1. 并发层面的问题

如果未来并发量持续增长，tomcat服务器可以部署集群+前面的负载均衡设备。

但是mysql是单库单表，会有并发的瓶颈，一般来说，mysql单服务器承载的并发不能超过2k。如果mysql单库的写并发到1.5k的时候，建议就要分库了；如果mysql单库的读并发到1.5k的时候，建议就要挂主从架构，读写分离，挂多个从库。

所以目前架构设计中的mysql单库单表可能未来会是一个风险点。但是现在系统刚起步，可以先这么做，但是以后要去考量这个事情。

2、伸缩性

tomcat是没有问题的，只要加机器就可以了。

但是你的业务系统，是否具备伸缩性，要仔细去考量一下。就是说，如果你的设计里面，你的业务系统必须用单个服务实例承接所有的请求才能正常工作，一旦你业务系统部署多个实例，请求落到不同的服务实例上，会导致请求处理出错，那么就意味着你的业务系统只能部署单个实例。

如果是这样子的话，那你的业务系统就不具备可伸缩的能力，因为他是没有办法去扩容的。

但是这里主要是提醒一下，在后面做详细设计的时候，务必要考量到这一点。

举个例子吧，假设，某个订单，被你的某个服务实例在内存中维护了一个什么状态。此时就导致，如果这个订单相关的请求落到其他的服务实例上去，会导致你的整个程序出错。那么此时，你的系统就是不可伸缩的。你只能要求所有的请求都要落到一个服务实例上，才能完成正确的操作，那你不能扩容了。

mysql，单库单表，是典型的不可伸缩。你即使加更多的mysql服务器，也是搞不定的。所以以后要让mysql可伸缩，必须结合业务去设计对应的分库分表的mysql架构。

3、可用性

1. tomcat+业务系统的部署，都是单机部署的，那么意味着，只要这个机器，或者是这个机器上的tomcat，或者是这个tomcat里部署的业务系统，挂掉了，那么整个系统就挂掉了，可用性很差
2. mysql，只要一个mysql宕机了，那么数据库就挂了，一旦数据库挂了，那么整个系统就挂掉了，可用性很差

所以在现在初期，先不用考虑，但是以后架构必须得去考量这个可用性的问题

4、扩展性

扩展性很差，因为你如果要大团队协作，多人开发，会发现说，这里的模块耦合太严重了。都耦合在了一个单块应用中，到了后期，几十个人修改一个工程，大量的代码冲突，不同的项目不同的版本之间的测试、开发的排期大量的冲突，互相之间占用测试环境，你等我，我等你。然后的话，上线，也是的。

假设，我们现在有3个项目并行在做，分别是3个小team在做，一起修改一个单块应用。然后其中一个项目的上线时间定在了07.01，那么这就意味着什么呢？项目二的上线时间就不能定在07.01，如果项目二和项目一同一天上线，就意味着大家要耦合在一起做测试啊什么的，是不可能的。这样的话，就导致项目二只能定在07.02，然后同理，项目三只能定在07.03.

这个还只是3个项目，如果是30个项目呢？上百人开发。

比如你临时修复了一个bug，要上线，这个时候怎么办？可能有其他团队正要上线呢，你还得等他们先上线，然后你再上线呢？

暂时不用解决，1+4的一个小team，单块应用。10个人以上，那就不太合适了。

5、安全性

目前没有考虑任何的安全性的措施，裸奔。内测，仅仅面向部分少量的用户，用一些测试数据来内测，所以还好。

6、高性能

性能肯定的是有问题的，大量的全部基于数据库来操作，没有任何的缓存，或者是MQ介入，那么性能肯定是会差很多的

7、稳定性

系统会不会频繁的出一些bug和未知的一些问题，排查问题的效率和解决问题的效率怎么样？日志，通过日志可以排查和解决问题。

但是监控这一块，做的比较low，只能通过spring boot提供的接口，人肉时不时上去看看，调用一下接口看一眼。没有那种自动化的监控系统，可以看到问题，然后直接发短信、邮箱通知RD。