# 云计算 1607C 白树浩

# JAVA工程师

1. 项目介绍(按照实训二做的项目写)

|  |
| --- |
| 面试官您好，我叫白树浩，现在已经从事java接近 4年，我毕业之后经家里人介紹，我去了**郑州信源有限公司,** 在那工作了两年多，然后离职来到了北京，在**华杰科技有限公司**工作到现在，我先后做了4个项目，有房产管理系统和聚融贷p2p金融项目，便民社交网项目，最近刚做完了一个掌庭网上金融项目。  我最近做的项目是一个金融项目，我先大体的介绍一下这个项目，他是一个个人对个人的项目，是由借款人通过平台发布借款，然后投标人通过平台进入借款页面，查看借款人相关的借款信息，比如说借款金额，年利率，借款期限，还款方式，以及借款进度，然后投资人选择投标，如果金额达到用户借款金额就会进入满标审核，由平台管理人员进行后台审核，如果审核符合要求，管理人员进行操作审核通过并且向银行发送请求放款， |

1. 功能模块介绍(按照自己得实际项目写)

|  |
| --- |
| **开户模块**（流程）：开户分三种情况 1.在我的账户设置填写开户信息，2.在借款人借款的时候会判断是否开户，3.投资人在投标的时候判断是否开户，如果没有进行开户，就会跳进开户页面，开户主要填写用户真实姓名，身份证号，预留的手机号，选择银行和填写银行卡号，交易密码，开户协议选钩， 填写完成以后进行提交，然后系统发送给银行，进行审核，审核完成以后，银行返回系统，系统在显示给用户结果，是否成功  **借款模块**：借款分为三种借款方式，  第一种是信用贷款我们要满足四个条件  信用贷款**:** 1.填写基本资料，2.身份认证，3. 风控资料分数达到可借款的分数，4.视频认证，  房贷款**:** 1.填写基本资料，2.身份认证，3. 风控资料分数达到可借款的分数，4.提交房屋抵押相关信息,5.视频认证，  车贷款**:** 1.填写基本资料，2.身份认证，3. 风控资料分数达到可借款的分数，4.提交车辆抵押相关信息,5.视频认证，  满足以上条件认证,就可以填写贷款金额,选择还款日期,以及还款方式,进行提交申请贷款  **我要投标**：投标人呢先进入我要投标页面,在投标页面可以看到借款信息,比如借款人姓名,借款金额、年利率、还款方式借款进度等信息,  投标**要求**：投标人不可以给自己投标,投标人只能对同一个标,投标5次,对每一个标最多不能超过总标的%20,平台也是为了让更多的投标人获取收益，  在满足这些条件后就可以输入金额和交易密码，提交这次投标，然后生成一个投标对象保存到数据库当中，并且生成这次投标的一条投标流水，对该次投标做一次投标记录，接着修改借款相关信息。之后在判断该标是否已经投满，如果已经投满，我们将该标的借款状态改为满标审核状态，等待后台管理员的满一审  **提现模块：**用户发起提现申请后，后端管理员不对其进行审批，直接传送到第三方支付，第三方支付也不进行审批，传送给银行进行审批，然后以银行反馈结果为准，未得到第三方反馈结果时，状态为提现处理中，反馈结果成功后为提现成功（前台显示为已提现）反馈结果失败时为提现失败，提示请联系管理员 |

1. 开发中遇到的技术难点以及解决办法

|  |
| --- |
| 难点1：redis到缓存击穿，缓存雪崩，过期策略以及内存淘汰机制。  解决方法2：比如redis只能存5G数据，可是你写了10G，那会删除5G的数据，怎么删的，还有就是数据已经设置了过期时间，但是时间到了，内存的占用率还是很高，有考虑过原因吗？  Redis采用的是定期删除+惰性删除策略  定时删除，用一个定时器来负责监视key，过期则自动删除。虽然内存及时释放，但是十分消耗CPU资源，在大并发请求下，CPU要将时间应用在处理请求，而不是删除key，因此没有采用这一策略  那定期删除+惰性删除策略是怎么工作的呢  定期删除redis默认每个100ms检查，是否有过期的key,有过期key则删除。需要说明的是，redis不是每个100ms将所有的key检查一次，而是随机抽取进行检查(如果每隔100ms,全部key进行检查，redis岂不是卡死)。因此，如果只采用定期删除策略，会导致很多key到时间没有删除。  于是惰性删除派上用场。也就是说在你获取某个key的时候，redis会检查一下，这个key如果设置了过期时间那么是否过期了？如果过期了此时就会删除。此时也会产生其他的问题,就有了内存淘汰机制。  在redis.conf中配置一行数据maxmemoey-policy volatile-lru/  推荐使用allkeys-lru：当内存不足容纳新写入的数据，在键空间中，移除最近最少使用的key。推荐使用。  就到了缓存穿透和缓存雪崩的问题  在中小型企业中很少一件这类的问题，缓存击穿解决方案（一）利用互斥锁，缓存失效时候，先去获得锁，得到了锁，再去请求数据库。没得锁，则休眠一段时间重试。  （二）采用异步更新策略，无论key是否取到值，都直接返回。Value值中维护一个缓存失效，缓存如果过期，异步起一个线程去读取数据库，更新缓存。需要做缓存预热。  缓存雪崩  给缓存一个失效时间，加上一个随机值，避免集体失效。  使用互斥锁，但是该方案吞吐量明显有所下降。  双缓存。我们两个缓存，缓存A和缓存B。缓存A的失效时间为20分钟缓存B不设置失效时间。  自己做缓存预热操作。然后细分一下几个小点  1从缓存A中读取数据库，有则直接返回。  2A没有数据，直接从B读取数据，直接返回，并且异步启动一个更新线程。  3更新线程同时更新缓存A和缓存B。  难点3：登录状态的问题（jwt） 。  解决方法3：分布式web应用的普及，通过session管理用户登录状态成本越来越高，因此慢慢发展成为token的方式做登录身份验证，然后通过token去取redis中的缓存的用户信息，随着之后的jwt的出现，校验方式更加简单便捷化，无需通过redis缓存，而是直接根据token取出保存的用户信息，以及对token可用性校验，单点登录更为简单。  使用JWT核心代码。引入依赖  会有一个id 主题 签发者 签发时间 签名的算法以及秘钥  Jwt工具类和token验证  难点2：借款模块需要对分布式事务进行解决  解决方法1：借款时就相当于是投资人向借款人转账的一个情况，但是仔细分析里面就有很多的业务，首先一个标肯定是有多个投资人共同投资，那么此时有一个投资人想一个标投资了500元那么在他的充值余额中的五百元就锁定，不能操作，或者是直接先将金额-500元。当次标满标而且每层审核都通过后，此时就进行实时的操作，投资人的账户减钱，借款人的账户加钱，就用到了分布式项目中TCC方案。如果中间出现了审 核不通过的情况就相当于是事务失败回滚进行对投资人的钱进行原样返回。  难点3：安全登录：我们使用Spring Security实现用户的安全登录。编写配置文件spring-security里面。我记得有个http的标签，可以对静态资源放行，并设置登陆成功的跳转页面，默认的登录页面。当用户发送登录请求时，通过配置文件。我们自定义一个认证类，实现UserDetailsService接口。重写loadUserByUserName方法，然后创建角色列表，将在配置文件里面配置的角色放到列表里面，根据用户名去数据库查询是否存在该用户，如果存在将用户的密码与输入的密码进行比对，返回userDetails。这样就完成了安全登录。  JWT token验证：  为了保证接口的安全性，我们使用JWT实现跨域访问，当用户登录成功后，后台生成一个token，就是一个String字符串，这个token包含一些信息，比如当前用户的唯一标识id，用户的手机号，Token的签发时间以及token的过期时间。过期时间我们设置为7200秒。  将这个登陆成功状态信息和token一起返回给前台。前台获取到这个token之后，每次请求接口时，都在请求头携带者token。我们在后台通过拦截器拦截到前台的请求，解析token，如果解析成功，将token重新放到request里面，当调用后台接口是，会首先判断request里面有没有token，如果token存在，可以接做接下来的操作，如果不存在，则向前台响应对应的状态码以及错误提示。  难点4：登录状态的问题（jwt） 。  解决方法3：分布式web应用的普及，通过session管理用户登录状态成本越来越高，因此慢慢发展成为token的方式做登录身份验证，然后通过token去取redis中的缓存的用户信息，随着之后的jwt的出现，校验方式更加简单便捷化，无需通过redis缓存，而是直接根据token取出保存的用户信息，以及对token可用性校验，单点登录更为简单。  使用JWT核心代码。引入依赖  会有一个id 主题 签发者 签发时间 签名的算法以及秘钥  Jwt工具类和token验证 |

1. 项目技术亮点分析

|  |
| --- |
| (1) 项目中我使用了RabiitMQ来实现:解耦、异步、削峰  解耦:如果使用 MQ，A 系统产生一条数据，发送到 MQ 里面去，  哪个系统需要数据自己去 MQ 里面消费。如果新系统需要数据，直接从 MQ 里  消费即可；如果某个系统不需要这条数据了，就取消对 MQ 消息的消费即可。这  样下来，A 系统压根儿不需要去考虑要给谁发送数据，不需要维护这个代码，也  不需要考虑人家是否调用成功、失败超时等情况。  异步:  一般互联网类的企业，对于用户直接的操作，一般要求是每个请求都必须在 200 ms 以内完成，对用户几乎是无感知的。  如果**使用 MQ**，那么 A 系统连续发送 3 条消息到 MQ 队列中，假如耗时 5ms，A 系统从接受一个请求到返回响应给用户，总时长是 3 + 5 = 8ms，对于用户而言，其实感觉上就是点个按钮，8ms 以后就直接返回了，爽！网站做得真好，真快！  削锋:  每天 0:00 到 12:00，A 系统风平浪静，每秒并发请求数量就 50 个。结果每次一到 12:00 ~ 13:00 ，每秒并发请求数量突然会暴增到 5k+ 条。但是系统是直接基于 MySQL 的，大量的请求涌入 MySQL，每秒钟对 MySQL 执行约 5k 条 SQL。  一般的 MySQL，扛到每秒 2k 个请求就差不多了，如果每秒请求到 5k 的话，可能就直接把 MySQL 给打死了，导致系统崩溃，用户也就没法再使用系统了。  但是高峰期一过，到了下午的时候，就成了低峰期，可能也就 1w 的用户同时在网站上操作，每秒中的请求数量可能也就 50 个请求，对整个系统几乎没有任何的压力。  如果使用 MQ，每秒 5k 个请求写入 MQ，A 系统每秒钟最多处理 2k 个请求，因  为 MySQL 每秒钟最多处理 2k 个。A 系统从 MQ 中慢慢拉取请求，每秒钟就拉取  2k 个请求，不要超过自己每秒能处理的最大请求数量就 ok，这样下来，哪怕是  高峰期的时候，A 系统也绝对不会挂掉。而 MQ 每秒钟 5k 个请求进来，就 2k 个  请求出去，结果就导致在中午高峰期（1 个小时），可能有几十万甚至几百万的  请求积压在 MQ 中。  这个短暂的高峰期积压是 ok 的，因为高峰期过了之后，每秒钟就 50 个请求进 MQ，但是 A 系统依然会按照每秒 2k 个请求的速度在处理。所以说，只要高峰期一过，A 系统就会快速将积压的消息给解决掉。 |
|  |

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 |  |