要选择选项：Add directory entries，这样就可以使用jar中的资源文件了。

由于要支持dubbox，Spring版本应为3.2.9。

* 1. **架构原则**
* 分层策略

分为三层：展现层（Spring MVC），服务层(Spring)和数据库访问层(MyBatis)。在服务层有持久化对象PO，在展现层有视图对象VO（可选），在展现层进行二者的转换。

画图

Spring MVC 3.2.9 + Spring 3.2.9 + MyBatis 3.4.1

数据库连接池用druid 1.0.26。

全文检索？

* 模块高内聚，模块间低耦合

使用三方库都需要封装后使用，这样可以解耦，便于以后对其替换。

* Http session使用

使用统一的分布式session库来处理session。使用Spring session开源分布式session库。见“9.1.分布式session库”。但它基于Spring 4.x版本，需修改源码来支持Spring 3.2.9。

* 按组件(或模块)建包

组件包内部再分层。

* 依赖注入

优先使用Spring注解装配方式，必要时使用xml配置。

* 使用微服务即分布式服务来实现服务层

服务可以组合。为了使用统一的事务控制，组合在服务层实现，展现层对每个请求只能调用1个服务。服务分为单一模块的服务和跨模块的服务，对前者使用JDBC的事务控制，对后者需在服务内编码实现事务控制。画图

* 前后端分离

视图模板引擎使用前端js模板引擎，这样就可以把页面文件都放到Ngnix服务器上来支持高并发访问。使用artTemplate，见“9.2.前后端分离”。

* 支持多种数据库

目前需支持两种数据库Oracle和MySQL。以后数据量增加到1千万条（MySQL的极限）时需支持使用NoSQL数据库MongoDB。

为支持Oracle和MySQL，需建两个mybatis-config.xml文件：mybatis-config-oracle.xml和mybatis-config-mysql.xml，分别支持这两种数据库。每个mybatis-config.xml文件都包含公用的mapper文件和该种数据库特有的mapper文件。在Spring上下文配置文件中，根据要使用的数据库来配置对应的mybatis-config.xml文件。对每个DAO接口，公用的实现放在一个mapper文件中，数据库特有的实现放在另一个mapper文件中。

画图

* 对外提供的API版本升级时要向下兼容
* 字符编码

为避免编码不一致导致的乱码问题，从页面到数据库的字符编码都统一使用UTF-8。

具体来说，包括以下方面：

1. 项目源码都使用UTF-8；

2. 数据库字符集使用UTF-8；

3. Ngnix中需设置charset为UTF-8；

4. Tomcat的server.xml 中设定 useBodyEncodingForURI="true"，URIEncoding="utf-8"；

5. html文件的header中加上：<meta charset="utf-8">；

6. 前端AJAX发起请求时，对URL或者数据中的非ASCII字符使用javascript的encodeURIComponent进行编码；

7. 在后端controller中，对要返回响应的方法加上设置：charset=UTF-8。

权限功能使用shiro

1. **软件架构**
   1. **容器图**
      1. **集群版**



Ngnix负载均衡是否可以公用？

db吴强

现有附件文件存储架构及优化

高并发请求文件：吴强

应用的context

* + 1. **非集群版**



不用redis, 单实例mysql.

* 1. **组件图**

需表示业务模块间关系。

1. **部署**
2. **运维**
3. **决策日志**
   1. **分布式session库**

候选方案有：

* Spring session库

是Spring开源项目。把所有session存入redis缓存中。应用服务器中立，设计简单。基于Spring 4.x版本，因此需修改源码来支持Spring 3.2.9。修改难度不大。

* tomcat-redis-session-manager

把所有session存入redis缓存中。个人项目，已经不维护了。发布的jar有bug，不能直接用，需手工编译源代码。与Tomcat绑定。设计复杂，维护难度大。

从维护和通用性角度考虑，建议选择Spring session库。

* 1. **前后端分离**

候选技术：

* react.js 陈锐 张总

facebook主导，轻量级，性能好，编程模型简单：使用界面组件的方式组合出界面，不需使用模板，降低复杂应用的编写难度。将来能与手机APP统一。社区非常活跃，可能是将来的主流前端框架。

成功案例：Facebook，Paypal，Instagram，Netflix，BBC，Yahoo的Email，支付宝新一代的框架基于React，淘宝订单页面，知乎的评论功能前端等，详见： <https://github.com/facebook/react/wiki/Sites-Using-React>。

* AngularJS

Google主导，编程模型复杂，大而全，性能较差。

* artTemplate

腾讯主导，简单易用，成功案例：QQ个人空间。

* baiduTemplate

百度主导，成功案例：百度贴吧。

* velocity.js

阿里主导，版本<1.0。成功案例：淘宝。

* 从长远考虑，选择react.js比较好，但目前项目时间紧，建议选择使用简单的artTemplate。

1. **遗留问题**
   1. **持续集成**
   2. **代码规范**

数据库表名：模块名\_类名，表与代码相对应，便于后期维护。

* 1. **Tomcat裁剪**

Tomcat裁剪其中不安全的模块。

* 1. **日志架构**
  2. **混淆、加密？**