虚拟宠物医院  
软件架构设计说明书

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本： | 2.0 |
| 作 者： |  |
| 完成日期： | 2021-4-21 |

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| 0.1/草稿 | 罗亦翔 | G05 | 2021/3/15-2021/3/17 |  |
| 1.0/发布 | 罗竣夫 | G05 | 2021/3/18-2021/3/24 | 添加api，发布1.0版本 |
| 2.0/发布 | 罗竣夫 | G05 | 2021/4/18-2021/4/21 | 根据项目实际进度更新文档 |

目录

[1 概述 3](#_Toc32848848)

[2 设计目标和约束 3](#_Toc32848849)

[3 架构设计 3](#_Toc32848850)

[3.1 总体方案 3](#_Toc32848851)

[3.2 架构说明 3](#_Toc32848852)

[3.2.1 架构图及说明 3](#_Toc32848853)

[3.2.2 架构设计关键点 3](#_Toc32848854)

[3.2.3 高可用性设计 3](#_Toc32848855)

[3.2.4 高性能设计 3](#_Toc32848856)

[3.2.5 可扩展性设计 3](#_Toc32848857)

[3.2.6 安全性设计 3](#_Toc32848858)

[3.2.7 其他设计 3](#_Toc32848859)

[4 部署方案 4](#_Toc32848860)

# 概述

本文档用于描述虚拟宠物医院项目的软件设计架构，对软件的范围、功能、实现技术进行规范说明，以满足系统设计的目标和相应约束。

# 设计目标和约束

虚拟宠物医院是一个web项目。应满足项目需求说明书中描述的功能性需求，非功能性描述等。在安全性上应有可靠的用户身份验证以及权限控制机制。同时，本项目应当满足用户并发访问需求，并且在访问期间保证相应的性能及数据安全。本项目在数据保存、运营等方面应满足当地在相关方面的法律法规。

# 架构设计

## 总体方案

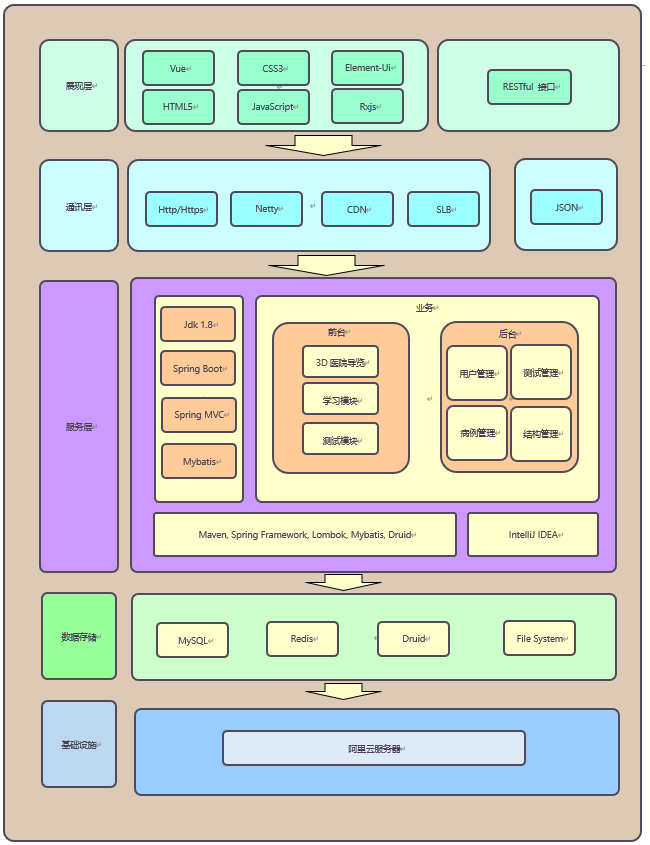


图 1总体架构图

软件开发前后端分离，并部署在云上。前端使用Vue框架，基于HTML5、CSS3、Typescript等设计前端UI界面，并与后端交互。前后端通过JSON格式数据对接。后端使用Spring Boot + Spring MVC + Mybatis框架实现功能，为前端提供RESTful接口，管理数据库连接池与数据库交互。数据库使用MySQL进行持久化存储。项目部署到云端。

## 架构说明

### 架构图及说明

整个软件的架构分为五个部分，自底向上分别是基础设施、数据存储、服务层、通讯层和展现层。

**基础设施：**主要指软件运行所需的基础设施支持，包括网络、硬盘和云服务器。云服务器用于支持项目部署。

**数据存储：**主要是使用MySQL和文件系统来支持数据存储。

**服务层：**主要包含立项、管理成员、管理项目信息、公示系统、管理设备等业务，以及支持业务所使用的技术，这些技术包括编程语言Java、编程框架Spring Boot、Spring MVC、Mybatis以及缓存支持Redis、数据连接池Druid。除此之外，为了保障客户明确的需求，还需要进行日志记录、登录认证、权限控制、数据校验、异常处理。其中日志记录起着错误检查等作用，登录认证是保证软件中各用户以各自的角色各司其职，权限控制结合登录认证保障业务正常进行；数据校验和异常处理则是为了保障软件产品的可靠性。

**通讯层：**主要包括Http/Https、Netty、CDN、SLB。项目通过Http/https请求进行通信及传输数据，使用Netty框架进行客户端和服务器的通信，项目作为微服务部署到云端，通过CDN实现静态数据加速、SLB实现服务器负载均衡。

**展现层：**主要使用到Vue、CSS3、Rxjs、HTML5等技术，以及使用到RESTful接口进行前后端通信，以JSON格式的数据进行交互。前端使用Vue框架，基于HTML5、CSS3、Typescript、Bootstrap、Rxjs设计前端UI界面，使用RESTful规范进行前后端交互。

### 架构设计关键点

前后端分离，Vue + Spring Boot + MySQL

**AB接口定义规则：**后端提供接口为Restful风格接口

**数据交换格式：**后端返回结果为一个实体类，具体定义如下：

public class ResponseMsg {

private int status;

private HashMap<String, Object> responseMap = new HashMap<>();

public int getStatus() {

return status;

}

public void setStatus(int status) {

this.status = status;

}

public HashMap<String, Object> getResponseMap() {

return responseMap;

}

}

**定义接口列表：**

**病例管理**：/case

获取全部病例信息，每页展示6条

get /admin/case

params: 无

response:

- result::StatusCode

- data

- cases::List, case.length()=6\*3=18 每个病例三个数据

- cases[0] 病例0所有文字信息

- cases[1] 病例0所有图片信息

- cases[2] 病例0所有视频信息

- ...

- totalCasesCount::Int 一共有多少病例

修改病例文字信息

put /admin/case

params: {

caseId,

caseName,

jieZhen, 文字信息

zhenDuan, 文字信息

zhiLiao 文字信息

}

response:

- result::StatusCode

新增病例

post /admin/case

params: {

caseId,

caseName,

jieZhen, 文字信息

zhenDuan, 文字信息

zhiLiao 文字信息

}

response:

- result::StatusCode

上传病例接诊/诊断/治疗图片/视频

post /admin/upload

params: {

caseId,

type = {1, 2, 3}, 当前上传的文件是属于:1接诊, 2诊断, 3治疗

file

}

response:

- result::StatusCode

**用户管理**: /user

查询全部账户信息

get /admin/user

params: 无

response:

- result::StatusCode

- data

- user[0]

- user[1]

- user[2]......

修改账户信息

post /admin/user/<userId::Int>

params: {

userId,

account,

name,

password,

type

}

response:

- result::StatusCode

添加账户

put /admin/user/

params: {

userId,

account,

name,

password,

type

}

response:

- result::StatusCode

删除账户

delete /admin/user/<userId::Int>

params: 无

response:

- result::StatusCode

**结构管理:** /structure

获取全部科室信息

get /admin/structure/section

params: 无

response:

- result::StatusCode

- data

- sections::List

- section[0]

- section[1]

- section[2]

- ...

获取全部药品信息

get /admin/structure/medicine

params: 无

response:

- result::StatusCode

- data

- medicines::List

- medicine[0]

- medicine[1]

- medicine[2]

- ...

获取全部化验项目信息

get /admin/structure/examination

params: 无

response:

- examination::StatusCode

- data

- exam::List

- exam[0]

- exam[1]

- ...

获取全部疫苗信息

get /admin/structure/vaccine

params: 无

response:

- vaccine::StatusCode

- data

- vaccine::List

- vaccine[0]

- vaccine[1]

- ...

获取全部住院信息

get /admin/structure/hospitalize

params: 无

response:

- hos::StatusCode

- data

- hos::List

- hos[0]

- hos[1]

- ...

修改科室信息

post /admin/structure/section

params: {

secId,

secName,

recDesc,

assissDesc,

docDesc,

secImageUrl

}

response:

- result::StatusCode

修改药品信息

post /admin/structure/medicine

params: {

medId,

medName,

medDesc

}

response:

- result::StatusCode

修改收费信息

post /admin/structure/fee

params: {

feeId,

feeName,

feePrice

feeDesc

}

response:

- result::StatusCode

修改化验项目信息

post /admin/structure/examination

params: {

examId,

examName,

examDesc

}

response:

- result::StatusCode

修改疫苗信息

post /admin/structure/vaccine

params: {

vacId,

vacName,

vacDesc

}

response:

- result::StatusCode

修改住院信息

post /admin/structure/hospitalize

params: {

hosId,

hosAnimalName,

disease

}

response:

- result::StatusCode

新增科室信息

post /admin/structure/section

params: {

secId,

secName,

recDesc,

assissDesc,

docDesc,

secImageUrl

}

response:

- result::StatusCode

新增药品信息

post /admin/structure/medicine

params: {

medId,

medName,

medDesc

}

response:

- result::StatusCode

新增收费信息

post /admin/structure/fee

params: {

feeId,

feeName,

feePrice

feeDesc

}

response:

- result::StatusCode

新增化验项目信息

post /admin/structure/examination

params: {

examId,

examName,

examDesc

}

response:

- result::StatusCode

新增疫苗信息

post /admin/structure/vaccine

params: {

vacId,

vacName,

vacDesc

}

response:

- result::StatusCode

新增住院信息

post /admin/structure/hospitalize

params: {

hosId,

hosAnimalName,

disease

}

response:

- result::StatusCode

**测试管理:** /test

新增题目

post /damin/test/question

params: {

questionId,

questionDescription,

score,

tag,

type

}

response:

- result::StatusCode

删除题目

delete /admin/test/questionId

response:

- result::StatusCode

修改题目

post /admin/test/questionId

params: {

questionDescription,

score,

tag,

}

response:

- result::StatusCode

查询题目

get /admin/test/question

response:

- result::StatusCode

- questions(包含question的全部信息)

- [q1, q2, q3......]

新增考卷

post /damin/test/paper

params: {

paperId,

paperName,

selectNum,

judgeNum,

qaNum,

duration

}

response:

- result::StatusCode

删除考卷

delete /admin/test/paperId

response:

- result::StatusCode

修改考卷

post /admin/test/paperId

params: {

paperName,

selectNum,

judgeNum,

qaNum,

duration

}

response:

- result::StatusCode

查询考卷

get /admin/test/paper

response:

- result::StatusCode

- papers(包含考卷的全部信息)

- [p1, p2, p3......]

新增考试

post /damin/test/exam

params: {

examId,

examName(paperName),

startTime,

duartion,

totalScore,

}

response:

- result::StatusCode

删除考试

delete /admin/test/examId

response:

- result::StatusCode

修改考试

post /admin/test/examId

params: {

examName(paperName),

startTime,

duration,

totalScore

}

response:

- result::StatusCode

查询考试

get /admin/test/exam

response:

- result::StatusCode

- exams(包含考试的全部信息)

- [e1, e2, e3......]

### 高可用性设计

部署在公有云上。

### 高性能设计

在数据库层设计合理的index，加快数据库的访问；

部署到云上，动态扩展服务器内存、网络等。

### 可扩展性设计

使用Spring Boot启动项目，基于Spring MVC框架实现服务器开发。Spring具有控制反转、依赖注入等特性，使得系统能方便地增加新功能新特性；

前后端分离及RESTful接口的设计，便于移动端（Android、IOS）的开发。

### 安全性设计

服务器端通过jwt进行用户身份识别，用户登录后返回token，在后续交互过程中验证客户端传来的token是否有效，同时在超出规定时间后设置该token失效；

数据库表中保存用户权限信息，根据用户身份及管理员操作维护权限表，在用户执行操作时，校验用户相关权限；

使用sha1将用户密码单向散列（提供全局salt和用户独有salt），使得即便数据库泄露，他人也难以还原用户密码。

### 其他设计

前后端分离，使得开发人员可以相对独立地开发同一个功能，便于分工。

# 部署方案

## 组件图



图 3组件图

## 配置图

图 4配置图