## Broker架构模式优点：

1. 将服务端部署到实现不同功能的服务器上

使用broker的架构模式后，对于client需要的服务请求，broker可以根据请求的类型将请求分派提供给server，由server对请求进行响应，再传回broker。这种架构决定了可以对server进行分类，不同的server实现不同的功能，不同的server可以根据自己实现的功能对相应的请求进行响应，而不是使用一个总的可以实现所有功能的服务器来处理所有的请求。这样的话，可以减轻server端的压力，将server端进行分布化，也使得server端更容易维护。

1. 分发控制，负载平衡。

通过broker可以控制对client端的请求的分发，这样在某个服务器压力很大的情况下，可以通过broker判断将请求转发到具有相同功能的服务器上，这样不会对已经负载很高的服务器继续施压，保持了负载平衡。

1. 可修改性，可扩展性。

在票务系统中，可能server端的架构会经常变化，client端根据客户需求的改变也可能会有所变化，但是他们都不依赖于彼此，他们只依赖于broker，所以这使得他们的可修改性很好，同时因为这样的原因可扩展性也很好。

1. 安全性，过滤恶意请求。

在票务系统中，经常可能有用脚本恶意刷票或者攻击服务器的行为。因为client对于server的服务请求和server对于client的服务请求回应都是要通过broker的，这种交流模式使得我们可以通过在broker进出口处添加安全验证来提高系统的安全性以及进行对用户购票行为进行验证。

# Client模块

## Client

client组件负责处理与用户相关的事务，帮助用户进行买票、查询余票等事务的操作

## Client-proxy

client-proxy组件负责client与broker交流，作为一个中间部件，它负责client与broker之间信息的编码、解码、发送的工作。在client组件向broker发送请求时，client-proxy组件从client组件得到相应信息，对需要传输的信息进行编码，然后发送给broker。而broker向client端发送请求回应时，client-proxy负责接收这些信息，然后进行解码工作，将解码得到的信息传送给client组件。client-proxy组件的存在减轻了client组件的压力。

## Request bus

request bus作为一个请求队列，负责协调client端与broker端信息的传输工作，管理信息的传输。

# Broker模块

## Monitor

Broker架构模式中broker的引入降低了整体架构的可靠性。同时，票务系统对软件的可靠程度有着较高的标准。在汲取broker模式的优点同时，我们需要有力的防御机制保证整体架构的可靠性。因为broker作为架构的枢纽，此处产生的错误必然给项目整体带来较大的影响，所以预防broker模块可能产生的错误是架构设计的一项关键一体。

Monitor被设计作为broker模块的监控组件，通过间歇性的消息传递机制（heartbeat）来侦测模块中可能出现的错误。错误产生时，Monitor可以采取有效措施来消除错误、规避风险、提交报告。

## AuthorityCheck

票务系统需要面对来自外部的恶意攻击和刷票行为，所以在UserBroker组件之前加入AuthorityCheck模块来过滤信息。一定程度上拦截无效或者恶意请求，保证Broker处理的请求皆为合法请求，处理压力不会过大。

## Broker

核心broker组件的任务主要有定位服务器，向合适的服务器转发请求，向客户端返回结果或异常。Broker使得客户端服务器两段持有更少量的知识，这部分知识集中在自己的身上，来提高整体架构的灵活性。但是，broker自己持有过多的知识也会造成负载的压力和知识的冗余。因此，设计多层broker来进行请求的两次转发可以有效的解决这项问题，降低组件之间的耦合程度。

UserBroker组件负责有关用户的请求转发处理，该broker向AccountBroker（处理账户信息）、OrderBroker（处理订单信息）、TicketBroker（处理票务请求）三个二级broker分送请求。//TODO

除此之外，虽然broker可能成为该架构模式的性能瓶颈，但是我们可以增加broker的数量，同时进行broker的冗余设计，作为某些broker崩溃的情况下的替代方案，保证整体性能。

## InfoBridge

Broker之间会进行数据交换，对于票务系统而言可能会传递车次信息、用户订单信息等等。设计bridge组件可以用于连接不同的broker，主要理由：

1. 隐藏了进行数据交换broker双方的实现细节，体现了面向接口的设计原则，降低模块间耦合。
2. Bridge可以对交换的数据进行封装，只传递核心有效的数据信息

# Sever模块

## Database

Database负责票务系统所有数据的存储工作，我们在这里分出了两个database，一个是存储和列车信息有关的数据，称为Train Schedule DB，另一个是存储和用户有关的信息，称为Ticket User DB。Database通过jdbc连接向server提供数据服务。

## Server and Server-proxy

server负责处理用户请求，对于传输进server的用户查票、购票等操作的请求，进行分析处理、查询数据库，然后返回用户需要的结果。

当broker传送数据给server cluster时，经过安全验证后，先由server-proxy接受信息，进行信息的解码，解释为server能够读懂的格式，接着将信息传送给server、调用server端相应的服务，由server进行处理后，对于server的请求回应，server-proxy将其编码然后发送给broker。这样server-proxy减轻了server的压力，处理了信息交流相关的事物，而使得server只关注于自己的工作。

## SecurityCheck

security check组件对broker端传输给server端的信息和server端传输给broker端的的信息进行验证。理由如下：

1. 这可以防止有的危险行为逃过了前一次安全性检查而对server造成攻击。
2. 如果有服务器被攻击，那可以将这些已经沦陷的服务器发出的有害请求过滤掉，不扩大其影响。

## 优化方案

1. 数据库分布式设计
2. Request bus 或者client-proxy在第一次进行请求访问时，由Distribution将客户端和某个特定的bus 或者 broker进行绑定，确保负载平衡
3. Broker和sever之间设计缓存模块，优化重复请求