# 云计算 1607C 朱传文

# JAVA工程师

1. 项目介绍(按照实训二做的项目写)

|  |
| --- |
| 项目主要是针对个体与个体之前通过平台为媒介，促进借贷关系建立的一个标准的p2p平台。  使用Spring Cloud、Spring Boot完成微服务构建，项目分为用户前台和后台管理员两部分，前台用户可以进行开户、充值、投资、招标等功能、后台管理主要是负责对用户的审核和验证，比如用户招标，需要后台管理员对用户提交的信息进行审核，用户也可以通过完善自己的信息，提交自己的一些财产照片来增加自己的信用积分，当达到一定的分数的时候，我们会给用户开通招标权限  基本模块划分：开户模块、充值模块、提现模块、发标模块、投资模块、债转模块、放款模块、还款模块、会员模块 |

1. 功能模块介绍(按照自己得实际项目写)

|  |
| --- |
| 开户模块：首先我们会让用户去选择投资用户，还是需要招标的用户，在他们选择的时候我们在后台是会根据这个为标识符给他们展示不同的界面，在注册的时候我们的注册信息是根据银行的存管信息开户的，因为平台不涉及具体的资金流动，我们平台显示资金也是我们和银行对接显示的虚拟余额，因为根据行业规则我们需要银行进行资金存管。这一点也是行业整顿之后的一个硬性要求，所以开户的时候我们是会让用户进行勾选同意协议进行银行开户的，手机验证码部分我们是调用的阿里大于的短信API接口进行操作的，因为阿里云有完善的短信服务模块，我们只需要花少部分资金就可以完成短信验证了。后面登录的时候可能使用验证码登录，所以验证码这些东西也都是存到Redis中的，因为Redis是有这个过期时间的，也不用我们再次手动去做这个过期时间之类的东西了，后面的实名认证模块我们也是会调用阿里的身份证照片识别服务进行操作的。在开户的时候我们先对用户名也就是手机号做一个重复验证，会先去我们的库中查询是否唯一，不是的话会抛出异常，如果是唯一我们接下来会对用户进行存放标识符和对密码进行MD5加密，还会存放一些IP地址等东西，IP是通过HttpServletRequest的getHeader方法获取到登录IP，存放IP的主要目的是后面对登录IP进行判断是否是风险登录，如果是风险登录的话，我们会对此次登录的交易进行限制，添加重复认证，还有就是让用户去看到登录日志，如果有异常记录的话进行修改密码等操作。两个目的主要是为了保证用户的资金安全  这个是前台用户的注册方法，后台管理员的话刚开始是肯定没有的，所以我们想了三套解决方案，1，登录后台，可以访问一个类似init.do地址，代表系统的首次初始化页面，在这个方法中，如果查询当前系统中没有任何一个后台管理员用户，则导向到一个系统初始化页面，可以在这个页面中创建第一个用户，这个用户是一个超级管理员；如果系统判断当前数据库中已经存在一个后台管理员用户，则访问这个地址不用返回任何内容；  2，登录后台，访问的不是登录页面，而是login.do；在这个controller中，判断如果当前系统中不存在任何一个后台管理员用户，则返回一个创建第一个管理员的页面，完成超级管理员的创建；判断如果当前系统中已经存在一个后台管理员用户，则返回登录页面；  3，在后台系统启动的过程中，判断当前数据库中是否存在一个后台管理员用户，如果不存在，则启动系统的过程中就创建一个默认用户名默认密码的用户，这个用户就是超级管理员用户；如果已经存在，则不管；后来我们一致采用第三种方案  Redis完成热数据缓存，浏览量、验证码等：  首先验证码是存到Redis中，Redis有key的过期时间，这个是可以自行设置的，Redis过期删除采用的是定期删除，默认是每100ms检测一次，遇到过期的key则进行删除，这里的检测并不是顺序检测，而是随机检测。那这样会不会有漏网之鱼？显然Redis也考虑到了这一点，当我们去读/写一个已经过期的key时，会触发Redis的惰性删除策略，直接回干掉过期的key。  热数据缓存我们就是把首页的轮播图，已经满标的不可改变的数据等热数据存放到Redis中，这样可以避免造成数据库压力过大而宕机  浏览量这一块我们是先在Redis进行操作，每24小时同步到MySQL中一次。  完成发标模块：前台使用jQuery的一个uploadfile插件去提交这个他们的需要认证的信息进行，在认证身份证的时候我们会去调用阿里的身份证认证接口，在用户绑定银行卡开存管账户的时候我们去调用银行的认证接口，每一项通过之后我们会在相对应的字段标示为已通过，在他填写资料的时候工资范围等数据也都是通过调用我们提前准备好的数据字典进行选择的数据字典中也都是一些基本资料的状态（已未婚，有无车，有无房，有无子女，工资区间），这边审核主要是去做一个信用积分的东西，只有信用积分达到要求我们才会对他开通招标权限，然后对他进行发标一审，发标二审。由于很多业务模块都涉及到审核，我们为了简化系统冗余代码，所以我们就做了一个通用的审核工具，也就是一个RualAuth对象，这个对象主要就是审核状态和审核人员、时间之类的，每次提交审核会有一个RualAuth对象,一个用户可以有多个RualAuth对象，但是只能有一个通过的对象，那我们这个时候靠RualAuth去判断哪个对象没有通过的话太麻烦了，应该去设置RualAuthID去判断一个用户有没有通过审核，如果通过审核则不能继续提交审核.如果没有通过则会添加拒绝理由。 |

1. 开发中遇到的技术难点以及解决办法

|  |
| --- |
| 难点1：每笔交易需要唯一订单号  解决方法：当时使用了第三方支付平台常采用的办法就是使用雪花算法进行操作；  难点2：缓存击穿  解决方法：采用 L1 (一级缓存)和 L2(二级缓存) 缓存方式，L1 缓存失效时间短，L2 缓存失效时间长。 请求优先从 L1 缓存获取数据，如果 L1缓存未命中则加锁，只有 1 个线程获取到锁,这个线程再从数据库中读取数据并将数据再更新到到 L1 缓存和 L2 缓存中，而其他线程依旧从 L2 缓存获取数据并返回。  难点3：Redis集群采用哨兵机制部署，节点个数如何安排  解决办法：把两个Redis服务分成一组，一个服务死掉之后，另外一个就变成Master，如果两个Master全部死亡，才会选出下一组Master  难点4：缓存雪崩  解决办法：对不同类型的请求使用线程池来资源隔离，每种类型的请求互不影响，如果一种类型的请求线程资源耗尽，则对后续的该类型请求直接返回，不再调用后续资源。将一个服务拆开，对于重要的服务使用单独服务器来部署 |

1. 项目技术亮点分析

|  |
| --- |
| (1)**项目架构**：初期考虑到项目以后的用户量会增大，为了避免以后用户量增大对系统产生压力，使项目模块的工作效率下降，搭建分布式集群来降低模块之间的耦合度等效果，本项目采用了Spring Cloud+Spring Boot来构建分布式系统，使用Spring Cloud的组件Eureka服务注册中心来代替ZK注册中心，使用zuul网关来过滤请求等。  (2)**redis技术**：采用了Redis集群来部署缓存服务器，对大量的热点数据进行缓存，这样我们就减少了很多对DB的操作，使得DB的压力大大减少 (3)RabbitMQ：RabbitMQ使用一些机制来保证可靠性，如持久化、传输确认及发布确认等。多个RabbitMQ节点可以组成一个集群，也可以根据实际业务情况动态地扩展集群中节点。队列可以在集群中的机器上设置镜像，使得在部分节点出现问题的情况下队仍然可用。RabbitMQ提供了一个易用的用户界面，使得用户可以监控和管理消息、集群中的节点等。 |
|  |

1. 项目答辩中亮点简历审核评定标准

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **考核内容** | **简历样式是否规范** | **项目内容是否完整** | **项目中是否有亮点** | **技术难点解决办法是否高效** | **存在问题** |
| **项目经理 （评定）** |  |  |  |  |  |
| **就业处干事（评定）** |  |  |  |  |  |
| 简历是否合格 | | | | 合格 | 不合格 |

1. 项目答辩成绩评定标准

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **考核内容** | **亮点简历与项目描述是否匹配** | **项目流程是否清晰** | **学生讲解是否流畅** | **项目介绍是否到位** | **技术亮点考核是否合格** | **存在问题** |
| **项目经理 （评定）** |  |  |  |  |  |  |
| **就业处干事（评定）** |  |  |  |  |  |  |
| 项目答辩是否合格 | | | | 合格 | 不合格 |  |