# 项目概要介绍

## 前言

在互联网的大环境下，越来越多的行业加入了数字化的赛道。积极整合资源完成产业结构升级，成为了国内大小公司前行的方向;与此同时，银行业也在积

极的跟上脚步，依托互联网进行技术重整与升级。伴随着大量新技术的涌现，积

极响应、快速迭代、持续集成等口号不再专属于科技公司，如今的银行业，同样

需要行业先进技术的使用;三高是互联网场景下亘久不变话题，高并发是实际业

务下不可避免的前提，高性能与高可用则直接决定着用户的体验，这三者如何进

行合理的结合，在银行业也将越来越举足轻重。

银行业竞争日趋白热化，获客成本持续增长，传统银行借鉴互联网营销方式

成为破局之道，其中定期推出不同的秒杀活动是最常见的几种营销方式之一。

本项目银行秒杀系统，正是应运而生。秒杀业务在很多电商领域，有很多的体现，大家熟知的618、双11大促等等。那么何为秒杀场景呢？简单来说，就是一件商品的购买人数远远大于这件商品的库存，而且这件商品在很短的时间就会被抢购一空。秒杀业务，是典型的短时大量突发访问增大的问题。由此可见，秒杀业务的特点：秒杀时网站的访问量大增（高并发）、秒杀购买的请求数量远小于库存。

接下来，我们详细的介绍本项目，如何面对短时间内大量的请求。

## 创意描述

本项目，在处理用户大量的请求系统登录的时候，我们会提前将用户的信息放在缓存（本地缓存和Redis缓存）中，来减少对数据库的访问，以便后面对用户的其他操作。秒杀的时候，做了验证码和安全校验，起到限流作用和防止同一ID或IP重复购买。同时如果同一ID频繁的请求，我们会直接拒绝该用户的请求。同时也会对秒杀的用户又一个状态的判断，每个用户在登陆我们的系统，都会携带UUID，也就是”ticket”，并设置了一个相应的过期时间。如果到了过期时间，系统会直接强制下线。

为保证秒杀接口的安全性，防止其他开发人员恶意透露秒杀链接地址，提前秒杀商品，在获取秒杀接口的我们做了接口地址隐藏，我们生成秒杀接口的地址，是根据随机UUID来拼接。同时在点击秒杀接口的时候，要求用户输入数字计算验证码，将所有的请求分到不同的时间片段上，来达到分散用户的请求的效果。

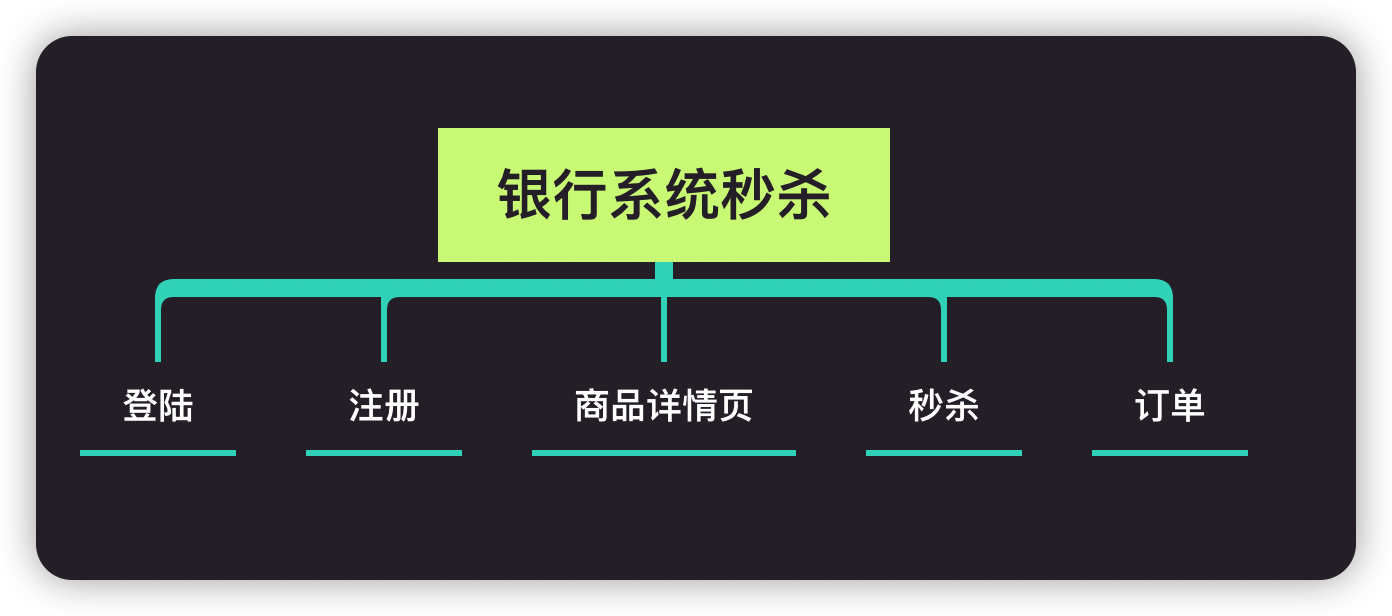
同时在秒杀的时候，会有一个对用户的筛选。满足条件的人才可以，继续后续的操作。

在后续的秒杀执行流程中，我们采用了Redis的单线程的特性，避免不必要的上下文切换和竞争条件、同时它的大部分请求都是纯粹的内存的操作，速度很快、同时它是一个非阻塞I/O多路复用，在I/O操作上浪费的时间很少。当用户点击了秒杀按钮，后台业务会去Redis的缓存中查询库存的数量，利用缓存可以减少服务器的响应时间，提高用户的体验度。

如果缓存中的库存还有剩余，会利用Redis的单线程去预减库存，这里的减库存是原子性的操作。同时为了减轻Redis的压力使用了Map标记库存。如果库存不足，会给用户友好的提示。同时为了防止用户重复购买，也增加一个判断，同时给用户友好的提示。

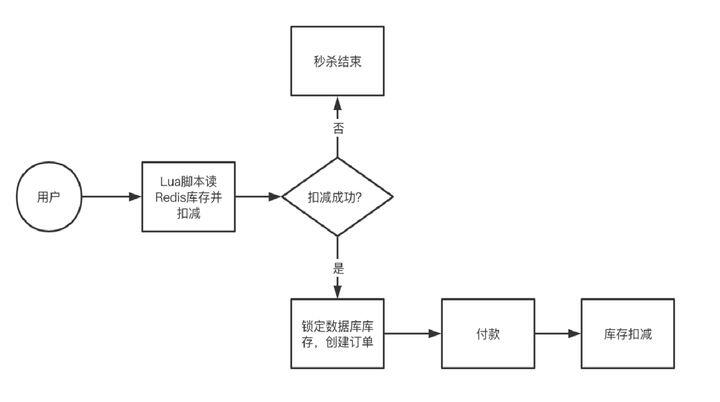
如果上述操作，都没有问题。在生成订单的过程，我们会利用RabbitMQ异步队列生成对应的订单，同时利用RabbitMQ的交换机模型，给用户一个提示，提醒用户支付，如果用户支付成功，才去数据库中真正的扣减库存。

## 功能简介



## 特色综述

在防止超卖做了具体的技术实现。由于网络带宽在很大的程度上会影响Redis的性能。



后台服务对秒杀页面，进行了页面静态化的处理。

后台服务加密算法，采用SM3密码杂凑算法。

项目使用Docker容器化，部署。

## 开发工具与技术

开发工具，IntelliJ IDEA 2021.3.3、Navicat Premium 15.0.29、Redis Desktop Manager、Docker。

开发技术，SpringBoot 2.5.6、MySQL 8、Redis 6.0.6、RabbitMQ 3.9.1、MyBatis-Plus 3.4 3.4。

## 应用对象

本系统应用对象主要为银行业解决业务拓展、业务的落地提供了具体的解决方案。

## 应用环境

生产环境：Tomcat 9、MySQL 8、Windows 10等。

上线环境：Docker、MacBookAIR M1等。

## 结语

通过这次系统的设计，让我们学习到了很多的东西。也感受到了什么叫做真实的业务场景。综上所述，高并发的应用是一个复杂的系统，它涉及到软硬件、服务器和中间件等方方面面的内容。构建高并发的WEB应用，基本出发点是系统的分层与解耦，将系统组件化，让每个组件充分发挥效能。