电子商城设计文档

技术框架

后端为Spring-cloud，前端为Vue2

后端dao层使用的是JPA，数据库使用的是MySQL。

微服务架构

技术选型

一共设计了6个微服务，服务之间的互相调用是通过**feign**实现的，将会被调用的服务注册到**feign-api**中。

网关使用的是**gateway**，服务注册与发现使用的是**nacos**，负载均衡使用的是**Ribbon**

具体架构

auth-service：负责进行身份的校验，用户/管理员登录后会得到对应的token令牌，并存在Redis内存数据库中，

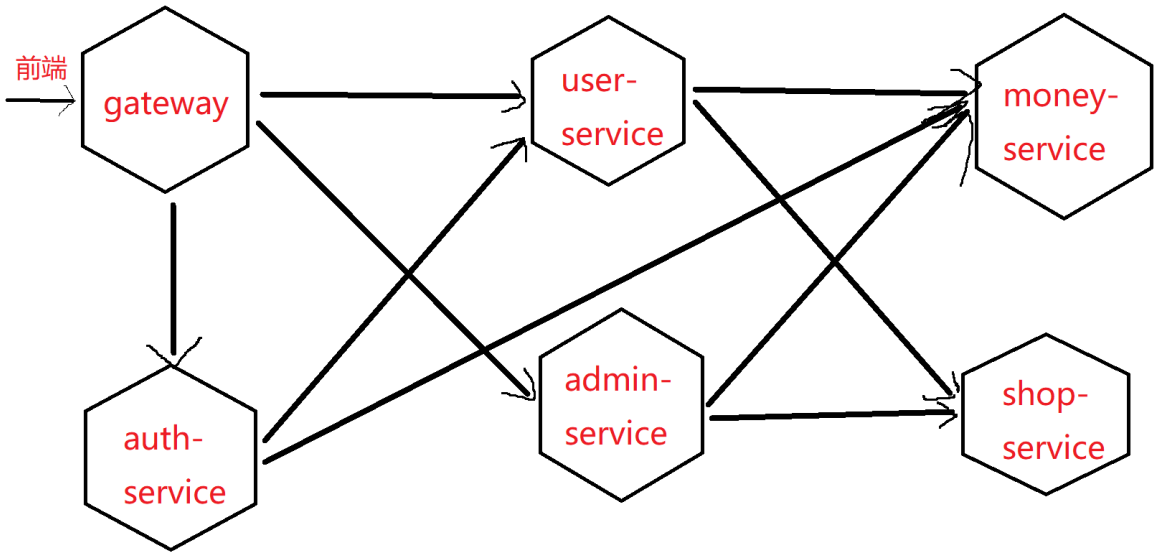
gateway：负责进行路由，前端的所有请求都发到gateway对应的端口，经过身份校验后（调用auth-service），再由gateway转发到对应具体业务的微服务。

money-service：负责和充值、消费有关的功能，管理所有用户的余额。

shop-service：负责和商品信息、订单、商品评价有关的功能。

user-service：负责用户个人信息、地址、购物车等信息的维护。

admin-service：负责管理员端特有的功能，例如查看所有用户的信息。



上图大致展示出了服务之间的相互调用关系，其中**auth-service**会调用其他微服务是因为，当用户注册账号时，需要在同时初始化用户的账号信息、个人信息、余额信息，这些信息是在不同的微服务中的管理的。

user-service和**admin-service**会调用**money-service**是因为当用户支付订单或管理员给用户充值时，涉及到用户余额的变更。

user-service和**admin-service**会调用**shop-service**是因为两者都需要查看商品和订单的相关信息，用户还会查看商品评价的相关信息。

每个微服务单独管理自己的数据库。

功能概述：常规功能

无需登录即可使用的功能：

登录、注册

管理员功能：

出版社的增加、修改和查询

对商品Tag信息的维护

对商品媒介信息的维护

商品的增加、修改和查询（支持限定Tag和媒介之后的查询）

可以查看用户信息，并为用户充值

可以查看订单（有四种，未支付，未发货、待收货、已完成），对于未发货的订单，管理员可以确认发货，并填写物流信息

用户端的功能

维护个人信息

管理收货地址

查看商品概览（支持限定Tag和媒介的搜索）

查看商品详情（支持加入购物车和立即下单）

查看自己的购物车，将购物车内的商品打包下单

可以查看自己的订单（有四种，未支付，未发货、待收货、已完成），支付订单时要顺便填写备注并选择收货地址

功能概述：一点探索

①下架/在售

管理员修改商品信息时，可以选择将商品调整为下架or在售。刚导入的商品默认是在售状态。

用户查看商品信息时，只能看到在售状态的商品，如果用户购物车中的商品下架了，则将购物车的商品打包为一个订单时，会自动去除已经下架的商品。如果购物车中都是已经下架的商品，则不会形成订单。

②商品评价

用户对订单确认收货后，即可对订单中的每一种商品填写对应的评价，并进行打分。

在查看商品详情的界面里，可以看到所有用户对这个商品的评价

数据库设计

决定好需要实现的功能后，就能敲定各个微服务的数据库中需要有表

auth-service：account表，存账户信息

user-service：user表存user的个人信息，address表存收货地址，shoppingcart表存购物车信息

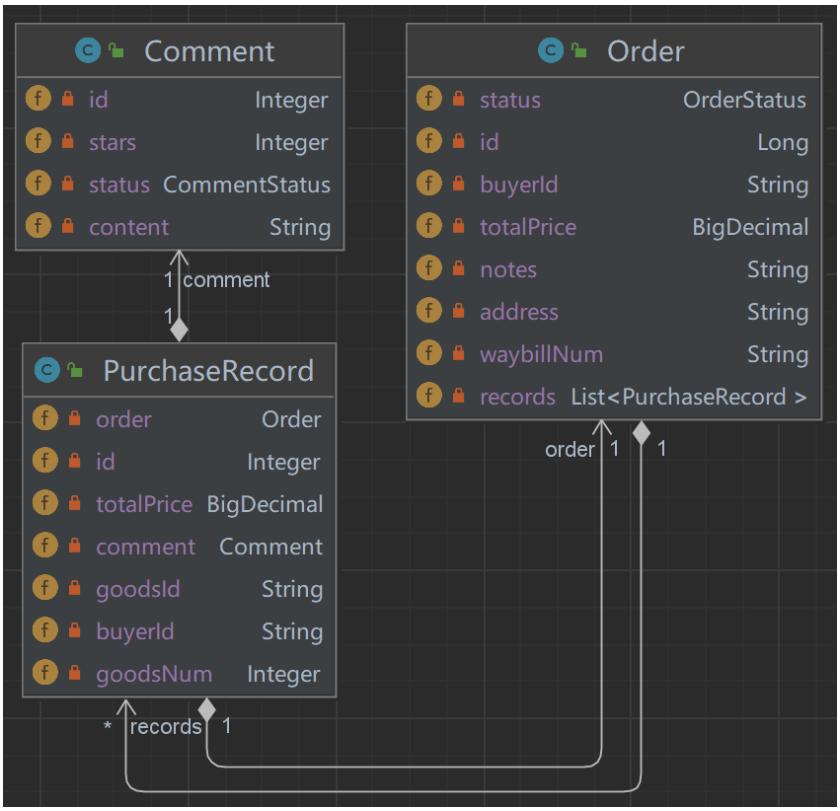
money-servie：money表存用户的余额信息

shop-service：tag表、medium表、publisher表、goods表存商品信息，order表存订单信息，purchase_record表存一个订单中每种商品的购买信息，comment表存评价信息

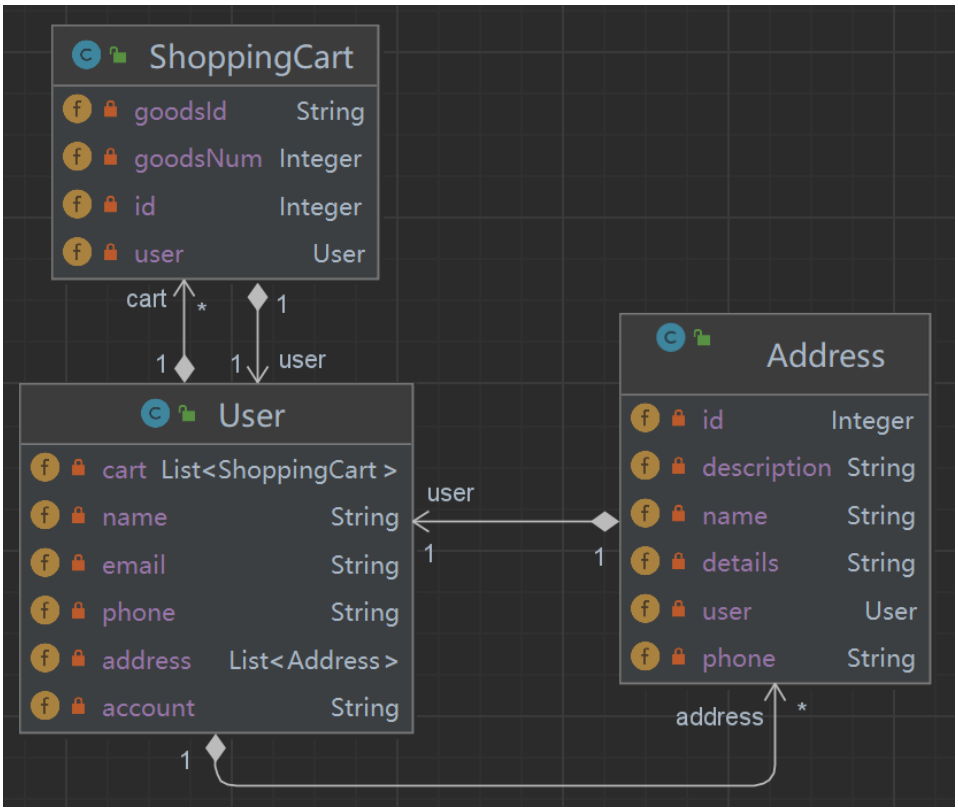
下面讨论个别表的具体实现：

首先是最基础的商品系统，商品的出版社，媒介，标签(类型)都应持久化，但商品本身不应对其有依赖。单独设计了商品（Goods），出版社（Publisher）,媒介（Medium），标签（GoodsTag）的实体类，并在其中包含必要的基础信息。此外，商品还有一个status用于表示其状态—下架/在售，只需对其进行修改并在查找时增加状态条件就实现了商品的上下架功能。

对于订单系统，如下图所示，首先有订单实体类，其中包含购买人、总价、快递单号、备注等基本必要信息。还有状态：未支付、未发货、已发货、已签收用于相关业务，还与购买记录维护一对多的关联关系。其中，每一条购买记录对应一个商品，并保存其购买数量和单独的总价格，此外为了方便业务与提升查找性能，在购买记录中添加了可以从order中获取的购买人id。而每一条购买记录都与评价维护一对一的关联关系，因为用户可以对买过的每一个商品进行评论。在评论实体类中保存评论的内容与打分(星)，并有状态status：待评价、已评价，在用户确认收货时便会生成相应的状态为待评价的空白评价，等待用户填写。



对于用户系统，如下图所示。除了基本的账号与个人信息外，首先是地址，每个user都可能有多个地址，所以其维护与address实体一对多的关联关系。然后是购物车系统，每个用户只能有一个购物车但购物车中可以有多个商品，所以我们把购物车每个不同商品的组成部分单独抽象出来作为一个小“购物车”，并让user维护与它一对多的关联关系，并且每个购物车都保存了其对应商品及其数量。



前后端交互风格

接口调用使用Restful的风格。

所有的**response都是data+code+msg。**

code为200表示请求成功，使用data中的数据，如果code不是200，则弹出一个表示失败的弹窗，并显示msg的内容

分工

张佳洵

负责后端微服务架构，完成和路由、认证有关的**gateway**和**auth-service**，以及**money-service**，完成其他微服务**controller**层的代码，使用**postman**对后端的所有功能进行测试，撰写接口文档。

参与讨论需求分析，一起决定需要实现的功能。

杨添淇

负责数据库表结构设计以及后端主要业务实现，完成**shop-service**和**user-service**和**admin-service**的持久化层和服务层的代码，并参与测试，修复相应Bug并优化功能。

参与讨论需求分析，一起决定需要实现的功能。

王骏飞

前端人员，负责前端界面的搭建，路由等，以及前端向后端的数据请求以及交互，以及数据在前端界面的展示等。对前端的功能进行测试。

参与讨论需求分析，一起决定需要实现的功能。