# 云计算 1607C 樊荣

# JAVA工程师

1. 项目介绍（4个后台，1个前端，2个运维，1个测试）

开发时间：2018.7-2019.5

公司地址：朝阳区朝外大街MEN大厦

上班路线：从广渠门乘坐特12外,经过3站, 到达朝阳门南站

|  |
| --- |
| 我最近做的一个项目是赚赚金融，赚钱的赚，这个项目从去年7月份开始的，它是一个类似中介的p2p网络借贷平台，资金流转不经过平台，都是银行那边操作的，平台只是提供金融信息。对于信用贷来说，风控资料是最核心部分，这是证明一个人是否有偿还能力的根本。如果达到了借贷条件，可以申请借款，借款人发起招标，数据库生成当前申请标的信息，标的信息包括总借款金额，偿还利息，最低投标金额，标的描述信息等，发起申请之后不是立即发布到平台上，后台审核通过后，才能被所有前台用户可见可投。对于未登陆用户，只能看到此标的的介绍信息，更详细的内容需要登录之后才能查看，已登录用户可以在标的界面查看关于招标人基本信息，认证信息，风控信息，以及当前投标人的信息等，投标人投标后账户冻结投标金额，满标后投标人和招标人也不会立即形成借贷关系，后台需要经过两次审核，如果审核不通过，投标人所投金额会解冻，招标人则可以发起下一次投标，如果两次审核都通过，则扣投标人冻结金额，增加招标人账户资金，平台从中收取手续费，并为招标人生成分期还款的信息，投标人可查看自己受益，招标人可查看自己当前账户分期偿还信息，直至还清借款后才能发起下一次借款。 |

1. 功能模块介绍(按照自己得实际项目写)

|  |
| --- |
| 1. 会员管理：就是用户的账号和基本信息管理 2. 推广管理：就是管理奖金、活动这些，其实都是为了做网站的推广 3. 宣称管理：这个就是做网站的公告，帮助中心 4. 系统管理：这个是后台账号，系统日志，短信推广这些功能 5. 统计管理：就是用户、资金等等的统计   我主要负责的是后台系统中放款、借款模块。  放款：就是在借款人的标的满标后我们进行放款，这个操作涉及到了银行，我们使用他们的接口，银行来操作金额。我们需要发送一个请求给银行，请求银行返回给我们一个状态码，如果成功我们就进行其他业务操作，比如更新标，计算利息，打款给借款人等。  借款：借款分为个人借款和企业借款，个人借贷把信息交由后台进行初审，复审，然后进行开标，初审和复审业务逻辑大部分相同，都是前台传递审核结果，标的的id以及审核备注信息，后台判断此标信息是否有效，以及借款人的账户资金情况，在审核结果记录表中添加审核记录，包括标的信息，审核人信息等，在根据审核结果是否通过再执行相关业务逻辑。审核不通过，将投标用户的账户资金解冻，增加可用金额，账户流水记录添加记录，满标一审二审的逻辑都一样，不一样的的是一审通过后，标的信息变为等待二审，而二审通过后，则扣除投标人账户资金，即扣除冻结金额，然后平台从借款金额中收取手续费，为借款人生成还款对象等等。企业借贷是不需要登陆的，在选择企业借贷时，在后台初步审核后，会在线下跟该企业联系，成功后给该企业一个账户，在进行绑卡，完善资料等操作后可以进行借款。 |

1. 开发中遇到的技术难点以及解决办法

|  |
| --- |
| 难点1：  在当时遇到比较难以解决的一个问题就是我们后期测试的时候，发现当我们成功开标后，多个人同时投标，看标满之后的收益，结果发现借款人应还的利息和多位投资人应收利息不能达成一致，差了几分钱，当时也是用了很多方法，都没能解决，最后才发现是因为投资方和借款方调用算法的不同而造成的，这个问题也是告诉了我，我们在开始做项目前，一定要把各方面都考虑一下，以免后期出现问题，影响效率。  难点2：  还有就是我们在测试的时候，有个充值接口，在前端操作的时候，不知道为啥，点击一次会发起两次支付，这个请求分散在这个服务部署的不同机器上，结果扣了两次款。这个其实就是没有保证幂等性。  解决方法：  我们做了一个支付流水，在支付的时候必须插入一条支付流水，它的主键id是唯一的，在支付之前，先插入一条支付流水，主键id就已经进去了，然后写一个标识到redis里面去，设置为played，下一次重复请求过来时，先查redis里面的id的value，如果是played就说明已经支付过了，就不要重复支付。  难点3：  在开发中有哪些优化sql的方式？  解决方法：  项目中常用到的字段查询条件建立索引优化查询速度,并且优化索引、优化访问方式，限制每次返回数据的结果集的数据量,每次join时小结果集驱动大结果集。  难点4： |

1. 项目技术亮点分析

|  |
| --- |
| 亮点一、使用Spring Cloud的优势：   1. Spring Cloud Eureka是Spring Cloud Netflix微服务套件中的一部分，它基于Netflix Eureka做了二次封装。主要负责完成微服务架构中的服务治理功能。 2. 它就是一个个简单的组件组合起来config、注册中心、feign、htstrix、sleuth等等都特别简单,使用方便，条理清晰。 3. springcloud 是基于springboot开发的，所以也继承了springboot的优点，开发部署等都非常简单。 4. springcloud更新频率快,所以后期肯定会发展的越来越稳定。   亮点二、分布式事务：  **比如说我们的业务，一般都是跟**钱**相关的，跟钱打交道的，**支付**、**交易**相关的场景，我们会用 TCC，严格保证分布式事务要么全部成功，要么全部自动回滚，严格保证资金的正确性，保证在资金上不会出现问题。**  **而且最好是你的各个业务执行的时间都比较短。**  **但是说实话，自己手写回滚逻辑，或者是补偿逻辑，实在太恶心了，那个业务代码是很难维护的。**  **TCC 的全称是：Try、Confirm、Cancel。**   * **Try 阶段：这个阶段说的是对各个服务的资源做检测以及对资源进行**锁定或者预留**。** * **Confirm 阶段：这个阶段说的是在各个服务中**执行实际的操作**。** * **Cancel 阶段：如果任何一个服务的业务方法执行出错，那么这里就需要**进行补偿**，就是执行已经执行成功的业务逻辑的回滚操作。（把那些执行成功的回滚）**   亮点三、使用Spring Data JPA的优势：  1、提供统一的接口，可避免我们再次重复编写基础的DAO类；  Spring Data JPA提供了几个基础的接口类，分别为CrudRepository、PagingAndSortingRepository、 Repository，其中，CrudRepository提供了基础的增删查改方法，PagingAndSortingRepository则在CrudRepository的基础上提供了对数据查询的分页支持，Repository建立在PagingAndSortingRepository的基础上，提供了完善的接口方法。  2、遵循JPA规范，同时也提供了灵活的数据访问方式；Spring Data JPA能够很好的兼容目前JPA2.0规范，在定义Entity类的时候，可以在类中定义标准的@NamedQuery查询，也可以在Spring Data JPA的Repository类的方法中定义@Query查询。   1. 通过方法名即可自动生成HQL语句； 2. 通过接口自动注入实现类，实现非常简单。  我们只需要定义一个类似DAO的接口类，并继承Spring Data JPA提供的Repository接口 类， 一切就大功告成了，不需要编写任何一行代码，你的DAO接口类就具备了增删查改、分页、记录数统计等功能。 |
|  |

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 |  |