**项目总结报告**

日期：2020年1月5日

|  |  |  |  |
| --- | --- | --- | --- |
| 组号 | 3 | 项目名称 | 小箱交云作业平台 |
| 编程语言 | Golang, JavaScript | 开发平台和框架 | VS Code, IDEA  Go-micro, Gin, Gorm, React |

|  |  |
| --- | --- |
| **软件需求特性** | |
| **1.是否实现了项目立项时的所有需求？列出实现的新增需求和未实现的需求。**  没有实现立项时的所有需求，但是我们将未实现的需求转换为了新增需求进行实现。  新增需求：  1）教师在开课时可以查看所有不在该课程的学生的列表，从中选取学生加入课程  未实现的需求有：  1）教师为所设课程批量从excel文件中导入学生信息。我们之所以没有实现该需求是因为我们认为该需求本身存在问题，我们无法保证教师传入的学生一定在系统中存在，同时我们也不知道excel文件的具体格式，所以我们将该需求转换为了上文提到的新增需求。  **2.采用哪种架构风格？哪些设计模式？**  总体上本项目采用了B/S架构风格。  服务端采用了微服务架构。  本项目采用的设计模式有构建器模式、原型模式、单例模式、装饰器模式、外观模式、享元模式、代理模式、职责链模式、中介器模式、观察者模式、访问者模式。  使用构建器模式的场景有在微服务之间通信的时候对于protobuf结构体的组装。  使用原型模式的场景有使用JWT签发token对用户进行授权。  使用单例模式的场景是应用网关来进行反向代理。  使用装饰器模式的场景是使用go-plugins配置个性化的微服务。  使用外观模式的场景是使用网关进行反向代理。  使用享元模式的场景是对于微服务中对于客户结构体的复用。  使用代理模式的场景是使用网关对于后端微服务进行反向代理。  使用职责链模式的场景是使用Hystrix对后端微服务调用链中出现的错误进行降级处理。  使用中介器模式的场景是使用Nats消息队列对服务调用者和服务被调用者进行解耦。  使用观察者模式的场景是使用消息订阅和消息发布机制对服务调用者和服务被调用者进行解耦，消息订阅方是事件驱动的，当其观察到某一特定消息被发送，它会执行特定的函数来进行处理。  使用访问者模式的场景是对于同一接口，用户以学生身份访问和用户以教师身份访问会获得不同的响应。  **3.技术方案有哪些亮点？**  技术方案的亮点在于前端使用了Canvas Draw并且集成了图像清晰度检测与优化；后端使用了微服务框架进行开发。  前端使用Canvas Draw来让教师在批改作业的时候可以直接在学生提交的作业上手写批改，最终会将教师批改的信息以点集形式存储。当再次获取批改信息的时候，前端会根据保存的点集信息重新渲染出教师的批改结果。  同时前端集成了一个使用python开发的图像清晰度优化服务，当用户要上传图像时，前端会先引导用户将图像上传到图像清晰度优化服务，该服务会检查图像的清晰度，并且优化图像的清晰度，并将优化结果返回给前端用户，用户再将经过优化的图像上传给后端存储。  后端用到了负载均衡、消息队列、断路器和链路跟踪这些技术。  首先是负载均衡。在后端微服务之间互相调用的时候或者是API服务调用后端微服务的时候，会根据被调用服务的地址采用随机负载均衡的方法选择一个服务发送请求，从而实现请求在被调用服务之间的负载均衡。  然后是消息队列。消息队列实现了调用者和被调用者之间的解耦。被调用的服务是事件驱动的，它并不需要调用者的显式调用。服务发布消息并不会影响到它正常业务逻辑的执行，就算订阅该消息的服务在执行时出错也不会影响到发布消息的服务的正常执行。  其次是断路器。断路器是微服务体系中的保险丝，避免小的故障通过连锁反应造成整个系统瘫痪甚至损坏。在Go-micro框架中可以使用插件库Go-plugins来为微服务配置断路器，我们可以在断路器中自定义当前微服务调用别的微服务的默认超时时间以及当前微服务的最大并发数。同时可以通过断路器设置当前微服务调用别的微服务的降级函数，用来自定义当当前微服务调用别的微服务的函数失败时的处理动作，从而避免因为某些故障造成连锁反应导致微服务集群崩溃。  最后是链路跟踪。后端使用Jaeger工具来进行链路跟踪。Jaeger是一个Go语言开发的微服务链路追踪工具，用于分布式系统监控和故障排除，它的主要特性包括:1) 分布式上下文传播2) 分布式事务监控3) 故障根本原因分析4) 服务依赖关系分析5) 性能/延迟优化。  **4.是否做了单元测试？是否做了系统功能测试？是否做了性能测试？是否做了兼容性等其他非功能测试？**  本项目进行了单元测试、系统功能测试、性能测试、兼容性测试。  单元测试使用的工具是Go test和Go convey。  系统功能测试使用的工具是Cypress，使用Cypress进行了端到端测试从而测试系统的所有功能。  使用JMeter进行了性能测试。  我们使用Edge浏览器、Firefox浏览器、Chrome浏览器、Safari浏览器和手机内置浏览器访问前端，从而测试前端在不同主流浏览器上的显示兼容性。 | |
| **项目组成员对项目的贡献度（%）** | |
| 程可：20%  邓诗羿：20%  蓝浩宁：20%  刘慎恒：20%  张澳：20% | |
| **软件度量** | |
| 软件代码行数（不包括注解行、空行和复用代码）： | 前端11327行 + 后端11839行 = 23166行 |
| 复用他人代码行数： | 前端215行 + 后端190行 = 405行 |
| 类的个数： | 前端157 + 后端349 = 506 |

|  |
| --- |
| **经验、教训和建议** |
| 1. **如何面对新技术？**  我们在本项目中第一次使用了Go语言的Go-micro框架进行开发，然而这个框架对于我们来说太过于陌生了，网上也几乎找不到相关的教程。在经过我们的不断摸索与试错之后，后端开发才得以顺利进行。  2. **如何进行软件过程管理？**  虽然我们在开发过程中尽力遵守软件开发过程规范，但是仍然会有因为进度紧张而不遵守规范的情况，而且后果往往是需要付出更多的时间来修复因为没有遵守规范开发而导致的错误。  3. **如何应对进度风险？**  本项目的开发进度相对来说比较紧张，我们的项目也险些无法按期完成。经过这次经历之后我们知道了按照软件需求优先级依次开发软件功能的重要性。 |