

## 快速开发指南

SysX

成都双链科技有限责任公司

2019.3

# 变更历史

| 日期 | 变更类型 | 变更人 | 说明 |
| --- | --- | --- | --- |
| 2019年3月13日 | 创建 | 张喜来 | 创建初期版本 |
| 2019年3月14日 | 更改 | 张喜来 | 增加后端定制功能 |
| 2019年3月15日 | 更改 | 张喜来 | 增加前端定制功能 |
|  |  |  |  |

# 前 言

本手册分为三个部分，第一个部分是讲述如何建模，并且在5分钟内生成一个基本系统并部署，产品经理和开发均可阅读，第二部分针对后端开发人员定制系统后端，第三部分针对前端开发人员定制前端系统。每个部分都讲述原理，工具，方法，过程，验证和最佳实践。

第一部分为建模：建模的过程主要是通过客户或者产品经理提供领域模型和界面原型，使用XML语法，建立不同类型的字段和对象之间的关联。

第二部分为后端定制：后端开发通过在特定的目录中使用或者加入新的Java类，这些Java类继承生成的代码来扩展系统的后端功能。

第三部分为前端定制：前端端开发通过在特定的目录中使用或者加入新的React类，这些React类继承生成的代码来扩展系统的界面功能。

# 第一部分：建模

## 建模的原理

建模的过程主要是通过特定格式的XML文件描述对象之间的关系和自身属性的过程，系统通过这些关系和属性进行关系推演，发现基本的逻辑和业务。这些信息会在后续的处理过程中，生成前端界面，后台服务，数据库规格定义和用于测试和演示的数据。

## 建模的工具

理论上，编写XML文件任何文本编辑器都可以，我们推荐使用微软公司出品的vs code，该软件在Windows， Mac，Linux下均可以使用。

该软件的主界面如下：

## 建模的方法

在开始建模之前，一定要对业务有一个比较深入的认识，对于建模，了解高层次的分析领域模型和有哪些角色使用这个系统是必须的。界面原型是比较好的输入，可以帮助产品人员确定相关的字段。一旦确定如上事项，就可以进入编写XML文件的工作。

## 建模的过程

### 认识XML文件

XML文件是一种W3C规范的国际通用标记语言，利于计算机处理和人工编写。XML有很多高级特性，为简单起见在本手册里面只是介绍了这种语言的一个子集，便于沟通。

我们用到的是主要有属性名，属性值。

第一部分：模板

建设为his.xml, 文件名为系统名称，为英文，小写，无需下划线

<?xml version='1.0' ?>

<root **chinese\_name**="医生排班系统" **english\_name**="Doctor Scheduling System"**org**="doublechaintech" >

</root>

其中root是标签，chinese\_name, english\_name, org 是属性名称，医生排班系统，Doctor Scheduling System，doublechaintech是属性值。

| 序号 | 属性名称 | 描述 | 参考 |
| --- | --- | --- | --- |
| 1 | chinese\_name | 中文名称 | 将会显示在界面首页上 |
| 2 | english\_name | 英文名称 | 显示在英文版的界面首页上 |
| 3 | org | 开发公司的名称 | 比如doublechaintech，后台代码会显示为包名 com.doublechaintech |
|  |  |  |  |

第二部分：建一个简单的系统

我们在模型中增加了一个hospital（医院）对象，这个对象是整个系统的树根，其他的对象直接或者间接会依赖这个对象存在。

在两个root标签之间，可以输入内容。

<?xml version='1.0' ?>

<root **chinese\_name**="医生排班系统" **english\_name**="Doctor Scheduling System" **org**="doublechaintech" >

<hospital

name="上和医院|[1,200]"

address="毕升路22号"

telephone="028-9123123"

/>

</root>

hospital为标签，这里的所有标签都可以自己定义。规则是，小写加上下划线分割，比如small\_hispital。对应到数据库，会对应一张表。

name为属性名，可以自动定义，规则是，小写加上下划线分割，比如hispital\_name，"上和医院"为属性值，[1,200]标明了这个字段的最大和最小长度。如果定义将在附表中体现

最佳实践：把不重要的属性放后面，这样显示表格的时候就可能隐藏起来。

标签和属性名最好使用无拼写错误的英文，这样很容易翻译为中文。拼写错误的单词，系统会给出警告。后续章节会指示如何发现拼写错误。

有了医院以后，我们希望管理医院里医生信息。以下的例子代码医院里面可以管理很多医生，医生在医院里面工作(这里没有考虑医生在多个医院工作的情况）。

黑体部分建立了医生和医院的关联，其中$(hospital)中的hospital必须是已经定义的对象。名称可以改为其他。

有了这个文本文件，我们可以根据指令上传文件，执行相关命令，就可以生成系统了。

登录界面

登录之后可以选择自己的app

进入主界面，可以看见生成了趋势图。

点开医生列表，可以查看医生列表。通过这个界面，基本的增删改查导航都做好了

<?xml version='1.0' ?>

<root **chinese\_name**="医生排班系统" **english\_name**="Doctor Scheduling System"**org**="doublechaintech" >

<hospital

name="上和医院|[1,200]"

address="毕升路22号"

telephone="028-9123123"

/>

<doctor

name="魏松全"

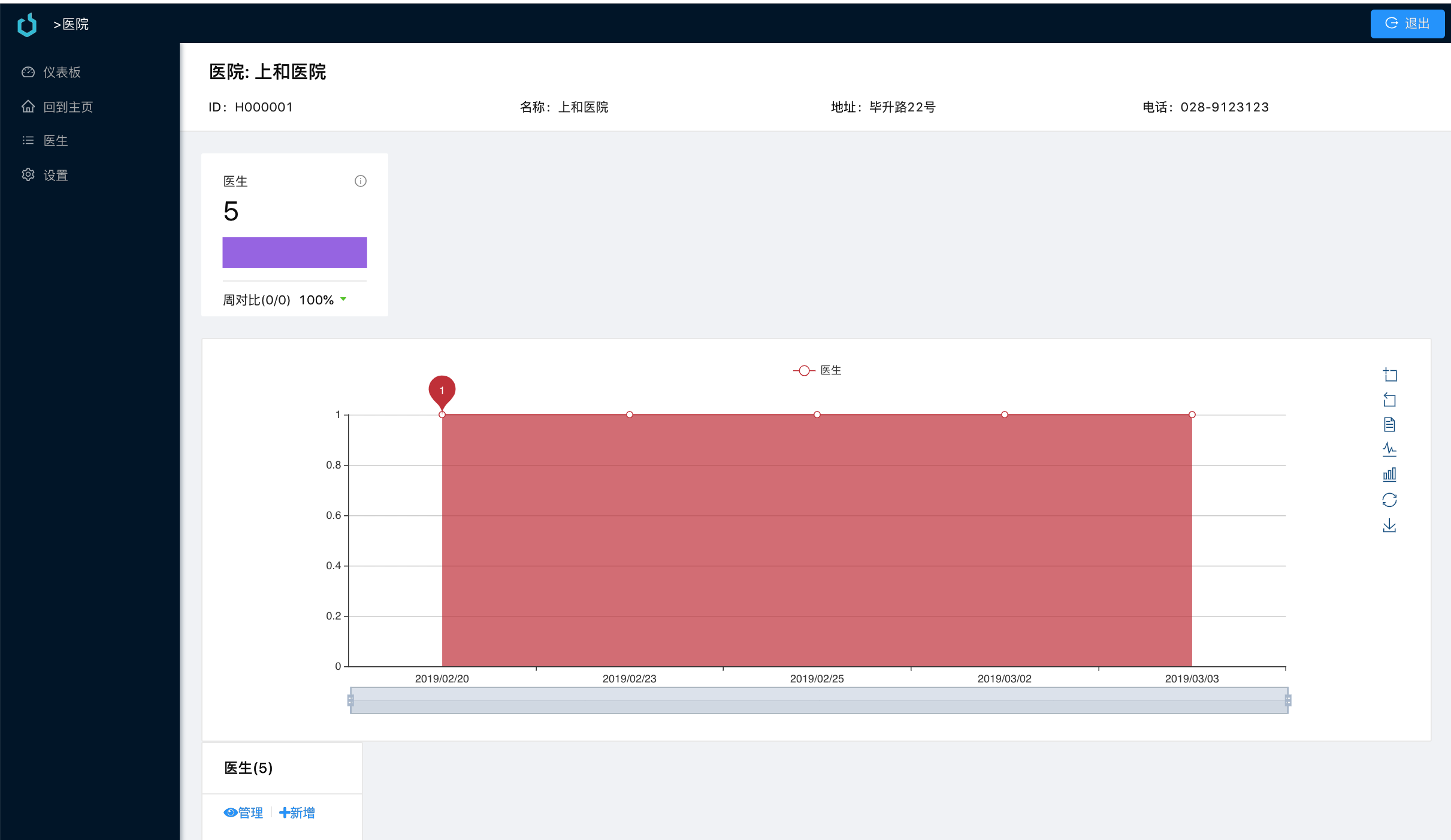
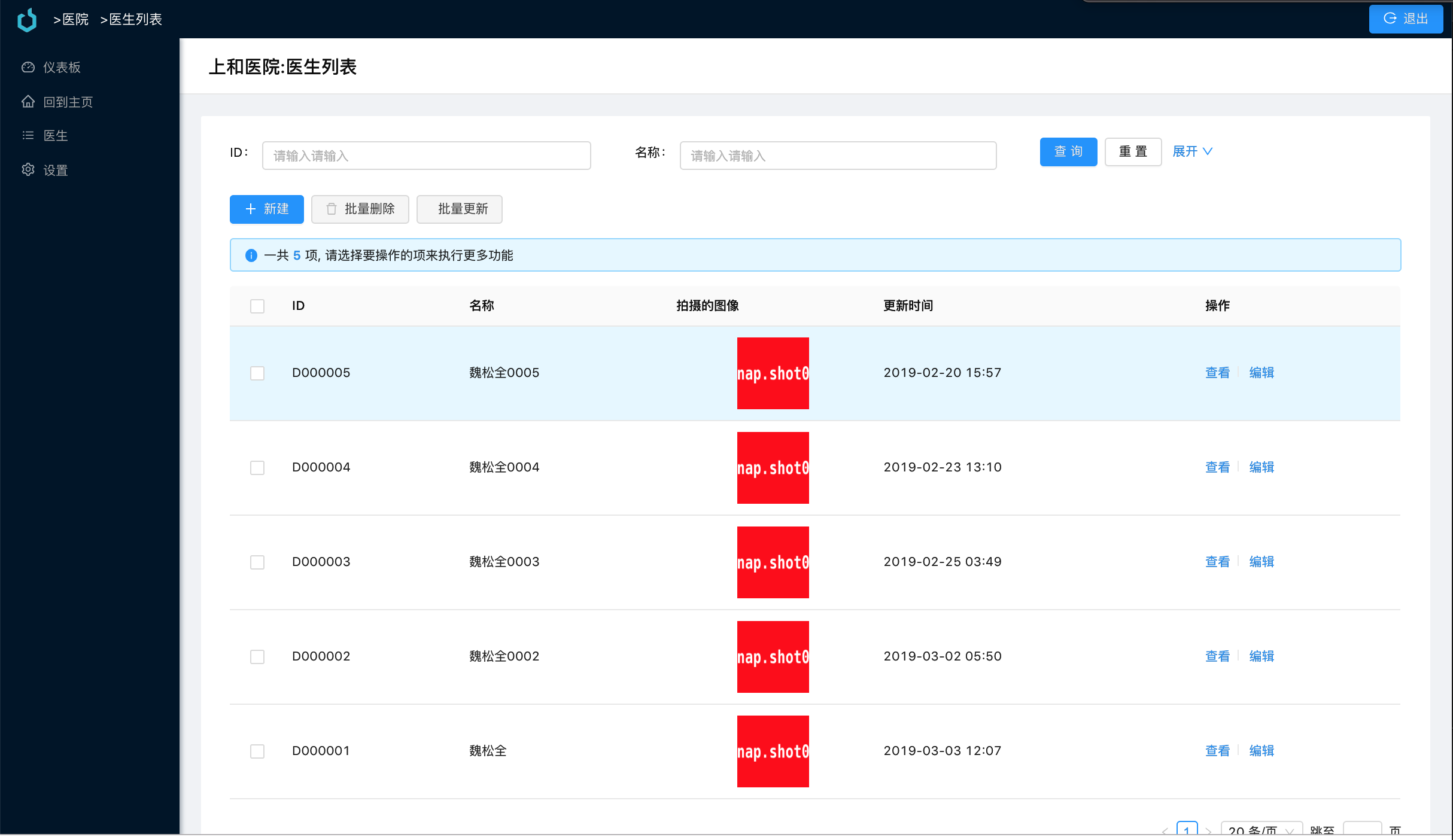
shot\_image="snap.shot-400-300-red.jpg"

**hospital="$(hospital)"**

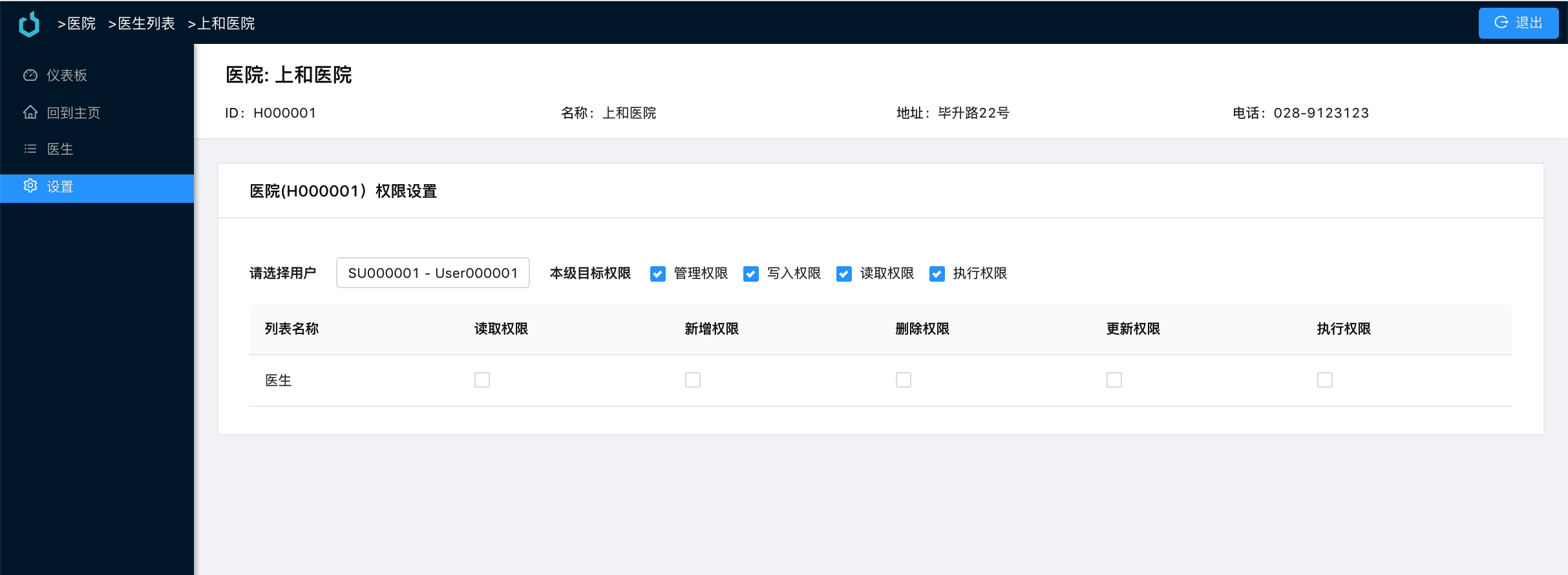
update\_time="updateTime()"

/>

</root>

每个字段按照定义顺序生成，ID是系统生成，除了文本字段以为，如果产品经理输入了 shot\_image="snap.shot-400-300-red.jpg", 系统就自动推断为是一幅图片，这样在展示的时候，系统就自动展示为图片，自动生成上传下载的代码，400-300-red标明了尺寸和底色。更多规则请参考附表。

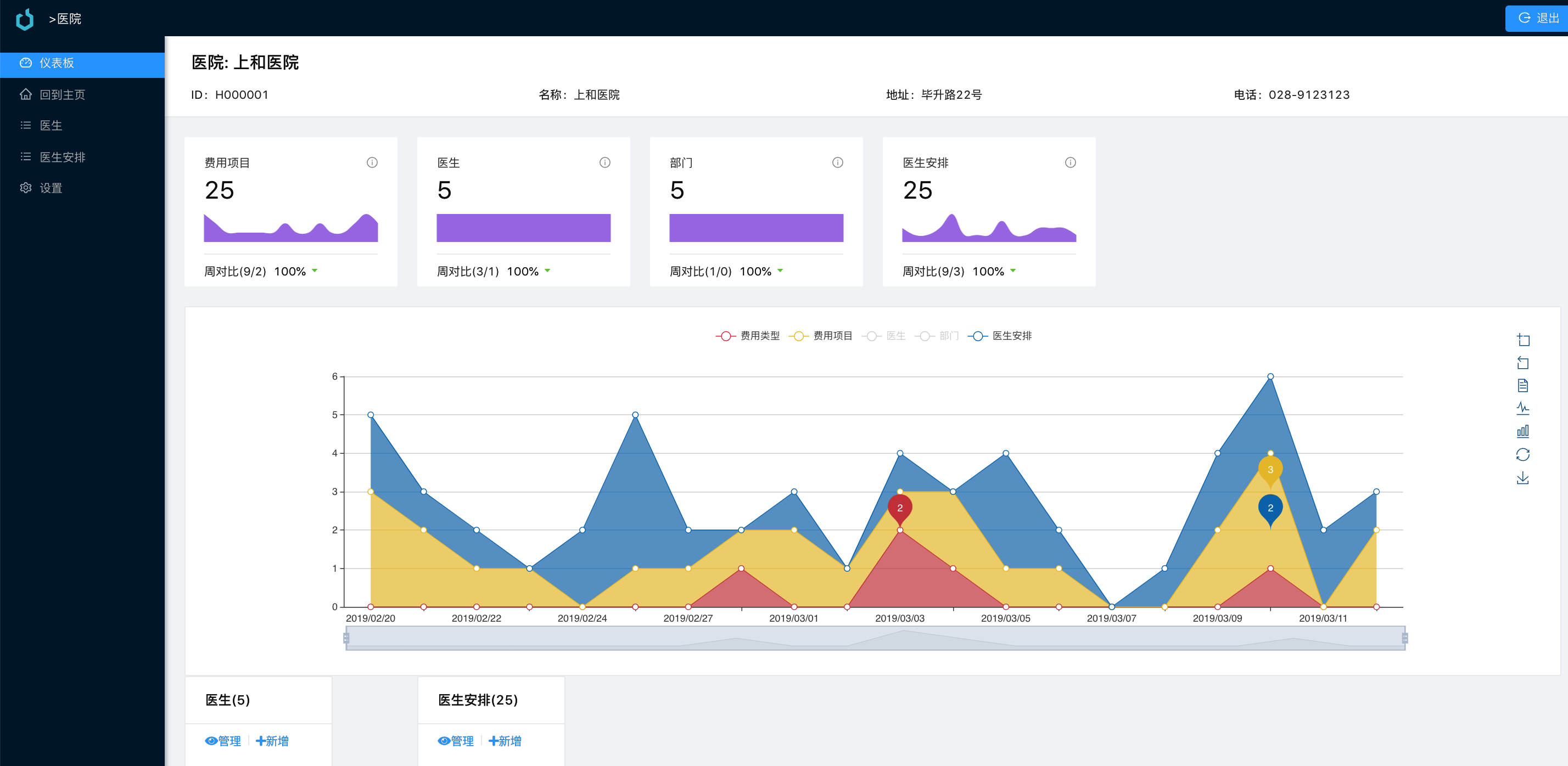
还可以设置哪些用户可以访问医生列表

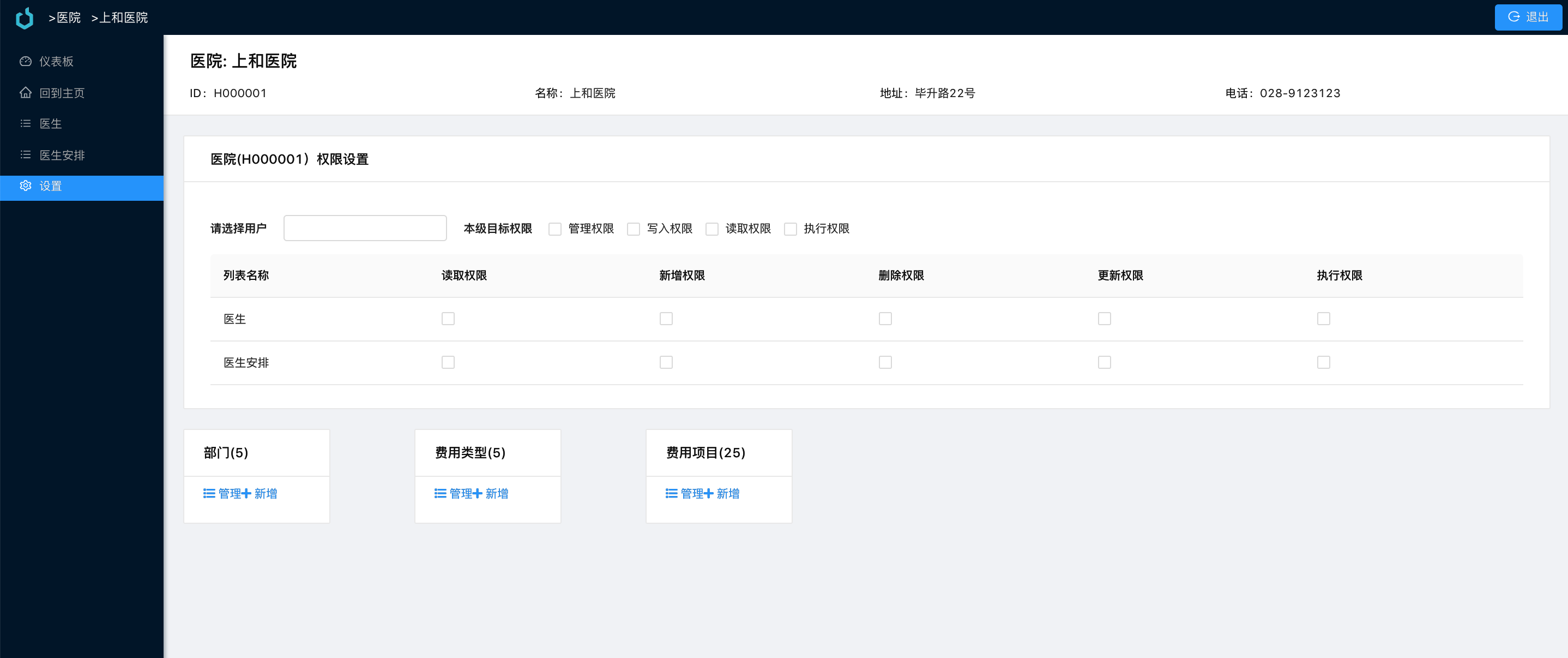


我们大约10行的文本，经过系统一系列处理，就可以生成前端，后端代码，并且配置好了演示和测试数据。

第三部分：建一个排班系统

这个例子我们将增加一些模型，这些模型增加完成以后，前端系统看起来是这样的。

系统自动生成了相关图表

权限管理增加了”医生安排“ 相应项目

我们会在后面解释为何能生成这些界面。

分析

医生排班系统的核心是这么一句话:

某个医生在某个科室在上午，下午或者晚上值班，每个排班可能有若干个号，用完为止。

<doctor\_schedule

name="2019年3月11日魏松全在内分泌科坐班收诊疗费,每个10"

doctor="$(doctor)"

schedule\_date="2019-3-11"

period="上午|下午|夜班"

department="$(department)"

available="20"

price="$123.99"

expense\_type="$(expense\_type)"

create\_time="createTime()"

update\_time="updateTime()"

hospital=“$(hospital)"

/>

记下这个事，我们需要如下的结构

里面提及了 doctor, department, expense\_type, 没有建立，我们让这些对象都存在于一个医院下。

在上述例子中，如果我们不需要，expense\_type等对象显示到主仪表盘上，我们加上了一个特性 \_features=“setting”。

<expense\_type

name="诊疗费|治疗费|检查费"

helper\_chars="zlf|zlf|jcf"

status="正常|停用"

hospital="$(hospital)"

description="text()"

update\_time="updateTime()"

\_features="setting"

/>

<expense\_item

name="专家诊疗费|血常规|煎药费"

price="$99.00"

expense\_type="$(expense\_type)"

hospital="$(hospital)"

update\_time="updateTime()"

\_features="setting"

/>

<doctor

name="魏松全"

shot\_image="snap.shot-400-300-red.jpg"

hospital="$(hospital)"

update\_time="updateTime()"

/>

<department

name="放射科"

hospital="$(hospital)"

update\_time="updateTime()"

\_features="setting"

/>

<doctor\_assignment

name="魏松全在内分泌科室上"

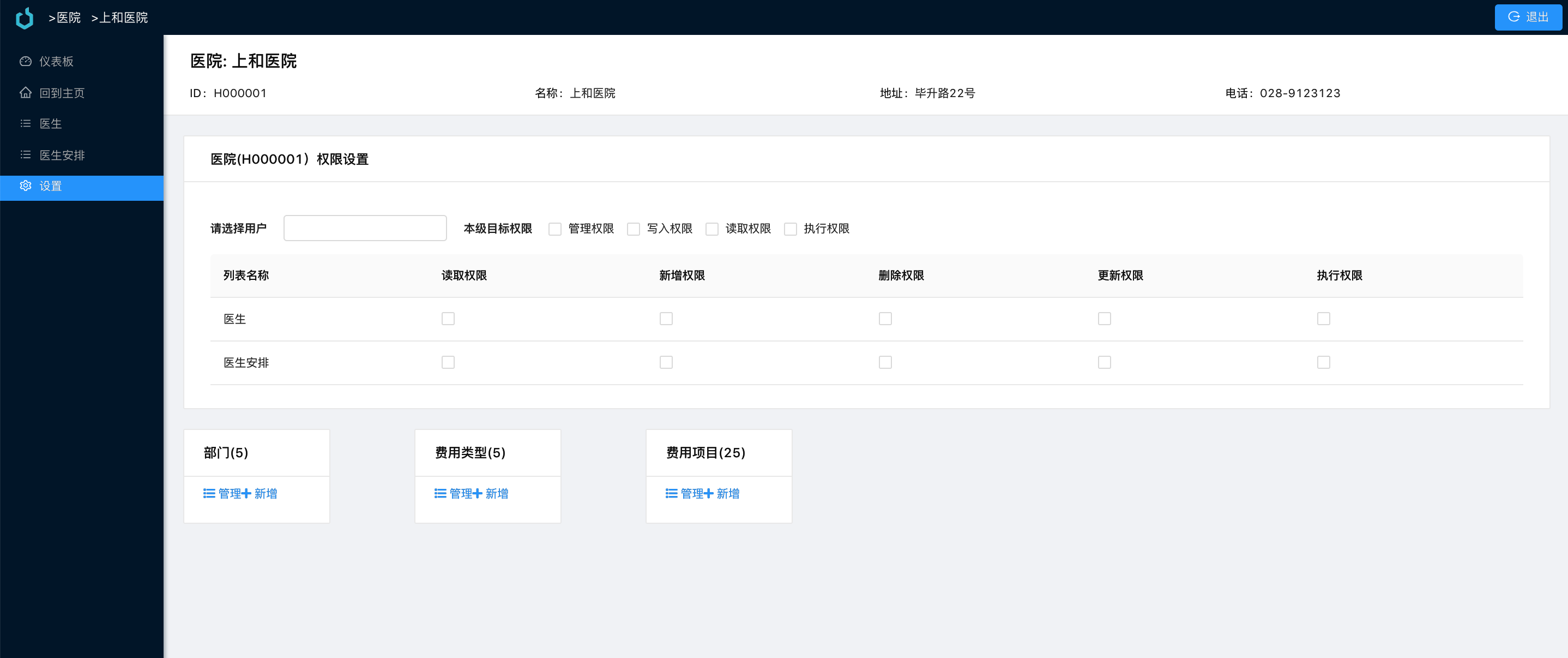
doctor="$(doctor)"

department="$(department)"

update\_time="updateTime()"

\_features="setting"

/>

加入了setting，就会出现在系统设置区域内。主面板不再出现该对象列表，请参考下图。

验证模型文件

使用如下地址来验证文件是否存在警告或者错误

<http://t420.doublechaintech.cn:2080/sky/javaweb/report.jsp?name=his>

主要常见的错误有引用的关系不存在，使用了编程语言中特定的关键字。

常用的警告主要是单词的拼写错误。

自动类型推断附表

| 例子 | 类型 | 说明 | 影响 |
| --- | --- | --- | --- |
| 简单 | 字符串 | 长度内部约束 | 约束检查 |
| 简单|[1,20] | 字符串 | 长度从1到20 | 约束检查 |
| text() | 字符串 | 长字符串，约束受WEB服务限制 | 约束检查 |
| 简单|复杂 | 字符串 | 长度内部约束 | 生成数据重复简单和复杂 |
| $(hospital) | 对象引用 | 表示这个对象”附着“在另外一个对象之上。 | 对象管理，导航 |
| 11 | 整形数字 | 表示最大值为11 |  |
| 3|11 | 整形数字 | 表示最大值为11，最小值为3 |  |
| $10.99 | 金钱类型 | 金钱类型 | 约束检查，最多为10.99元，界面会显示红色，Java内部为BigDecimal |
| 10.99 | 定点小数 | 精度为2的数字 | 会自动格式化 |
| createTime() | 日期时间类型 | 创建时自动赋值 | 无需开发人员管理 |
| updateTime() | 日期时间类型 | 更新时自动赋值 | 无需开发人员管理 |
| banner-400-300-red.jpg | 图片类型 | 表示一个长度为400X300的banner，底色为红色，也支持0x00ff00这样的16进制颜色表示 | 会相应生成上传下载代码 |
| 13988889999 | 手机号类型 | 严格验证一个手机号 | 显示的时候默认会屏蔽掉4位 139\*\*\*\*9999 |

特性表(\_features)

| 特性 | 含义 | 影响 |
| --- | --- | --- |
| setting | 设置 | 在菜单和仪表板都显示该对象，在设置中能看见 |
| log | 类似于日志 | 在界面上不提供增删改的功能 |
| status | 状态 | 在界面上不提供增删改的功能，并且数据只是生成在模型XML定义的 |
| custom | 定制组件 | 会生成前端默认的定制仪表板和表格 |
| blockchain | 区块链支持 | 自动上链，目前还不支持 |
| iot | 物联网支持 | 物联网节点，目前暂不支持 |

# 第二部分：后端定制

## 概述

SYSX因为以下三点使得二次开发变得容易，也实现了自动生成的代码和手写代码隔离

* 吸收了大型可定制产品设计思想
* 经过若干个迭代调整
* 数个实际上线的项目中调整

## 工具

后端开发使用到如下工具：

* Oracle JDK1.8或者OpenJDK1.8
* 开发IDE环境Eclipse或者Idea较新版本
* 编译环境Gradle5.1+

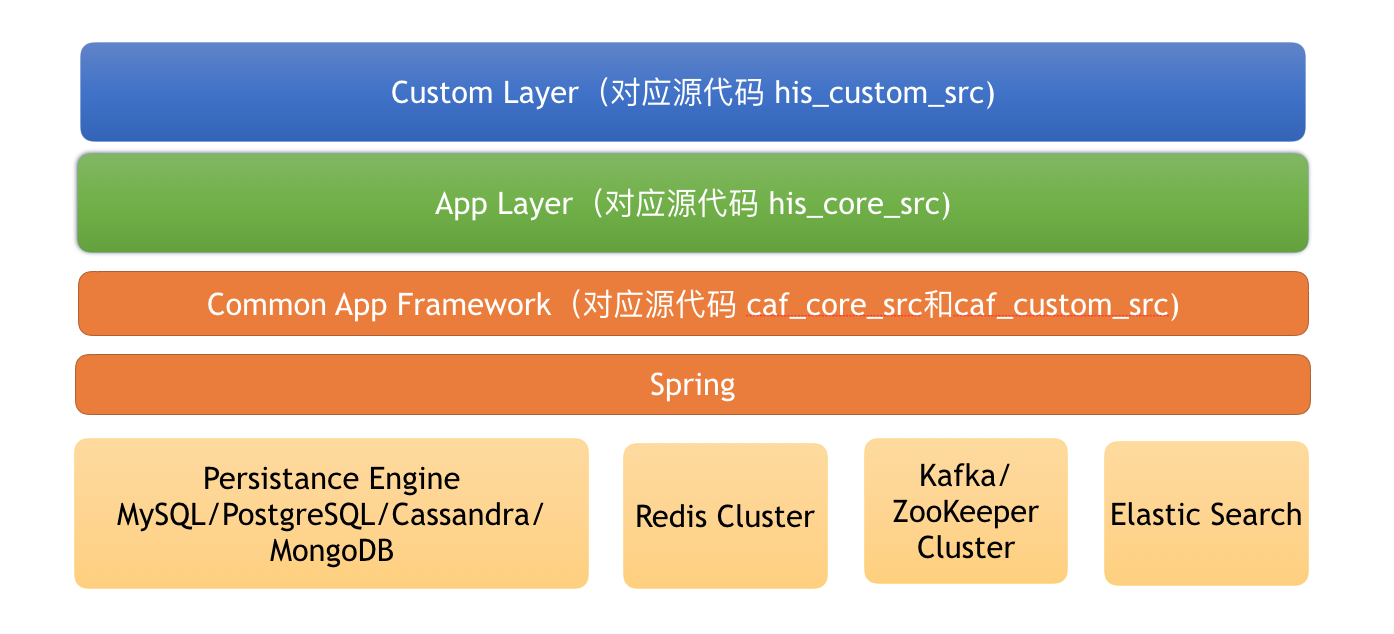
运行时环境

* 实时类装载和调试服务器Resin-3.1.16
* MySQL 5.7+
* Redis 3.2+
* Kafka, ZooKeeper, Arrangodb可选，后期运维人员即可处理

## 方法

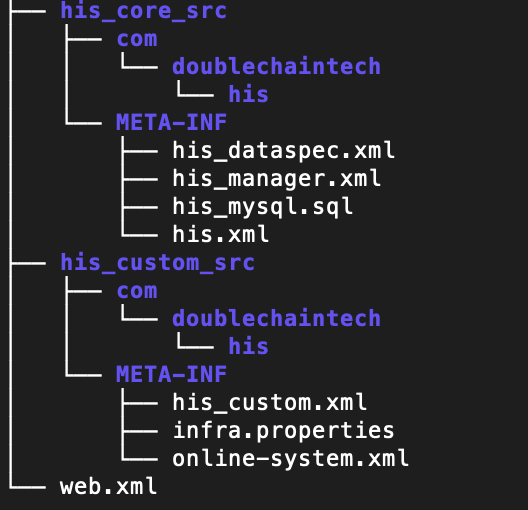
### 逻辑架构

下图就是软件系统的逻辑架构，层和层之前使用目录进行了完全的隔离。在定制的时候，core里面的代码只是作为参考，因为每次模型变更后生成代码都会被覆盖。后端代码的位置在 bizcore/WEB-INF/里面，和war目录结构保持一致的原因是开发人员更容易找到java文件编译成的class，也利于在Resin中重新装载类以免去系统启动的时间。

本章节假设读者了解如何利用重载，调用父类方法，来增加或者修改已经存在的Java对象行为。

bizcore下面有gradle文件，通常情况下运行gradle classes来编译并同步目录即可。

项目目录介绍



以下是his系统的项目介绍

这个目录显示了高层的包结构，其中

his\_core\_src中

* his\_dataspec.xml: 规格文件，在开发时候可以作为参考
* hist\_manager.xml: 所有的manager bean的定义文件
* his\_mysql: 改系统的mysql脚本和数据初始化脚本
* his.xml, 基础的DAO BEAN定义文件

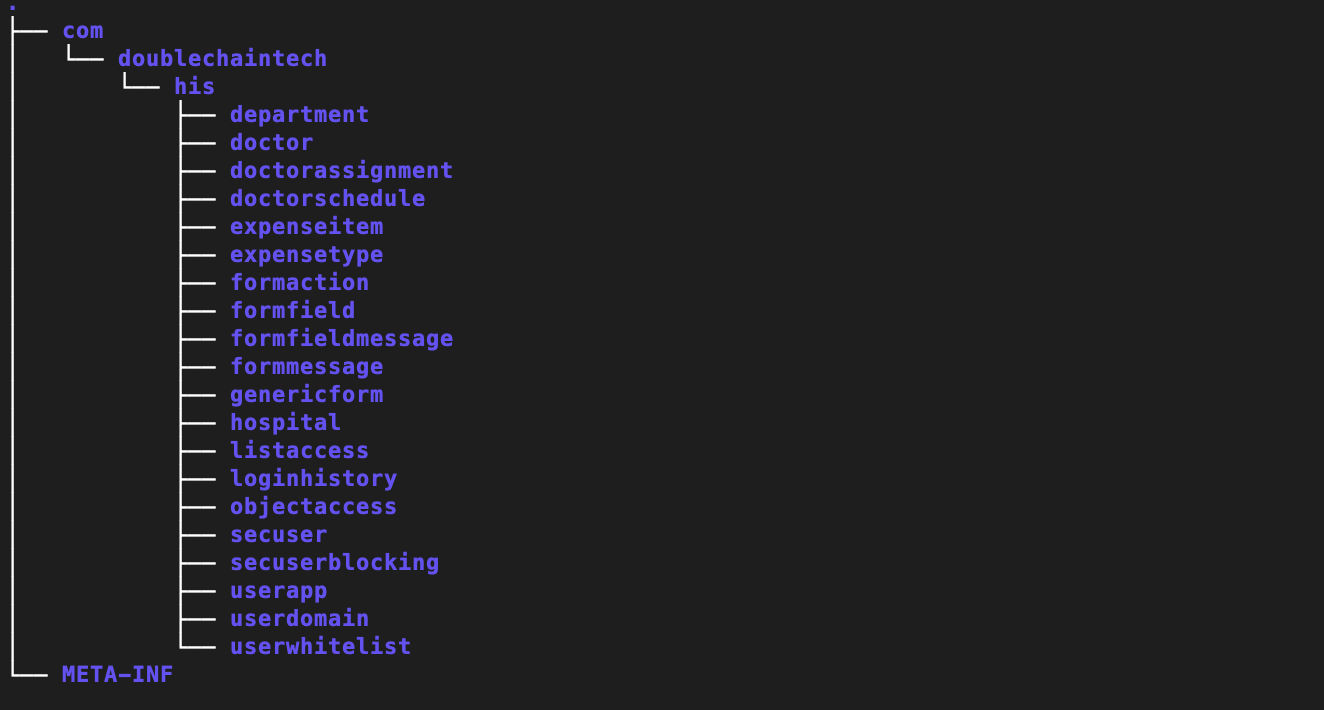
his\_custom\_src中

* his\_custom.xml: 定制化的bean定义文件，通常通过覆盖相同beanid的文件，已有的bean定义，
* infra.properties: 基础设施参数文件，里面定义了数据库连接和redis连接等参数
* online-system.xml: xml bean定义系统的入口文件，可以增加其他定义文件来扩展系统功能

注意：在his\_custom.xml中，系统自动推断某些manager具有极大可能重写，这些类无需在his\_custom.xml定义。

JAVA源代码目录介绍

这是core的源代码组织，custom按照同样目录进行源代码组织



每个对象都对应了一个源代码包，在his下面，有很多基础类，这些类大大减少了源代码的行数。

以下是源代码文件定义，后缀统一为类型，前面为对象类型，里面涉及DAO和Manager接口定义等。

| 文件 | 描述 |
| --- | --- |
| CandidateDoctor.java | 产品一个选择列表的对象 |
| DoctorDAO.java | DAO的接口定义，可以实现该接口支持新的数据类型 |
| DoctorJDBCTemplateDAO.java | JDBC Template的实现 |
| DoctorManagerException.java | 从Manager里面抛出的异常定义 |
| DoctorManagerImpl.java | 默认的Manager实现，里面包含了很多可能用到的方法 |
| DoctorManager.java | Manager接口定义 |
| DoctorMapper.java | JDBC Mapper |
| DoctorNotFoundException.java | 找不到的时候抛出的异常 |
| DoctorSerializer.java | JSON的序列化类，可以重写改方法 |
| DoctorTable.java | JDBC Template的Table实现 |
| DoctorTokens.java | 搜索，排序，分析，分页，保存令牌 |
| DoctorVersionChangedException.java | 当对象发生版本变更时候抛出的异常 |
|  |  |

开发人员需要在一般情况下已经集成已经存在的类，这样能够最大量的减少源代码的编写和调试工作。

## 过程

在一个实际项目开发中，可能按照顺序考虑如下情况

* 需要在不同Manager里面增加方法
* 定义新的POJO对象，以支持接口
* 在DAO中增加方法来支持新的查询

根据以往经验，应该尽量按照以上顺序来获取解决方案。

# 第三部分：前端定制

## 概述

SYSX因为以下三点使得二次开发变得容易，也实现了自动生成的代码和手写代码隔离

* 吸收了大型可定制产品设计思想
* 经过若干个迭代调整
* 数个实际上线的项目中调整

## 工具

前端开发使用到如下工具：

* 基础环境Nodejs
* 开发IDE环境VS Code
* 编译工具 yarn 1.13+

前端

* 在浏览器总运行
* 生成的静态文件部署到服务器上

### 逻辑架构

前端代码在源代码目录的bizui中，下图是软件系统的逻辑架构，有如下特性：

* 后端代码的位置在 bizui/src/里面
* 层和层之前使用目录进行了完全的隔离在定制的时候
* bizcomponent里面的代码只是作为参考，因为每次模型变更后生成代码都会被覆盖
* customcomp是手写代码的位置，用于注册，重写已经存在的组件或者增加新的组件



通过这个架构，开发人员无需修改生成的代码，就可以：

* 定制表格
* 定制弹出框
* 定制搜索表单
* 定制更新界面
* 定制新增界面

项目目录介绍：

以下是项目目录以及说明

如果要定制某个组件，可以在 \_features=“custom”, 则可以自动生成要定制的组件的代码，可以在其上进行修改。

生成的文件说明

| 文件 | 描述 |
| --- | --- |
| Doctor.app.js | App文件，用于组织基础的界面 |
| Doctor.associateform.js | 关联的弹出界面 |
| Doctor.base.js | 该包共享的数据 |
| Doctor.createform.js | 创建该对象用的表单 |
| Doctor.dashboard.js | 仪表板 |
| Doctor.modaltable.js | 弹出的表格，目前用于确认删除的对话框 |
| Doctor.model.js | dvajs风格的模型 |
| Doctor.preference.js | 设置界面 |
| Doctor.search.js | 搜索界面，包含表单和结果表格 |
| Doctor.searchform.js | 搜索表单 |
| Doctor.table.js | 显示结果的表格 |
| Doctor.updateform.js | 更新表单 |

每个界面元素还有一个less文件，这个文件可以直接修改用于更改生成数据的规格。

├── axios

├── bizcomponents(自动生成的代码，可以查看，但不能修改）

│   ├── department（每个概念对应一个目录）

│   ├── doctor

│   ├── doctorassignment

│   ├── doctorschedule

│   ├── expenseitem

│   ├── expensetype

│   ├── hospital

│   ├── listaccess

│   ├── loginhistory

│   ├── objectaccess

│   ├── platform

│   ├── profile

│   ├── registration

│   ├── secuser

│   ├── secuserblocking

│   ├── userapp

│   ├── userdomain

│   └── userwhitelist

├── common（基础函数）

├── components（通用组件，大部分为Ant Design Pro组件，少部分是双链提供）

├── custcomponents（定制组件的位置)

│   ├── customindex.js(注册定制组件）

│   ├── customrouter.js（定会router）

完成后的系统 <https://demo.doublechaintech.com/admin/his/>

管理员 用户名/密码：13900000001/DoubleChain!y1

医生 用户名/密码：13900000003/DoubleChain!y1