数据库说明：

1. 带红色的条目为主键，只有1对1关系的数据库才会设置主键
2. 无主键的数据库中数据关系不是1对1，可能是1对多等其他关系(比如一个用户可以有多个身份角色，一件商品可以有多个类别）
3. 对于积分规则这个部分，我们采取设计：平台规定签到获取积分规则（签到一天给多少积分，连续签到获取多少积分）。商户规定消费赚取积分和使用积分的规则（消费获取积分和积分抵钱）

接口说明（设计模式思想体现）：

1. 总体思想：Service层对一个或多个DAO层进行的再次封装，封装成一个服务，提供最基本的接口（增删改查等），而controller层提供对外的实际功能接口（比如和积分有关的积分兑换和积分消费接口。）
2. 在service层使用了一个模板接口的思想，设置了一个BaseService <T>的模板接口，里面有增删改查（查分为单项查找和按页查找），这样只需要具体完成BaseService的实现即可，大部分service层中的接口都可以直接通过继承service而完成增删改查，不用再单独去写实现。如果有不需要提供的CRUD的接口，可以单独去实现。这样就大大简化了代码复杂度。

补充：我们使用了Mybatis-spring-boot-starter这种较新框架，并使用了Mybatis的通用

Mapper，辅助BaseService的实现，借助通用Mapper提供的一些直接可以使用的增删改查方法，BaseService层也为Service层提供了基本的一些增删改查方法。

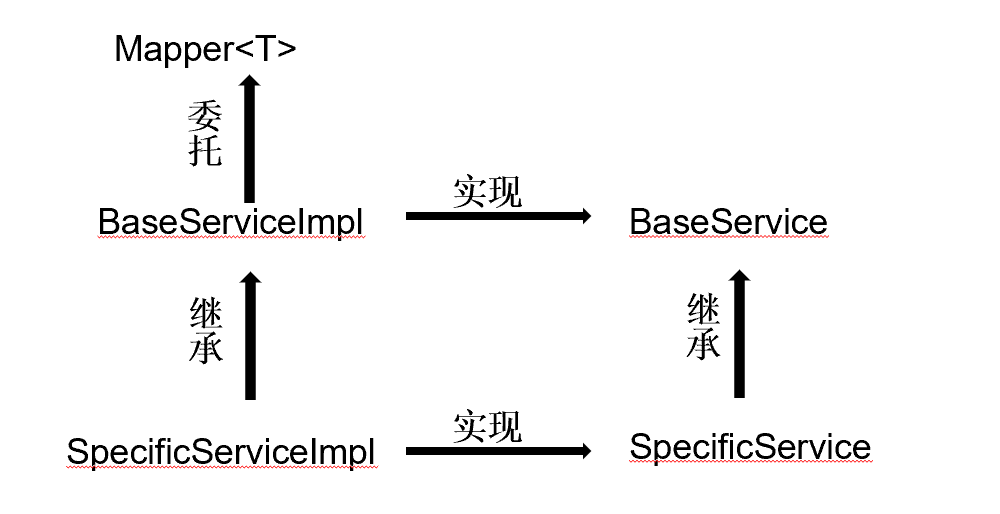


图1：项目后端Service层架构

3.对于某些一些特殊的服务，我们采用接口隔离原则，以UserService为例，普通用户只需要获取权限，则使用commenUserService这个接口，而对于管理员来说，不仅需要获取权限，还要进行增删改查的操作，所以会使用commenUserService和restrictedAuthorityService这两个接口。



4.对于需求中提到的先消费快要过期的积分，我们的设计是这样的，积分实体中加入积分余额CreditBalance这个字段，对于用户赚取的积分，它的积分余额即积分量；而当积分花费时，系统会先在数据库中，在增加的积分中查找即将过期（通过expireTime字段）且积分余额不为0的积分，然后在积分余额上做减法，（一条的积分余额不够，就多条凑一下），这样就完成了快要过期的积分先花费的需求，然后再在数据库中增加一条花费积分的记录，积分余额置0.