个人信息

姓名: 吴东磊

籍贯: 江苏

电话: 18862859383

邮箱: 18862859383@163.com

求职意向: Java 后端开发

教育背景

2022.09-至今（大三） 南京工业大学　计算机科学与技术

四六级已过，多次获得校综合奖学金，GPA=3.42/4.0

专业技能

- 具备的扎实的编程基础，熟悉掌握 Java，熟悉 JVM，熟悉若依框架，Gmall 商城框架，有 web 开发经历
- 熟悉 SpringBoot、Spring MVC、Mybatis 后端框架
- 在实习过程中熟悉并使用 MySQL、Redis、RocketMQ、ES 还有流程引擎
- 实习过程中了解了微服务架构，nacos
- 熟悉使用 Git 进行项目管理，能够有效地协助团队进行代码开发和协作。
- 能利用浏览器的开发者工具来调试一些问题
- 有丰富的联调经验

实习经历

南京六朝云智科技有限公司(阿里外包)

Java 后端开发实习生

2024 年 11 月 21 日-2025 年 4 月 3 日

技术栈：后端采用 Springboot 框架；微服务架构，服务间采用 dubbo 接口调用。基于阿里云 Gmall 商城标品开发；I18n 国际化；nacos；流程引擎；中间件（MySQL,Mybatis-plus，Redis，RocketMQ，Kafka，ES，调度任务 - XXL-JOB - 分布式任务调度平台，ELK 追踪

日志，MinIO，k8s 部署，MongoDB）；项目中还应用了多种设计模式。

项目背景：

项目为哈萨克斯坦中央银行的电商项目 - 平台型 B2B2C，有 C 端线上商城，B 端运营管理平台及商家后台，提供线上交易服务，支持商家入驻；项目采用微服务架构，服务间采用 dubbo 接口调用。

项目是基于阿里云 Gmall 商城标品开发，包含了 28 个子项目，比如说有交易模块、营销模块、支付模块、聊天模块（gim）、会员模块、商品模块、中间件模块、业务引擎（流程引擎等）模块等等。项目中的核心模块都是由两部分组成：center 和 platform，以交易模块为例，分为 trade-center 和 trade-platform 两个子项目。platform 提供了领域模型扩展，流程扩展，业务定制；center 引用了 platform，主要是完成一些定制业务逻辑；一般核心业务都是写在 platform 项目中，在部署时会把 platform 项目打成 jar 包，并依赖到 center 项目中。项目通过 RocketMQ 实现服务解耦与异步通信，基于 ES 进行商品搜索。
项目中还应用了多种设计模式，其中流程引擎应用了工厂模式；业务代码中也应用了门户模式（将订单、支付、售后等等模块的接口放在一起便于查找）；项目代码中也用到了构建器模式用于分布构建复杂对象。

主要职责：

在实习过程中我熟悉并使用了 MySQL、RocketMQ、ES、nacos、流程引擎等等技术。我负责的主要是交易那块的业务，刚开始的话，以改 bug 为主，在改 bug 过程中熟悉项目代码。比如说创建订单接口，在项目中涉及到了多个服务，当调用创建订单接口时，首先 gmall-frontmanager 服务会对请求参数进行一系列的校验封装，然后通过 dubbo 远程调用 trade-platform 服务，在 trade-platform 服务中首先会对传过来的对象进行转化，然后调用流程引擎来处理该业务。以创建订单该业务为例，在流程引擎中会经过：基本入参检查、业务流程这两个状态，在业务流程状态里面有很多 node 即业务处理逻辑，举几个例子：基础校验、拆包、收件人查询检查、商品查询检查、价格计算、库存、保存等等业务逻辑。通过流程引擎可以实现代码的解耦，如果业务逻辑有调整，可以很方便增加或减少相应的 node。

在实习过程中我也熟悉并使用了 RocketMQ，在使用 RocketMQ 的过程中我也遇到了一些问题，比如说 RocketMQ 消费状态显示为：NOT_CONSUME_YET（可视化界面上显示生产者成功发送了消息，但是消费者没有消费）。后来排查，猜测原因可能是有其它的应用错用了消费组名，RocketMQ 将消息意外投递到该应用中去了，后来将消费组名更改后，消费者就成功消费了。然后的话就是这次实习期间我对使用消息队列目的的理解：一个是为了业务解耦还有一个就是为了应对高并发场景。以创建订单为例，如果我没有使用消息队列，业务代码需要同步调用积分发放、分账处理等多个下游服务，任何一个下游服务异常都会导致主流程中断，并且在高并发场景下同步调用可能会引发系统雪崩。而如果我使用了消息队列，

那么当订单完成后，我会将与该订单相关的数据（比如说 primaryOrderId，orderId，primaryOrderStatus 这些数据）封装成消息发送到指定的 topic 中去，RocketMQ 采用长轮询机制主动推送消息到消费者，而非被动的"扫描"机制。消费者订阅了该 topic，并持续监听其中的消息，一旦有新消息到达，消费者便会自动接收，并调用相关的方法（会将消息载体以参数的形式传入进去）。使用消息队列会让订单服务不再感知下游业务的具体实现，只需关注自身业务逻辑；同时消息队列也可以进行异步削峰，突发流量被消息队列缓冲，将大量的消息直接放到 MQ 中，然后系统去按自己最大的消费能力去消费这些消息，避免下游服务被冲垮。

在实习过程中，我也熟悉并使用了 ES，在使用 ES 过程中我也遇到了一些问题，第一个问题是：在原索引中添加了一个新字段 seller_name，新添加的字段映射不会应用到旧文档中，只会对映射更新后新写入的文档生效。当使用 exists 查询时，ES 会检查字段是否在索引中定义且非空值。由于旧文档的 seller_name 字段未被新映射重新索引，它们会被视为“不存在”。这时候需要通过 _update_by_query 强制重新索引所有旧文档，使新映射生效。第二个问题是：ES 与数据库不同步的问题，在实习期间，经常遇到测试或开发人员直接在数据库中添加数据导致线上环境出现 bug 的问题。

在实习过程中，我还参与了一个小场景的算法优化 - 如何分配最少的城市来运送所需的商？简单来说就是买家在商城下单了很多商品，这些商品分属在不同的城市，现在有一个业务需求：需要设计一个算法使得分配最少的城市来运送买家购买的商品。

实习项目的技术架构 - 应用分层架构：首先对于读请求和写请求，通过 DTO 将请求的参数进行转换校验；然后进入业务层，这边负责 center 核心业务处理 还有 数据服务组装、模型转换、日志、异常流处理等；业务层也可以调用 repository，repository 提供对 DAO 数据库访问的封装，Adaptor（Manager）提供对其他中心 RPC 接口的封装和适配、或 rest 接口的调用，也可以封装调用链路中独立的业务模块。

当然在实习过程中，我也学到了一些非常实用的小技巧，比如说看日志：在将本地代码发布到线上环境后，时常会出现线上服务挂掉的情况，这时最好的办法就是使用终端模拟器连接线上服务器查看日志，找到报错原因；比如说对集合元素进行流操作处理，可以很方便的将流中的元素进行转换比如 .map() 操作可以将原本存储 ItemSkuId 的列表，转换为存储 ItemSkuId 中的 cartId（ItemSkuId 的一个属性）的列表；比如说在 ELK 的 Dev Tools 控制台页面编写一些 ES 查询语句查看某索引库下的文档或者该索引的映射。

南京达牧网络科技有限公司(小厂)

Java 后端开发实习生

2024 年 7 月 1 日 - 2024 年 8 月 30 日

技术栈：SpringBoot、MySQL、若依框架

主要职责：

1. 参与完成南京交通建管集团数智建养管理平台中智能决策报告模块的工作，项目中需要用到道路的车辙、平整度等数据，我在若依框架的基础上开发和维护了多个接口，确保从数据库中提取出所需要的数据。

2. 由于甲方需要，将公司很早之前用 .NET 开发的项目转化为用 Java 语言编写；我在充分理解原项目代码的基础上，基于若依框架进行项目的开发，对原项目的每一个功能，在理解了其代码逻辑后，将每一个功能通过三层架构封装为接口；项目完成后，对比发现 Java 要比用 .NET 开发的项目快的多。

3. 参与完成南京交通局“最美交通人”投票系统项目的开发，该项目的功能主要由三部分组成：所有候选人信息的展示、投票功能的实现、具体候选人信息的展示；基于若依框架将三个功能封装为接口；为了预防爬虫对投票系统的干扰，我在设计接口时，通过获取并接收客户端的微信号，并将其存储在数据库中，规定每一天同一个微信号只能投十票，否则投票无效；同时，我还深入理解分析了若依框架验证码功能的源码，在充分理解了若依框架验证码的逻辑后，我在起始页添加了一个验证码功能，防止刷票。

项目经历

https://github.com/wdlwdlwdlwdlwdl/competition

项目一：ImmunoFlash--基于 Transformer 的免疫大语言模型

大二上学期，跟着研究生导师做了一个智能医疗类的项目（ImmunoFlash--基于 Transformer 的免疫大语言模型），并为项目搭建了一个网站。

项目功能：基于 SpringBoot 框架快速搭建和部署了一个网站，该网站实现了两个功能：

1.智能预测 TCR-抗原亲合度：利用大语言模型，根据用户输入的 TCR 序列和抗原序列来智

能预测 TCR-抗原结合的亲合度，并将结果以邮箱的形式发送给用户。

2.TCR 和抗原对接可视化：用户上传一个 PDB 格式的 TCR 序列文件和 PDB 格式的抗原序列文件，调用 Docking Station，将 TCR 与抗原的复杂相互作用转化为可视化的清晰图像，帮助用户深入理解免疫学中的关键互动。

实现过程：1.基于 Spring 提供的 MultipartFile 接口处理文件上传功能，客户端可以上传需要预测的 TCR 序列和抗原序列，网页 ajax 将预测序列传到后端，后端代码通过使用 Java 标准库中的 Runtime 类来执行外部命令，运行模型端的 python 文件进行预测，预测结果将以表格形式返回至网页。用户可以点击链接进行下载，也可以选择基于 Spring 框架提供的 JavaMailSender 邮件发送功能将预测结果发送至自己的邮箱中。

2. .基于 Spring 提供的 MultipartFile 接口处理文件上传功能，客户端可以上传两个 PDB 格式的 TCR 序列和抗原序列文件，借助网站所集成的 HDOCK[20]分子对接模型算法，将 TCR 与抗原的复杂相互作用转化为可视化的清晰图像。用户可以点击链接进行下载，也可以选择基于 Spring 框架提供的 JavaMailSender 邮件发送功能将 gif 图像发送至自己的邮箱中。

竞赛经历

大一：
校二等奖学金 - 2023.4.20
江苏省第二十届高等数学竞赛本科 A 组三等奖 - 2023.7
第十四届蓝桥杯全国软件和信息技术专业人才大赛江苏赛区 C/C++程序设计大赛二等奖 - 2023.4.23
大二：第十七届中国大学生计算机设计大赛三等奖 - 2024.5

证明材料

作品集链接：
https://github.com/wdlwdlwdlwdlwdl/shi-xi-project.git

实习证明材料链接：
https://github.com/wdlwdlwdlwdlwdl/shi-xi-zheng-ming.git

竞赛证明材料链接：
https://github.com/wdlwdlwdlwdlwdl/jing-sai-zheng-ming.git