# 云计算 1607C 王龙

# JAVA工程师

1. 项目介绍(按照实训二做的项目写)

|  |
| --- |
| 我当时主要做的项目是互联网交友项目,如果你是一位技术大牛，你可以在上面发表一些技术言论,在这里言论自由,可以与技术小白加好友关注,然后进行问题的探讨。如果不想错过各种技术交流会，还可以关注一些自己喜欢的频道然后进行预约。可以留言自己的问题，技术名人会为我们一一留言进行解答。如果你有太多的苦恼可以进行多种吐槽，让你放飞自我。项目的主打模块有基础模块、用户模块、吐槽模块、交友模块、问答模块、搜索模块、登录模块。项目主要使用的是前后台分离模式开发的，利用前后端之间的协议API进行开发的,利用Result接口规范进行前后的接口对接,前端所用到的技术是Vue.js,后端采用的是微服务架构SpringCloud,分布式架构SpringBoot.项目中主要的功能模块我们将利用虚拟化技术Docker,在VMware客户端里面操作Docker命令将功能模块制作成镜像,将镜像制作成容器,利用SpringCloud进行微服务之间的通信。而在微服务通信时我们利用了服务治理Eureka处理服务的注册与发现。当然在项目中假如我们想要调用某一个微服务里面的方法时,我们首先就要把微服务注册到Eureka上，这样就实现了服务之间的方法调用了。以上就是我的项目介绍。 |

1. 功能模块介绍(按照自己得实际项目写)

|  |
| --- |
| 在项目中我主要负责的模块有用户，吐槽和交友，参与了登录和搜索的开发，那么就由我来介绍我说负责和参与的模块。  用户模块：用户模块有用户登录和用户注册两个模块。  在登录页面时,有是否去登录或者是否去注册,如果去注册的话我们输入手机号进行短信验证,对于短信验证这一块我们利用的是阿里大于短信接口进行验证的,当前台点击验证码时后台随机生成一个6位数(我们利apache.commons.lang3)将随机数利用redis进行存储，另外发送到消息中间件rabbitMQ中，通过监听类进行监听，然后调用阿里大于短信接口实现短信发送 之后将验证码填入与之前存储的进行比对，这样就可以实现用户注册了。  用户登录模块分为普通用户和管理员用户。而对于普通用户时只需要完成注册用户名和密码就ok了,对于管理员用户登录时我们是不可注册的,是有一个超级管理用户员登录后添加用户。当普通用户想进行吐槽和交友操作时都必须先通过前后台的验证，在此时我们还是要判断当前用户的角色身份，当在客户端使用用户名和密码请求登录时,服务端收到请求时去验证用户名和密码,验证成功后服务端生成一个Token,我们把这个token发给客户端,客户端收到Token以后进行存储.在每次进行微服务调用的时候需要带上Token然后进行验证.如果验证通过的话,就可以进行好友互相关注，发表自己的吐槽等内容。  吐槽模块的话，首先这些都是登录之后的操作了，否则是没有权限的，对于吐槽这一模块来说我们首先选择的是MongoDB数据库，因为吐槽的信息数据量，写作频繁，而且我们吐槽一些问题和人的话数据丢失几条也没有什么影响的，正好MongoDB正好符合这一特点,还有吐槽对于表结构设计时是有特点的,我们在最后设计一个parentid 与主键\_id设置主外键关系,不是真正的主外键而是为了有树形结构这种体系。我们都知道在吐槽时是可以接着你吐槽的东西继续吐槽的，当然在相互点赞的同时我们也是利用了MongoDB里面的一个自增列,使用 $inc 运算符，对某列进行增加或减少。当然在吐槽时我们如果聊得比较嗨的话是可以添加关注成为好友的，接下来我们可以聊一下交友模块。  对于交友模块的话,当用户登陆后在推荐好友列表中点击“心”，表示喜欢此人 ，在数据库tb\_friend表中插入一条数据，islike 为0,当你点击了喜欢过的人，也喜欢了你,表示互粉成功！也向tb\_friend表中插入一条数据，islike为1并且将你喜欢她的数据islike也修改为1，当你点击了不喜欢某人（点击了叉），向tb\_nofriend添加记录.当两个人互粉后，其中一人不喜欢对方了，删除好友表中的记录 ，向非好友表中添加记录。我们在用户模块里面写了一个更新好友粉丝和用户关注的方法时,我们让交友模块进行调用,利用的是SpringCloud里面的一个服务调用的组件Feign,通过在交友模块写了一个接口,利用@FeignClient进行方法的调用。  搜索模块：  我们的搜索模块使用的是Elasticsearch分布式搜索引擎,IK分词器和Logstash数据库同步。Elasticsearch是一个实时的分布式搜索引擎，可以快速的去处理大规模数据，可以实时搜索，提高搜索效率。但是对于搜索的关键词我们才用了IK分词器，因为IK分词器能够把搜索的关键词进行细分,这样让我获取我们所要得到的信息更加准确更加详细。对于Logstash来说是一款轻量级的日志搜集处理框架，可以方便的把分散的、多样化的日志搜集起来，并进行自定义的处理，将数据库的信息输入（配置到自定义的文件中）以json格式进行输出(配置Elasticsearch的一些信息)。 |

1. 开发中遇到的技术难点以及解决办法

|  |
| --- |
| 首先项目主要用到的是微服务架构和分布式架构,项目中就会遇到有关分布式场景的一些问题,例如分布式事务,分布式锁,跨域等问题.分布式事务产生的原因是在多个系统中数据库垂直分割,数据库之间不能跨数据库进行访问,所以每个数据库的事务,都是在自己数据库独立进行管理的.这样就需要解决分布式事务的解决方案.而解决分布式事务的原理则是利用了Java的JTA,而java的JTA则是java操作XA接口,XA相当于协调者提供资源管理器与事务管理器之间进行通信的标准接口。而他们遵循2pc,3pc协议而我们绝大部分采用柔性事务都遵循CAP和Base理论,可以暂时不一致,但是最终要保证数据的一致性而我们公司最开始的探讨研究过分布式的解决方案第一种利用的是java Automatic框架,最开始感觉java Automatic有一定的缺点他遵循的是两段提交协议,他可能出现延时,当别人没有进行通知时可能进行卡死而我们公司发现Automatic用于多数据的情况下,用于传统的项目,而我们公司采用的是分布式项目所以不适合采用.在当时我们有采用另外一种方案,采用MQ解决分布式事务,但是由于我们的项目利用的是Springcloud,用MQ会比较麻烦所以最后又采用新的方案支付宝回调方式,就是类似于支付方式去做分布式事务管理但是又要自己写补偿机制和重拾机制还有要解决一些幂等性问题还是比较麻烦的,最后在github上找到一个新的框架tcc,lcn框架,发现lcn框架写的比较好,然后我们自己研究了一番,发现lcn底层原理的最大核心我不管理事务,而只是事务的搬用工lcn的原理是发起方调用接口,去xml创建事务分组,将事务分组的id传入,参与方接收到id时,操作一个假的关闭,不提交事务,当发起方代码执行完成以后,通知协调者,然后转发给参与方,看事务回滚还是提交.这是我们分布式事务的一个解决方案.  我们公司遇到的第二个难题就是分布式锁,什么是分布式锁,我们当时在项目当中遇到一种场景,需要做一个全局的id,但是全局id需要一个时间戳,如果在集群的情况下,时间戳因为服务器,可能在时间戳产生相同的情况下,当时想了好多的办法去解决全局id,之后利用分布式锁去解决.分布式锁相当于在多个服务器之间进行通信时,能够保证JVM性能的安全性问题,分布式锁跨Jvm,分布式锁在多个jvm进行通信时,能够保证多个JVM只有一个进行执行.在当时也有两种解决方式一个是利用reids,但是redis的有效期是比较麻烦的,有可能产生死锁,另外一种通过zookeeper的节点机制,首先zookeeper通过临时节点,当别人可以将临时节点创建成功时,就可以拿到我们这个锁了,因为节点是不允许重复的,当别人创建节点时,是不可创建的,需要去等待,等锁释放完成以后才可以拿到锁,那么zookeeper里面是怎样管理锁的释放了,zk有一个比较好的概念,就是有一个临时节点的特征.临时节点相当于session会话一关闭的情况下,整个连接将会断开,断开以后节点清空.所以锁就会被释放。  在前后端分离的情况下难免会出现跨域问题,在当时前端通过Ajax发请求时访问不到,跨域问题产生于浏览器方面,不是在服务端,而在当时我们就在服务器端解决跨域问题,同时也考虑过客户端解决,但是客户端解决有一些缺陷,通过jsonp,jsonp本身原理是通过标签进行转发,有这样一个缺陷,这个缺陷在当时我们解决方式是因为它只支持get请求,不支持post请求,所以我们当时就搭建网关的方式,网关方式相当于全部请求到网关,然后进行拦截,将部分分发到具体服务器上,从而保证域名相同,或者我们可以使用httpclient请求进行转发,但是这个比较占内存,或者使用设置请求头进行跨域这是我在项目中遇到的问题,分布式事务,分布式锁,和跨域问题,这几个问题的遇到让我有很大的提升。 |

1. 项目技术亮点分析

|  |
| --- |
| (1)项目框架：项目采用得spring boot+springcloud微服务框架，好处是对于springboot默认依赖web组件SpringMvc,简化了xml配置,方便整合一些我们项目中所用到的框架,注解开发让我们的开发效率更加迅速。而SpringCloud在平时开过程发中，对于一些相关服务的业务逻辑实现，我们可以利用SpringCloud中的Eureka,将服务注册的eureka下，然后服务消费端完成服务调用即可。同时，springcloud它提供了可视化的监控平台，在方面大大方便了后期项目的维护。  (2)Docker技术：项目中我们是利用虚拟化技术Docker进行开发的,对于Docker技术,我们项目是将项目中用到的一些软件MySQL,Elasticsearch,MongoDB等从Docker的远程仓库中拉取镜像,把我们项目中的一些功能模块也是制作成镜像,然后将镜像制作成容器,最后就变成一个一个的微服务。Docker相当于一个大容器，Docker的镜像相当于class，容器可以当做镜像的对象。一个镜像可以放到不同的容器进行实例化，一个镜像（相当于class文件）只有一份，里面可以存放多个容器。  (3)MongoDB技术:我们主要用MongoDB来存储我们项目里面的操作日志和吐槽日志记录的。对于操作日志我们结合Aop来使用,首先配置一个AOP的切面类,再给Aop的使用规则,当哪个类里面的哪个方法使用当前的切面类,利用后置通知类获取当前方法来操作日志,将操作日志存储到MongoDB,然后进行分类管理查看。利用后置通知传到数据库。评论这块的话我们做了一个数表结构，就是用户可以回复用户的评论（吐槽）。主要字段有（吐槽的\_id,parentid,内容,评论人,评论发布时间）为了提高可用性和高并发性用了3台服务器做了mongodb的副本集，其中一台作为主节点，另外两台作为副本节点，这样在任何一台mongodb服务器宕机时就会自动进行故障转移，不会影响应用程序对mongodb的操作，为了减轻主节点的读写压力过大的问题，我还对mongodb副本集做了读写分离，使写操作在主节点进行，读取操作在副本节点进行。  (4)分库分表:分库分表按照一定的规则，对原有的数据库和表进行拆分，把原本存储于一个库的数据分块存储到多个库上，把原本存储于一个表的数据分块存储到多个表上。项目中为什么需要分库分表​微服务较多,业务较多,库和表中的数据会越来越大，而一台服务的资源终究是有限的，因此需要对数据库和表进行拆分，从而更好的提供数据服务。分库分表的方式垂直分库/分表​垂直划分数据库是根据业务进行划分，例如用户表可根据业务分成基本信息表和详细信息表等。  (5)SpringCloud技术:首先，我们这个交友项目开发前期，就做过技术选型这块的探讨，因为我们这个项目是分布式的，考虑到后期的维护与拓展，我们技术经理主推springcloud技术，因为现在开发主流框架都是spring,而springcloud在springboot基础上能做到与spring无缝集成，这一块对于我们开发人员来说，大大降低了学习和开发成本。使用Springcloud关键在于配置，在项目中，我们使用Eureka、Feign、Zuul、Hytrix等组件技术，其中eureka作为服务注册与发现中心，这个配置起来感觉比zookeeper方便很多，只需要在springboot中添加eureka相关依赖，并且在main方法中添加@EnableEurekaServer注解就可以了。服务提供端呢，只需在main启动类中使用@EnableEurekaClient注解，并在application.yml中配置好注册中心地址等信息，即可完成服务的注册。服务消费端我们选择Feign来实现客户端的负载均衡，其实基于客户端负载均衡的实现形式还有另外一种是ribbon+RestTemplate，本质上说Feign就是对Ribbon的封装，在使用Feign时我们使用@EnableEurekaDiscovery和@EnableFeignCleints注解来注明该服务为服务消费端，同时Feign可基于接口+注解实现服务rpc调用，例如我们在交友微服务接口添加一个@FeignClient(value = "xxx")的注解，即可远程调用用户的接口服务。  (6)reids技术:我们在项目中用redis来做缓存，在登录的时候将用户登录时获取的验证码存放到了redis中，并设置了验证码的有效期为5分钟，当用户登录时我们从redis中获取出用户的点击时获取的验证码，如果获取不到直接返回前台，并返回相应的错误码，否者取出redis中的验证码和前台传过来的验证码进行比较。我们还用reids缓存点赞数量,控制不能重复点赞。首先是处于登录状态下,利用已经存储的点赞数量进行判断,如果点赞过的话,提示您已经点赞过了,如果没有的话就更改数据库点赞的利用注入的reidsTemplate里面的opsForValue().set()方法进行点赞数量的变化。  (7)断路器的使用:在微服务架构中，存在着那么多的服务单元，若一个单元出现故障，就会因依赖关系形成故障蔓延，最终导致整个系统的瘫痪，这样的架构相比较传统架构就更加的不稳定。为了解决这样的问题，因此产生了断路器模式。在项目中我们在问答模块进行了断路器的配置,当基础服务进行调用时如果有一方发生故障的话,会立马跳到熔断器这边,然后给我这边返回一个信息。  (8)JWT技术:在当时我们项目中用Jwt实现Tonken验证机制的,jwt包括头部、载荷与签名,头部描述的是关于该JWT的最基本的信息，类似于一个json对象,在头部指明了签名算法是HS256算法,我们进行BASE64编码得到jwt的第一部分,载荷就是存放有效信息的地方,利用自己的一些规则进行第二部分的加密(过期时间必须要大于签发时间),第三步是通过前两部分进行加盐加密实现的中,将这三部分组成则就生成一个大的字符串这就是我们的Token。在进行测试时我们利用Postman工具进行测试,在测试的时候我们按照之前前后端的约定有一个头信息Authorization然后在加上自己设定的简单密钥进行测试,这样就保证我们后台的安全了。 |
|  |

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 |  |