MASTER THESIS XIDIAN UNIVERSITY

西安电子科技大学

硕士学位论文

基于Redis的股票交易系统的设计与实现

作者姓名 孙杰杰

学校导师姓名、职称 鲍亮 副教授

企业导师姓名、职称 张小芳 高工

申请学位类别 工程硕士

学校代码 10701　　学　号 1503121744
分 类 号 TP31　　密　级 公开

西安电子科技大学

硕士学位论文

基于Redis的股票交易系统的设计与实现

作者姓名：孙杰杰

领　　域：计算机技术

学位类别：工程硕士

学校导师姓名、职称：鲍亮 副教授

企业导师姓名、职称：张小芳 高工

学　　院：计算机学院

提交日期：2018 年 6 月

Design and Implementation of Stock Trading System Based on Redis

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Computer Technology

By

Sun Jie Jie

Supervisor: Bao Liang Title: Associate Professor

Supervisor: Zhang Xiao Fang Title: Senior Engineer

June 2018

摘要

近年来，随着“互联网+”这一新兴互联网概念的普及，互联网与传统产业的结合已经成为我国经济发展的新形态，金融证券产业与互联网的结合已经成为大势所趋。虽然现在市面上已经存在股票交易软件，但是通过对软件的研究分析发现存在着缺陷和不足。首先，随着用户规模的不断增长，系统的稳定性就会下降；其次，在可交易时间内，股票行情显示略显滞后，给交易用户带来很大的困扰。除此之外，部分软件提供的股票信息过于专业，不利于散户的学习，在界面设计上没有充分考虑到用户的操作习惯。因此，开发一个功能完善、性能良好且界面优良的股票交易系统变得至关重要。

本文首先结合热点新闻、股票详情、K 线走势、股票交易等多种方式建立股票交易体系，通过提供多维数据，使得用户尽可能的了解多方位的信息，在此基础上，利用动态时间规划算法进行相似 K 线的计算，通过对观测样本的选取以及对结果的加权计算提升预测结果的准确度；然后以软件工程思想为指导，通过需求分析，设计实现了一套基于 B/S 架构的股票后台管理平台和基于 C/S 架构的移动交易平台，并通过 SpringMVC 框架完成视图、控制、模型三层结构的代码编写。该系统采用分布式的系统架构，并利用 Nginx 进行负载均衡控制，考虑到后期用户量增多的问题，通过横向扩展机器节点，解决系统的瓶颈和效率问题。利用 JS 脚本定时的调用第三方数据服务提供商所提供的 WebSocket 接口，把行情数据及时地存储在 Redis 中提高接口的响应速度。为了提升用户的交易体验，整个交易流程被划分为用户委托、委托上报以及成交回报三个独立的子流程，通过 Quartz 调度框架 Job 并发的处理不同的交易流程，提升系统的交易效率。系统包括行情模块、交易模块和交易管理模块三大部分，其中交易模块通过相似 K 线查询分析功能的开发，用户可以查看所查股票走势最为相近的 K 线详情，为投资选择保驾护航。交易模块包括资产查询、买入卖出委托、撤销委托、委托记录查询四大功能。该模块是系统的核心模块，通过对该模块的实现，用户在 APP 注册成功之后通过获得赠送的模拟币，可以体验真实的股票交易过程。交易管理模块面向系统管理员，支持账户管理、风控管理、交易管理以及结算管理，系统管理员通过交易管理平台对股票交易过程中涉及到的交易日、交易费率、用户账户、账户流水等信息进行综合管理，保障系统运行安全。最后搭建系统测试环境，对系统的功能和性能进行测试，使系统在访问量增多的情况下仍然可以正常运行，达到企业的预期效果。

系统经过完整的功能测试和性能测试，达到了最初的设计要求。在线上投入使用后表明，本软件很好的解决了当前市面上同类软件所存在的问题，使得用户的交易体

验得到很大的提升，具备很好的现实意义。

关 键 词：互联网+， Redis， 高并发， 实时性， 股票交易

ABSTRACT

In recent years, with the popularity of the “Internet+” Internet concept, the combination of the Internet and traditional industries has become a new form of China’s economic development. The combination of the financial industry and the Internet has become a general trend. Although stock trading software already exists on the market today, it has been found that there are flaws and deficiencies through research and analysis of software. First of all, as the scale of users continues increasing, the stability of the system will decline. Second, in the tradable time, the stock market shows a slight lag, giving traders a lot of trouble. In addition, the stock information provided by some software is too professional, which is not conducive to the learning of retail investors. The interface design has not fully taken into account the user's operating habits. Therefore, it is very important to develop a stock trading system with complete functions, good performance, and excellent interface.

This paper first establishes the stock trading system in combination with hot news, stock details, K-line trends and stock trading. By providing multi-dimensional data, the user can understand as much information as possible, and on this basis, the calculation of similar K-line by using the dynamic time planning algorithm, and the accuracy of the prediction result is improved by the selection of observation samples and the weighted calculation of results; Then, based on the software engineering idea, a set of stocks based on the B/S structure is designed and implemented through demand analysis. Background management platform and mobile trading platform based on C/S architecture, and through the SpringMVC framework to complete the view, control, model three-tier structure of the code. The system uses a distributed system architecture and uses Nginx to control load balancing. It takes into account the increase in the number of users in the later period, and extends the machine nodes horizontally to solve system bottlenecks and efficiency problems. Use the JS script to periodically call the WebSocket interface provided by the third-party data service provider to timely store the market data in Redis to improve the response speed of the interface. In order to enhance the user's trading experience, the entire transaction process is divided into three independent sub processes, which are user entrustment, entrustment and transaction return, and deal with different transaction processes.The Quartz scheduling framework concurrently handles different transaction

processes to improve the transaction efficiency of the system. The system includes three parts: market module, transaction module and transaction management module. The transaction module is developed through the similar K-line query analysis function. The user can view the K-line details of the most closely related stock trends to protect the investment. The transaction module includes four functions: asset query, buy and sell mandate, withdrawal mandate, and entrusted record query. This module is the core module of the system. Through the realization of this module, the user can experience the real stock trading process by obtaining the simulated coin after the APP registration is successful. The transaction management module is for system administrators and supports account management, risk management, transaction management, and settlement management. The system administrator uses the transaction management platform to deal with the trading days, transaction rates, user accounts, and account flow involved in the stock trading process. Comprehensive management of information ensures the safe operation of the system. Finally, the system test environment is set up, and the system functions and performance are tested to ensure that the system can still operate normally under the condition that the number of visits increases, and the expected effect of the enterprise can be achieved.

The system has undergone complete functional testing and performance testing to meet the initial design requirements. After being put into use online, it shows that the software solves the problems existing in similar software on the market and makes the user's trading experience greatly improved. It has very good practical significance.

Keywords: Internet+, Redis, High Concurrency, Real-time, Stock Trading

插图索引

表格索引

符号对照表

符号	符号名称
X	时间序列 X
Y	时间序列 Y
$head(Y)$	时间序列 Y 的第一个元素
$rest(Y)$	时间序列 Y 除第一元素外的剩余元素
$<>$	空时间序列
D	动态规整距离
D_{base}	欧几里得距离
$p(t)$	第 t 个交易日的收盘价格
$v(t)$	第 t 个交易日的日成交额
x_t	第 t 个交易日的观测样本
X_L	L 个交易日的时间序列
r_t	第 t 日的收益率
$\hat{r}_{t+1}$	样本 K 线的交易次日收益率估计

缩略语对照表

缩略语	英文全称	中文对照
SNS	Social Networking Services	社交网络服务
DTW	Dynamic Time Warping	动态时间规整
HTML	HyperText Markup Language	超文本标记语言
RPS	Requests Per Second	每秒请求数量
URL	Uniform Resource Locator	统一资源定位符
JSON	JavaScript Object NotationJS	对象标记
IO	Input/Output	输入/输出
ER	Entity Relationship	实体联系
RDBMS	Relational Database Management System	关系数据库管理系统
SQL	Structured Query Language	结构化查询语言

目录

第一章 绪论

1.1 选题的背景与意义

近几年来，随着互联网技术的迅猛发展，国家顺应时代的发展潮流创新性的提出了“互联网+”这一概念，“互联网+”是在互联网思维的基础上延伸出来的，通过利用互联网的优势，转型和升级传统行业的经济发展模式[1][2]。随着“互联网+”这一概念的深入人心，各大金融证券公司一直致力于将证券行业与互联网进行深度融合，以期利用互联网的便利，为广大用户提供一个快捷、安全的平台。

股票交易系统通过对用户所持股票的历史涨跌幅进行分析，从而帮助用户合理买卖股票并获取收益。由于股票市场的行情瞬息万变，需要及时地让用户了解到股票的行情变化，所以本系统对实时性的要求较高。由于传统的关系型数据库对于频繁的数据读写操作支持能力较差，而且只支持结构化数据，可存储的数据类型过于单一。为了保障系统的实时性，确保系统的性能要求，引入了基于键值对的内存数据库Redis[3]，它是一个开源的，具备优秀读写性能的缓存存储数据库[4]，操作简单而且具备丰富的功能，受到了市面上各大公司的青睐，而且相对于传统的关系型数据库具有无法比拟的优势。首先，它支持丰富的数据存储结构，如字符串类型、散列类型、列表类型、集合类型、有序集合类型[5]，这种字典结构的存储方式可以应对大部分场景下的数据操作需求，提升开发者的开发效率；其次，内存数据库中的所有数据都是存储在内存当中，就读写性能而言，内存的读写性能远快于硬盘，所以相较于传统数据库，Redis 在读写性能上具备明显的优势[6]；最后，通过对并行处理方法、数据缓存以及快速算法等方面的改进，内存数据库极大地提高了对于数据的处理效率。因此，对于内存数据库以及 Nginx 负载均衡系统的选择，可以有效地满足股票交易系统对于实时性和高并发性的要求，有效地解决了在高频读写情况下系统负载过重的问题，使得系统的性能在各个方面都得到了极大地提升。

本论文来源于作者研究生阶段在实习公司参与研发的股票交易项目。项目主要包含行情模块、交易模块和交易管理模块三大部分，整套系统是建立在 SpringMVC 框架上，由于股票行情根据市场表现不断变化，交易模块在开盘时可能会遇到大量的交易请求，对系统的实时性和并发性提出了很高的要求。经过对数据访问频率、数据类型以及技术成熟度等多方面的考虑决定采用 Redis 内存数据库，对于经常需要频繁访问的数据如股票行情数据，热点新闻数据等，利用 JavaScript 脚本定时把第三方数据服务提供商提供的高频数据定时的存储在 Redis 数据库中。传统数据库则用来存储用户基本信息、系统活动信息等对访问性能没有过高要求的数据。

1.2 国内外研究现状

现在国内虽然也有很多股票交易软件，但是都或多或少的存在着一些缺陷和不足，尤其表现在对炒股用户股票知识储备情况考虑不足、系统性能以及操作体检性较差方面。

国外股票市场的建立时间领先于国内将近百年的历史，因此对股票交易系统的方面的技术研究水平比国内更成熟一些，欧美国家的股民早已经普遍利用互联网的便利进行网上交易。得益于互联网的迅猛发展以及手机技术的不断成熟，使得越来越多地欧美股民加入到网上股票交易的行列当中，高频交易在此阶段也得到了迅速发展。以美国为例，根据最新的根据美国股票交易市场的调查发现，美国各大股票交易公司的交易额大部分都达到了历史最高点，规模更是前年的四倍，达到 6 万亿之多。另外，通过查阅美国证监会的资料，美国交易市场上高频交易的日均交易量占总日均交易量的 50%以上[7]，高频交易量所占份额已经从 2012 年的 41%上升到 2016 年的 81%[8]。另一方面，随着互联网的普及，美国大部分民众可以随时随地的查看财经信息，同时美国也有大量的股票交易软件为股民服务，股民也可以通过软件更加全面的了解最新资讯。除了美国之外，其它欧洲国家情况也比较类似。股票交易软件经常被股民频繁使用，并在总体的股票交易份额中起着举足轻重的作用。

国内的股票市场起步较晚，近几年随着我国经济的发展，虽然已经缩小了一部分差距，但是不可否认的是国外股票市场机制还有很多值得学习借鉴的地方[9]。我国目前比较流行的炒股软件有东方财富、益盟操盘手、易盟等，这些软件虽然功能比较全面，但是也都存在着一些弊端，即软件的功能过于专业，对于刚入门或者股票知识储备不足的散户很不友好，不能够从中提取出对自己有效的信息，稍有不慎便会造成经济方面的损失。同时国内的股票交易软件设计界面过于注重提供股票信息的价格走势以及分析图标，没有多维数据的帮助，用户往往需要对看到的图表进行二次加工才能做出自己的投资决策，且在交易量较频繁的时间段内，会出现卡顿、闪退的现象，因此还有很多需要改善的地方。

由此可以得出结论，国内外对于股票交易系统的研究还存在着较大的差距，国外的研究水平更高，技术也更成熟，然而由于国内外股票市场机制的不同，不能简单的将国外的研究水平直接应用到国内的股票市场。本系统针对目前股票交易软件存在的问题，进行了部分改进。

（1）由于我国股民人数众多，但是相关股票知识储备水平参差不齐，考虑到初次股票交易的用户，一方面系统开发出了相似 K 线分析查询功能，能够对股票未来的走势进行一定的预测，另一方面，还向刚注册的用户提供了一定数量的模拟币，可以完全按照真实的股票交易流程进行买卖，提升用户在股票交易方面的经验。

（2）在系统架构方面，采用了分布式的系统结构，利用 Nginx 进行负载均衡控制，有效地提升系统的并发能力和扩展能力[10]。在数据库方面使用 Redis 作为系统的缓存服务器，将股票行情数据，常量数据以及股票交易费率存储在 Redis 中，提升接口的响应时间。

（3）对设计界面进行一定的优化，尽可能的迎合用户的使用习惯。

1.3 本文的主要研究内容和结构

1.3.1 本文的主要研究内容

本论文名为基于 Redis 的股票交易系统，研发本系统的主要目的是为了给用户提供一个便于进行股票交易的操作平台。系统在股票分析，股票买卖功能的基础功能上，为了保障用户交易的安全性，同时对用户的交易信息进行后台管理。以下对本文的研究内容进行简要的说明。

（1）针对股票交易系统的特点，在调研金融市场的基础上，制定出适合本公司发展战略的产品需求说明书，然后从系统的实时性、并发性和可扩展性等角度进行多维的分析，选取合适的技术框架，制定出可行的技术分析说明书。

（2）在需求分析研究结果的基础上对系统的整体架构进行详细的讨论设计。考虑到项目会随着用户的反馈不断地进行调整，因此项目的后台业务逻辑使用了 SpringMVC 框架，减少了代码之间的耦合度，方便后期维护。随着用户的不断增长以及交易复杂度的提升，系统采用了分布式任务调度框架 Quartz 以及 Nginx 负载均衡以提高系统的处理效率，提升用户的使用体验。由于本系统需要为用户提供实时的行情信息，需要存储热点新闻，股票行情，图片之类的非结构化数据，同时系统也需要存储容量较大的用户信息、交易日信息、合约信息等，因此系统采用了内存数据库 Redis+MySQL 的存储结构，以满足系统的存储需求。

（3）对股票交易系统进行开发和测试。在系统全部开发完成之后，需要对模块进行详细完整的功能测试和性能测试，确保系统能够完成既定的功能，使系统在上线之后能够稳定、安全的运行。

1.3.2 本文的组织结构

第一章，绪论。主要介绍了基于 Redis 的股票交易系统的背景与意义，总结了国内外在相关领域的研究现状并引出本文的章节安排。

第二章，相关技术介绍。对在实现股票交易系统的过程中所涉及到的 NoSQL、内存数据库 Redis、相似 K 线算法、WebSocket 长连接以及定时任务调度系统 Quartz 技术进行了介绍。

第三章，系统需求分析。重点介绍了系统在总体结构、功能和性能上的需求，并根据模块的不同，通过用例图对各个功能点进行了阐述。

第四章，系统的设计与实现。首先根据系统的的业务，对系统的架构、数据库以及功能模块进行了总体设计，然后根据模块的不同，结合序列图和类图，详细的描述了各个功能模块的设计和实现方式。

第五章，系统测试。介绍了系统搭建的测试环境，对系统的每个功能模块设计用例进行测试并分析总结性能测试结果。

第六章，结束语。主要是对股票交易系统的分析以及实现过程进行总结，阐述了本文的不足之处，提出了以后的改进和发展方向。

第二章 相关技术介绍

2.1 NoSQL 简介

据有关部门统计，随着互联网 Web 2.0 网站的飞速发展，我国的网民数量已经得到了快速增长。在此背景下，互联网数据也日益增多，由于大数据背景下数据呈现出了海量化、多样化、实时化的特性[11]，传统的关系型数据库在面对此类场景的互联网应用时已经显得捉襟见肘[12]，暴露出难以解决的弊端。

（1）高并发的读写需求。在互联网刚刚兴起的时代里，静态网页一般就能满足大多数网民的要求，但是随着 Web 2.0 时代的兴起，用户对网站的性能需求愈发高涨，一万次每秒的用户请求已经成为普遍现象。现在的社交网络服务（Social Networking Services，SNS）网站对数据库的并发要求非常高，而传统的关系型数据库由于其 ACID 特性的限制不能很好的应对上述需求。

（2）多样化数据模型存储需求。Web 1.0 时代，静态页面的单一性导致对数据类型没有过多的需求，传统的关系型数据库完全可以轻松应付数据存储的压力。但是随着 Facebook、微博、Instagram 等社交网站的兴起，用户已经习惯于分享图片、视频、音频、文章等非结构化数据，因此，开发者们需要能够解决高效地存储此类数据的问题，但遗憾的是，传统的关系型数据库由于其自身的限制无法容纳新的数据类型，对于非结构数据的存储更是显得捉襟见肘。

（3）数据库的高扩展性的需求。在传统的分布式 Web 系统架构中，随着访问量的日益增多，给应用服务器的负载能力带来很大的考验，传统的做法是通过添加更多的硬件和服务节点来缓解压力，但是数据库服务器由于是中心化的，很难通过简单的横向扩展来缓解高访问量所带来的系统压力。

通过对互联网 Web 2.0 网站下大数据特性的分析，不难发现传统的关系型数据库由于其自身的局限性已经远远不能胜任数据存储的要求，为了应对上述问题，给用户带来良好的上网体验，2009 年人们提出了 NoSQL 的概念，作为一种开放型的数据库系统，NoSQL 是建立在 CAP 定理和 BASE 理论上的。

如图 2.1 所示，CAP 定理是由一致性（Consistency）、可用性（Availability）、分隔容忍（Partition tolerance）三个要素组成[13]，它指出对于一个分布式计算机系统而言，最多只能同时满足两个要素，基于此定理，NoSQL 数库主要分成 AP 原则、CA 原则、CP 原则三种类型。

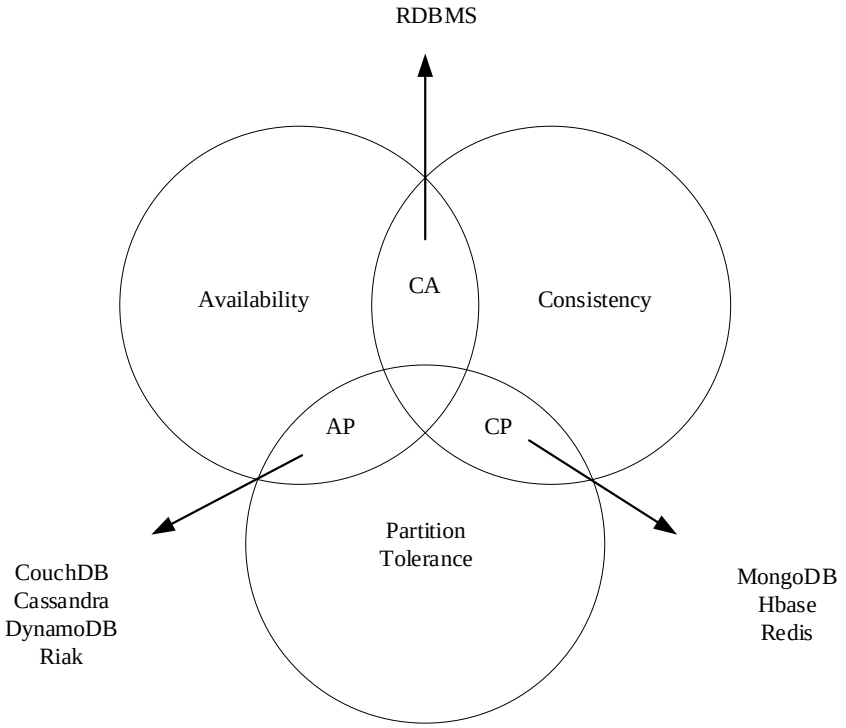


图2.1　CAP 定理

BASE 思想，是在 CAP 定理的基础上提出的，是对 CAP 定理的进一步细化，其目的是为了解决传统的关系型数据库因为其强一致性的特点进而导致的可用性降低的问题[14]。BASE 思想主要是在某一时刻通过放松系统对数据的一致性来换取系统整体性能和伸缩性上的改观，但是前提必须要完成系统最基本的服务功能。

最终，CAP 定理和 BASE 理论奠定了 NoSQL 在数据存储领域的地位，在传统的 RDBMS 的基础上，完美地解决了传统数据库难以横向扩展的问题，针对传统关系型数据库的运用，这一概念的提出给分布式数据存储领域带来了全新的理念。目前，市面上存在很多基于 NoSQL 的数据库都得了广泛应用，主要分为四大类：键值数据库、文档型数据库、图形数据库、列存储数据库[15]，具体的分类如表 2.1 所示。

表2.1　NoSQL 分类

分类	Examples 举例	典型应用场景	数据模型	优点	缺点
键值	Tokyo Cabinet/Tyrant,Redis, Voldemort,Oracle BDB	内容缓存，主要用于处理大量数据的高访问负载	Key 指向 Value 的键值对，通常用 hash table 实现	查找速度快	数据无结构化，通常只被当作字符串或者二进制数据

表 2.1（续） NoSQL 分类

列存储数据库	Cassandra,Hbase,Riak	分布式的文件系统	以列簇式存储，将同一列数据存在一起	查找速度快，更容易进行分布式扩展	功能局限
文档数据库	CouchDB,MongoDb	Web 应用	Key-Value 对应的键值对，Value 为结构化数据	数据结构要求不严格，表结构可变，不需要像关系型数据库一样需要预先定义表结构	查询性能不高，而且缺乏统一的查询语法
图形数据库	Neo4J,Info Grid,Infinite Graph	社交、推荐系统等。专注于构建关系图谱	图结构	利用图结构相关算法。比如最短路径寻址，N 度关系查找	需要对整个图做计算才能得出需要的信息，而且这种结构不太好做分布式集群方案

2.2 Redis 概述

Redis 是一款基于内存、开源、高性能的键值对 Key-Value 数据库[16]。Redis 的读写性能非常高，根据官方提供的测试数据表明，即便是在一个普通的 Linux 机器上运行，Redis 的读写性能也可达到每秒八万次的效率。同时，Redis 也具有非常丰富的数据存储结构，与其他非关系型数据库不同的是，在 Redis 数据库中存储的数据结构不仅仅限于 String 类型，还包括诸如 Hash、Zset、List、Set 四种抽象数据结构。除此之外，Redis 还具有非常优秀的特性。

（1）支持数据的持久化操作

Redis 所具备的强劲性能在很大程度上是因为把数据存储在了内存当中，但是由于内存相对于硬盘存储容量十分有限，一旦发生断电等无法预料到的事故时，就会导致存储在内存中的数据全部丢失，给用户和公司带来无法估量的经济损失。所以数据的持久化就显得尤为重要。持久化是指能够把数据从内存中以某种方式存储在硬盘中，

使得数据能够在发生意外的情况下恢复到内存中。目前，Redis 支持两种持久化操作，一种是 RDB 模式，即全量备份模式。这种模式是在符合一定条件时把内存中的数据以快照的方式存储在硬盘中，一旦发生内存数据丢失，Redis 便会通过读取已经保存的 RDB 文件把数据恢复到内存当中。另一种是 AOF 模式，即周期性增量备份模式。这种模式会把所有对数据的操作命令写入到 AOF 文件中，当需要恢复数据时，通过执行 AOF 文件便可以把数据重新加载到内存当中，以此保障数据的安全性。虽然这两种备份方式都能实现数据的持久化，但是在性能上都存在一定的缺陷，因此现在的主流做法是两者结合使用。

（2）Redis 集群

Redis 集群是一种分布式数据库方案，主要保障在多个 Redis 节点间共享数据[17]。常用的集群方案主要有三种。第一种，Redis Sharding 集群。其主要思想是通过 hash 将 Redis 的数据进行散列，通过哈希函数，映射到对应的 Redis 节点上，有效地解决了多个节点之间协同服务的问题，但是其节点扩展和收缩并不友好。第二种，使用第三方中间件进行集群配置。现在最常用的是由 Twitter 开源的 Redis 代理 Twemproxy，此种集群的配置方式操作简单，不需要更改任何的代码逻辑，能够有效地减少客户端与 Redis 实例的连接数，但是会造成性能损耗，尤其是在因为业务增长的背景下不能够平滑的增加 Redis 实例。第三种，Redis Cluster 集群。是官方认可的集群技术，有效地解决了多个 Redis 节点之间协同服务的问题，结构简单、扩展方便且可用性比较高。

综合以上信息不难发现，在应对数据结构复杂，并发量高的且扩展性要求较高的业务场景时，Redis 做为缓存服务器是一个非常理想的选择[18]。持久化的操作在系统出现崩溃或者不可控的状况时，可以通过对数据恢复以保障数据的安全性避免给公司和用户带来不必要的损失。Redis 的集群化可以大大的增强整个系统的负载能力，在保证系统健壮性和扩展性的同时带来良好的用户体验。所以，股票交易系统选取 Redis 作为缓存服务器。

2.3 相似 K 线相关算法概述

相似 K 线是通过历史形态对未来股票走势的一种预测，基于证券市场技术分析的三大假设：市场行为包含一切信息、价格沿趋势方向运行、历史会不断的重演。其中历史会不断的重演这一假说是股市技术分析的重要基石。技术分析实际上就是从证券价格的历史走势中寻找出规律，再按规律预测未来走势。这些规律包括各种形态、角度、涨跌幅度、涨跌周期、成交量等。从量化投资的角度来说，数量化模型是建立在对历史数据的观测和思考的基础上的，历史数据对未来的投资具有重要的借鉴意义。

2.3.1 K 线简介

K 线又称”阴阳烛”，是反映价格走势的一种图线，其特色在于一个线段内记录了多项信息，相当易读易懂且实用有效，广泛的用于股票、期货、贵金属、数字货币等行情分析。K 线分为阴线和阳线，每一 K 线皆由开盘价、收盘价、最高价和最低价四部分组成，开盘价和收盘价之间的绝对值称为实体，最高价和最低价之间称为影线[19]。实体与影线的区别在于实体较影线粗，影线则会依附在实体的上下两端，影线又可以分成上影线和下影线两种，阳线的上影线表示最高价和收盘价的差距，下影线表示最低价和开盘价的差距；阴线的上影线则表示最高价和开盘价的差距，下影线表示最低价和收盘价的差距。具体的 K 线图如图 2.2 所示。

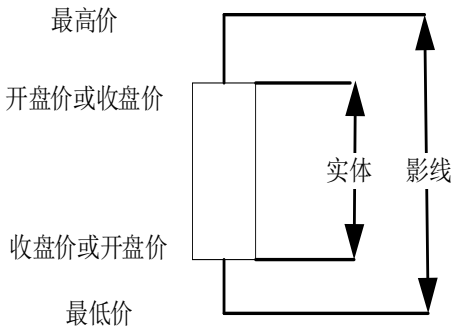


图2.2 K 线图

2.3.2 欧几里得距离

欧几里得距离是一个经常采用的距离定义，指在 m 维空间中两个点的之间的真实距离[20]。计算公式如（2-1）所示。

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \qquad (2\text{-}1)$$

欧式距离虽然在解决实际问题中有着非常广泛的应用，但是对于一些特殊情况，欧式距离存在着明显的缺陷。在基于欧氏距离的时间序列匹配中，要求两个序列等长且在时间轴上完全对齐，对于在形态上非常相似，但是形态特征点（波峰、波谷）不能一一对齐的时间序列而言，相较于动态时间规整算法，其计算结果并不符合用户的直观认识。

2.3.3 动态时间规整算法

动态时间规整算法（Dynamic Time Warping，DTW）是一种衡量两个时间序列之

间相似度的方法，常用于识别两段语音是否表示同一个单词，是语音识别领域最常用的算法之一[21]。DTW 算法主要用来解决形态匹配问题，可以把此算法应用于市场技术分析领域，即定义一种市场形态时，通过动态规划的方式获得两个时间序列的时间对应关系。该算法可以良好的解决寻找相似 K 线的问题。时间序列 X 和 Y 之间的动态规整距离如下所示。

$$D(<>,<>)=0 \tag{2-2}$$

<>表示两个时间序列为空，公式（2-2）表示两个空的时间序列之间的 DTW 距离为 0。

$$D(X,<>)\ =D(<>,Y)=\infty \tag{2-3}$$

公式（2-3）表示一个为空的时间时间序列与另一个非空的时间序列之间的 DTW 距离为无穷。

$$\begin{aligned} D(X,Y) &= D_{base}(head(X),head(Y))+D_{\min} \\ D_{\min} &= \min\left\{D(X,rest(Y)),D(rest(X),Y),D(rest(X),rest(Y))\right\} \end{aligned} \tag{2-4}$$

在公式（2-4）中，head(Y)表示时间序列 Y 的第一个元素，rest(Y)表示时间序列 Y 除第一个元素外的剩余元素，$D_{base}(head(X),head(Y))$ 表示两个元素之间的距离（此处采用欧氏距离）。

2.4 Quartz 作业调度

Quartz 是 OpenSymphony 组织一个受欢迎的开源作业调度框架[22]，并由他们负责研发并进行后期维护。Quartz 高度抽象了任务调度的领域问题，整个调度框架是建立在触发器、任务和调度器这三个核心概念之上[23]。

调度器是整个作业调度框架的核心，主要的功能是负责框架的运行时环境，调度器本身并不能独立的工作，它必须依赖框架内部其他的部件来完成，Quartz 框架是基于多线程的架构，这样在保持框架可伸缩性的同时，又可以保证多个作业的并发运行。应用启动时，调度器负责将任务和触发器进行关联，保障整个应用可以有条不紊的运行。触发器主要用于当设定的任务触发了开发者的设定时间点时，便把达到触发条件的任务交给调度器处理，触发器主要分为简单触发器和可配置触发器两类，开发者可以设定在某个特定的时间节点或者某个时间段内进行任务的调度。任务是可被执行的

作业，本身与调度器和触发器均没有关联，它仅仅是开发者所要实现的业务逻辑，如定时的进行新闻资讯的爬取，股票行情的定时获取等。任务和触发器是属于多对多的关系，多个任务可以被同一个触发器触发，同一个触发器也可以触发多个作业。此种设计方式，极大地增加了 Quartz 作业调度框架的性能，提高了框架的运行效率，保证了应用程序简单、高效的运行。

股票交易系统为了保证股票行情的实时性和新闻热点的时效性，会涉及到很过需要定时处理的业务场景。而 Quartz 作为一款强大的作业调度框架，由于其使用简单、功能强大，性能高效的特点成为本系统调度框架的理想选择。

2.5 WebSocket 长连接

在软件开发中最常使用的是 HTTP 请求，为了保障数据能够可靠、安全的传输，客户端和服务器端使用了“三次握手、四次挥手”的连接释放机制，用户可以通过客户端向服务器端发送一个请求，服务器在接收到请求对数据进行一定的处理返回给客户端。这种请求响应的方式能够满足一般场景下用户的需求，但是在面对实时性高的场景就会显得力不从心，尤其当下用户体验成为每个互联网公司的关注焦点。原有 HTTP 请求的模式在应对高并发、实时性的业务场景有着天然的瓶颈，在此基础上诞生了 WebSocket，它是基于 TCP 之上，在客户端和服务器端建立连接之后，服务器和客户端都能主动地向对方主动推送消息[24][25]，直至被迫中止连接，HTTP 和 WebSocket 的交互图如图 2.3 所示。

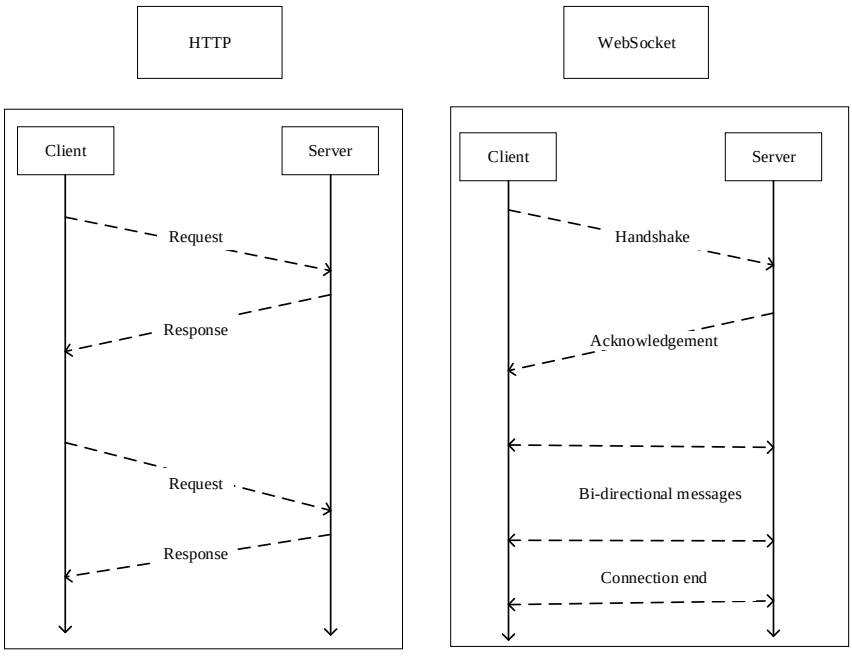


图2.3 HTTP 和 WebSocket 交互图

从图中可以看出，HTTP 请求每次都需要客户端和服务器之间建立连接时才能够进行通信，而 Websocket 是全双工的长连接通信方式，在建立连接之后便可以通过帧序列的形式不断的传输数据，WebSocket 在高并发、实时性的业务场景下能够极大的提升客户端和服务器端的数据交互效率。

2.6 本章小结

本章主要对 Redis 内存数据库、相似 K 线算法、Quartz 作业调度框架以及 WebSocket 长连接等论文涉及到的主要技术进行了简要说明，并简单的叙述了选取的原因。由于股票交易系统对实时性、并发性的要求较高，Redis 的出现在很大程度上弥补了其他关系型数据库的不足，因此选取 Redis 作为系统的缓存存储。Quartz 作业调度框架因为其自身的特性也成为系统完成定时任务处理的不二选择。WebSocket 长连接可以完美的替代传统的 HTTP 请求满足在股票交易的业务场景下对于实时性的要求。动态时间弯曲算法具有概念简单、算法鲁棒等优点成为分析相似 K 线的首选算法。以上技术的相互配合使用为基于 Redis 的股票交易系统的实现打下良好的基础。

第三章 系统需求分析

3.1 总体需求分析

本系统最终是要实现一个性能优良、稳定的股票交易系统。股票交易系统的总体结构图如图 3.1 所示。从整体上系统可以划分为访问端、股票后台交易平台、券商平台和万德行情数据服务提供商四大部分。访问端主要包括面向用户的交易客户端和面向管理员的交易管理端。股票后台交易平台是本系统的核心部分，用于接受用户的请求委托，行情数据查询、风险控制管理、账户管理等，所有的服务均以接口的形式提供。券商平台是系统的第三方平台，主要为系统提供交易接口，系统把用户的委托数据上报给券商平台，经过券商平台撮合交易成功之后，将委托回报给系统。万德行情数据服务提供商主要用于向系统提供股票行情、新闻热点等数据服务，利用 JavaScript 脚本定时的将行情数据存储在 Redis 中，供用户查询使用，极大地提升了行情服务的响应效率。

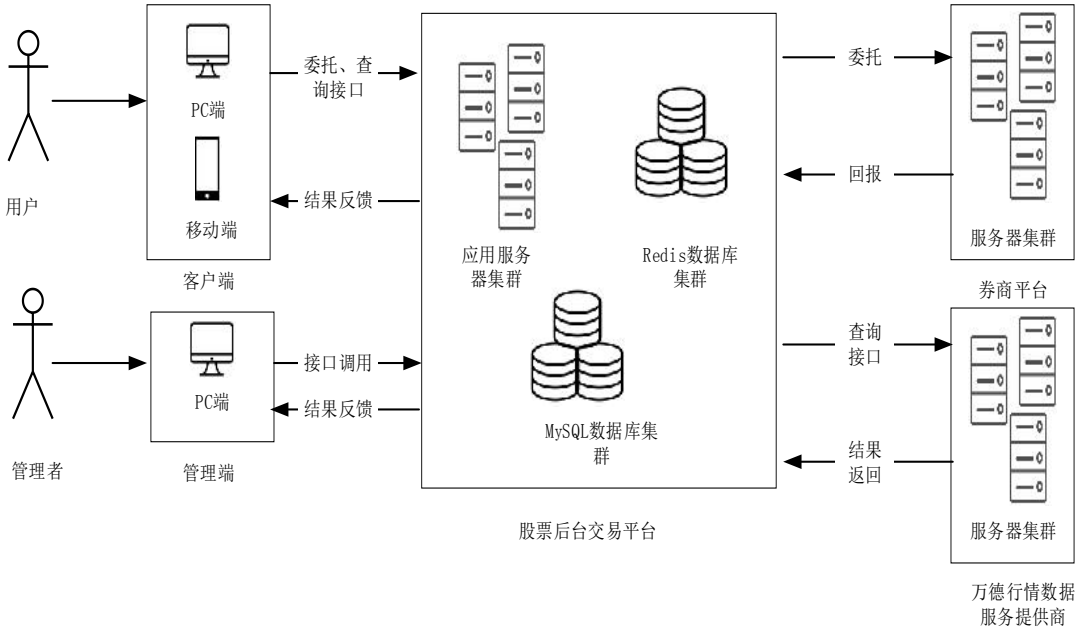


图3.1 系统总体结构图

3.2 系统功能需求分析

3.2.1 行情模块需求分析

行情模块主要是面向用户使用的，其功能是要向注册用户提供准确且完整的股票市场信息，并辅以相似K线功能对股票未来的涨跌幅进行预测供用户参考，筛选出对

用户而言性价比较高的股票。由于用户在行情模块所进行的操作全部都是查询操作，对逻辑接口的访问效率提出了很高的要求。系统采取把万德行情数据服务提供商所提供的行情数据定时存储到Redis中，以此提高系统的访问性能。注册用户在行情模块中可以浏览热点新闻、查看股票行情、查询K线走势、查询分析相似股票K线操作。具体的用例图如图3.2所示。

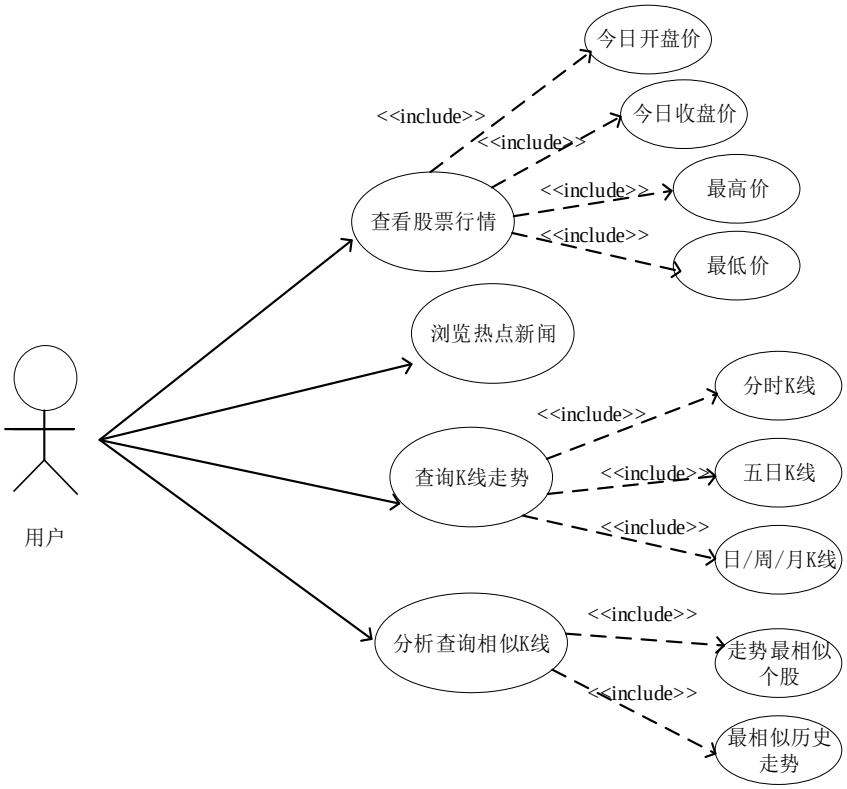


图3.2 行情模块用例图

（1）热点新闻浏览

用户可以点击热门资讯选项卡查看当前的热点新闻列表，列表中的每条新闻均包括新闻标题、新闻来源、以及显示的时间和新闻配图，用户如果对某条新闻感兴趣，可以点击相应的新闻标题查看详细的新闻信息以及新闻中所涉及的股票价格。用户也可以通过下拉页面刷新新闻列表页面查看最新的新闻列表。

（2）股票行情查询

行情的实时查询是系统能够进行正常交易的前提。用户既可以通过搜索按钮查找自己感兴趣的股票信息，只需要在输入框中输入所要查询的股票代码便可以查询到该股票的信息，也可以在自选列表中查看自己收藏的股票信息，点击股票名称便可查看股票的行情信息，包括今日开盘价、昨日收盘价、最高价、最低价等日度信息，还包括股票的实时价格及涨跌幅信息等。

（3）查看K线走势

用户通过该功能可以查看自己所感兴趣股票的K线综合信息，K线图分为分时K线、五日K线、日度K线、周度K线、月度K线五大部分，这是股票交易系统的重要组成部分，也是为用户提供投资信息参考比较基础的部分，用户通过K线走势图可以直观的看到股票在过去一段时间的涨跌幅变化，从而做出自己的投资决定。

（4）相似K线查询分析

股票市场变化极快，对于刚进入炒股领域的用户具有非常大的不确定性，因为专业知识缺乏稍有大意就可能会带来不小的损失。相似K线查询分析功能主要就是面向股票专业知识不扎实的用户，帮助他们在做投资选择时提供一种维度的参考。用户在自己感兴趣的股票详情页面点击相似K线按钮，便可以查看与该股票走势最为相似的个股以及与该股票走势最为相似的历史走势。同时，用户也可以通过相似K线页面的搜索框查看自己感兴趣的股票相似K线分析结果。

3.2.2 交易模块需求分析

交易模块是本系统的核心模块，主要面向注册用户，提供了诸如 PC 端、移动端、APP 端等多端访问方式，极大地丰富了用户进行股票交易时的终端选择。交易模块的主要功能包括：资产查询、买入卖出委托、撤销委托以及委托记录查询，具体的用例图如图 3.3 所示。

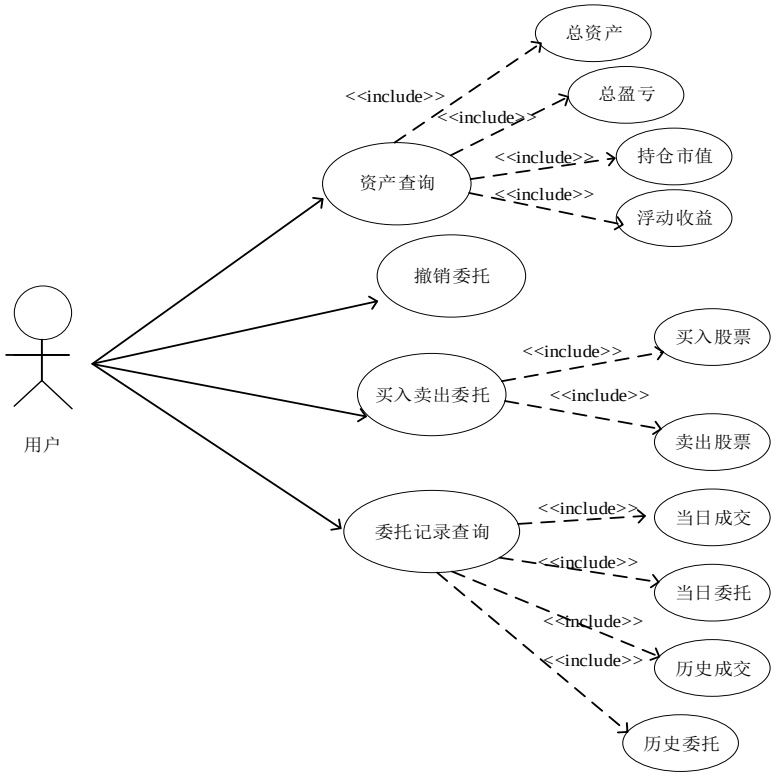


图3.3 交易模块用例图

（1）资产查询

用户点击资产选项卡，可以查看自己当前的资金账户信息，包括总资产、总盈亏、持仓市值、浮动收益、可用余额等信息，系统会根据股票的价格变动以及用户的买入价格自动计算用户的资产总值。除此之外，用户还可以看到自己所持股票列表，显示每只股票的名称/代码、市值/盈亏、持仓/可用、成本/现价。

（2）买入卖出委托

用户通过登录自己的注册账户进行股票的交易，主要是指买入和卖出股票

买入股票：指用户在资金账户充足的情况下，按照自己的投资意愿买入某只股票，其中买入股票的数量必须为 100 的整数倍，不能超过用户总资产所能购买股票的最大数额。

卖出股票：指用户卖出自己所持股票获取收益或者减少损失的行为，卖出股票的数量没有做过多的限制，但是不能超过用户所持股票数量的最大值。

（3）撤销委托

用户在发起委托请求之后，可能会因为某种原因对自己的投资选择不满意，撤销委托功能的存在可以很好的满足用户的此类需求。用户点击撤单选项卡，在撤单页面可以查看自己的委托记录，选中自己想要撤单的委托数据，点击撤单按钮，可以进行撤销委托操作。

（4）委托记录查询

对于已经委托成功的交易，用户可以点击查询选项卡，查询自己不同时间段的委托交易记录，分别为当日成交、当日委托、历史成交、历史委托。

当日成交：用户在交易客户端发起委托请求上报给券商端以后，如果券商平台撮合交易成功，用户买卖成功的股票就会显示在当日成交列表中。

历史成交：历史成交列表会显示用户所有买卖成功的股票列表信息。

当日委托：用户在交易客户端发起买卖股票委托请求，用户所委托的股票信息就会显示在当日委托列表中，委托状态根据券商平台撮合交易成功与否分为未成交和已成交。

历史委托：用户在交易客户端发起所有委托买卖请求的股票信息都会显示在此列表中。

3.2.3 交易管理模块需求分析

交易管理模块是整个交易系统的重要组成部分，主要功能包括：账户管理、风控管理、交易管理、结算管理，具体的用例图如图 3.4 所示。

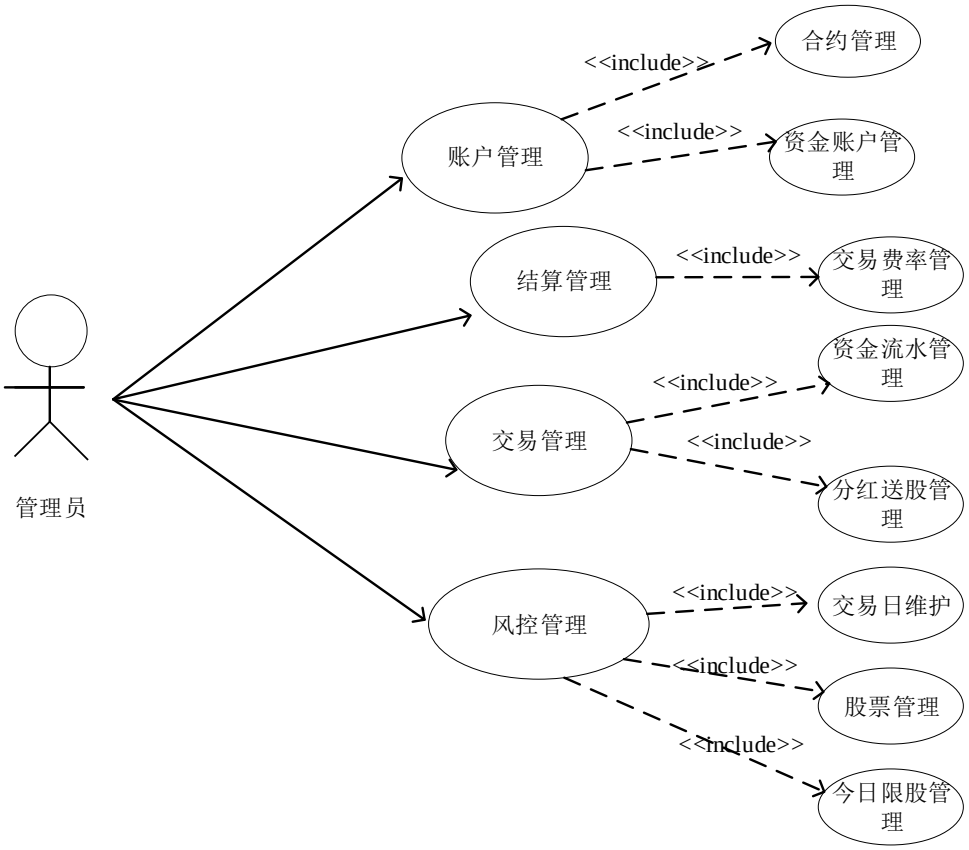


图3.4 交易管理端用例图

（1）账户管理

账户管理是对股票系统中用户的账户信息进行管理，包括涉及股票交易的合约管理和涉及用户资金的资金账户管理。

（2）风控管理

风控管理是对交易过程进行合理的风险控制，从而保障用户的资金安全。主要是由股票管理、今日限股和交易日维护三大部分组成。

股票管理：主要是对系统有关的股票信息进行基本维护，后台系统中用到的所有股票均来自这里，实现对股票相关信息的模糊查询和 CURD 操作。

今日限股：系统会新增高风险股票或者停牌股票，并限制此类股票的交易，确保用户的资金安全，实现对限制股票的 CURD 操作，也可以通过 Excel 批量上传限制股票。

交易日维护：由于每年国家的法定节假日的日期不同，或者由于不可抗力的因素导致证券交易所在某些特定的时间段内无法交易，系统管理员就必须对特定的交易日期进行维护。

（3）交易管理

交易管理是整个交易管理端的核心模块，主要包括资金流水查询管理和分红配股

管理。

分红送股管理：是指上市公司给盈利的股票投资者的一种分红方式。每当上市公司有分红送股时，管理员可以新增一条记录，设置股票信息、分红配股比例以及数量等信息。

资金流水管理：有资金变动的地方就会有流水记录，管理员可以查看今日流水，也可以通过设置起止时间查看历史流水。

（4）结算管理

股票交易系统每天都会涉及到大量的用户交易，如何保障交易的准确性就需要对交易金额计算所涉及的交易费率进行管理，由于交易所的不同，在交易的过程中就会涉及到不同的交易费率，系统管理员可以通过新增、删除、修改等按钮对交易费率信息进行维护。

3.3 非功能性需求分析

本论文所要设计的股票交易系统是一款以交易为主的股票平台。注册用户在客户端可以浏览热点新闻、查询资产余额、发起买入卖出委托等，系统管理员在管理端可以对风控、账户、交易等各个模块进行管理，系统后台系统根据接收到的不同请求进行相应的业务处理。由于系统所涉及的所有业务都和股票息息相关，与股票相关的行情信息也会不停的刷新，为了让用户获取股票信息的时间与证券交易所同步，对系统在数据的实时性上提出了较高的要求。此外，由于国内大多数股民都比较倾向于短线炒股，伴随而来的是大量的快速频繁交易，因此在满足数据实时性的基础上，系统的并发性也是需要重点关注的焦点。

（1）实时性

时刻变化是股票行情的常态，由于股票市场瞬息万变，股票行情信息也会以秒为单位不断地刷新，注册用户能否及时准确地获取股票行情信息对于用户是否能够委托成功起着至关重要的作用。如果系统不能提供实时的股票行情信息，导致用户获取行情信息滞后，就会带来委托不能及时成交、大量的委托交易挤压等严重后果，给用户和公司造成经济损失。因此，保证行情数据的实时性是最为关键的一部分。

（2）并发性

任何与金融领域相关的系统都应该是以稳定性为前提，随着系统用户量的逐渐增多，在特定的时段可能会遇到大规模的交易请求，系统必须具备能够短时间处理大量请求的能力，并保证每一位用户具备良好的使用体验。因此，并发性也是系统设计的关键所在。

针对以上需求，提出以下性能指标：

接口延时：系统能够为用户提供高效的服务，所提供的接口延时不能高于 200ms。

接口并发：系统要有应对高并发的压力，鉴于本系统上线时间较短，用户量离预期还有一定的差距，因此单机必须保障每秒能够处理 300 次以上的并发访问。

3.4 本章小结

本章对基于 Redis 的股票交易系统进行了详尽的需求分析。

（1）在总体需求分析一节，结合系统总体结构图对系统的各个组成部分的功能进行分析。

（2）在系统功能需求分析一节，介绍了交易系统的参与者，并结合用例图，对核心部分进行了功能需求分析。

（3）在非功能需求分析一节，对系统的使用场景进行充分的调研，从实时性、并发性两大方面总结出系统需要满足的非功能性需求。

本章从业务流程、业务需求分析以及性能等多个角度对系统的需求进行了阐述，为后续系统的设计与实现打下坚实的基础。

第四章 系统设计与实现

通过第三章对股票交易系统在功能和性能上的分析了解，本章将对系统的整体架构、数据库、功能模块进行详细的设计，并根据模块的不同，分别介绍了数据库的设计和类的设计，在此基础上完成系统的整体实现。

4.1 系统架构设计

股票交易系统的架构设计主要由三部分构成：业务层、逻辑层、数据层。业务层根据用户的请求向逻辑层发起接口请求，逻辑层根据请求的不同对数据层的数据进行相应的逻辑处理，最终把处理结果返回给业务层展示。具体架构图如图 4.1 所示。

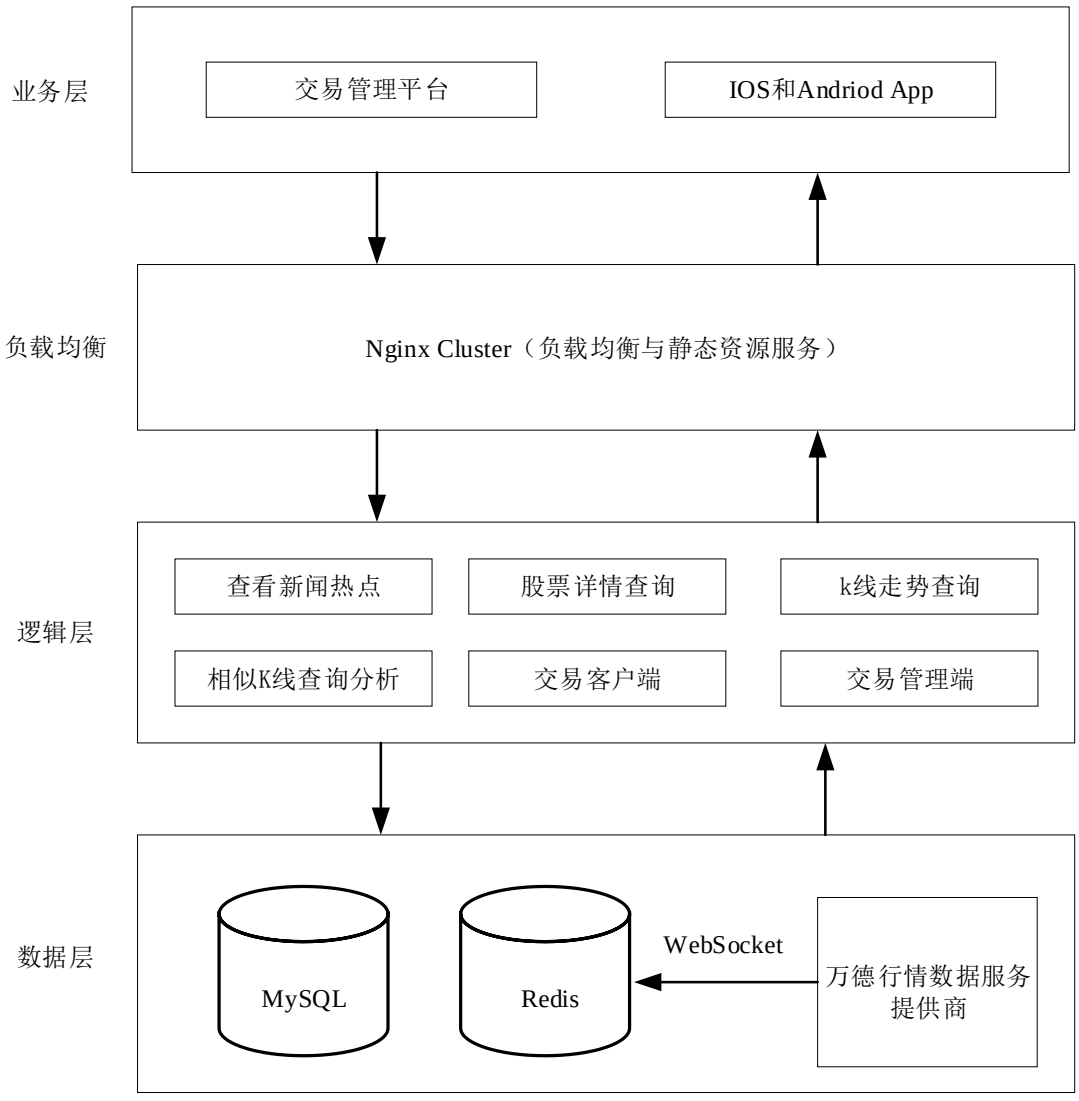


图4.1 系统架构图

业务层：系统可以通过业务层进行具体的业务展示。系统的视图层主要分为 Web 端和移动端，其中 Web 端主要用于系统交易管理平台，此平台主要面向公司管理员使用，对交易过程中所涉及的信息进行管理，从而保障用户的资金安全。而移动端主要是 APP 端，面向广大注册用户，用户通过 APP 可以查看股票行情、查询分析相似 K 线、股票买卖等功能。

无论系统业务的展示方式有几种，从本质上来说它们均属于 MVC 中的视图层，用于展示数据与用户进行交互。Web 端主要使用超文本标记语言（HyperText Markup Language，HTML）和 JavaScript 实现，移动端根据手机系统的不同分别用 Java 和 Objective-C 来实现。用于展示的数据主要是逻辑层通过接口的方式提供，界面开发主要是由专门的客户端开发人员完成。

逻辑层：系统的核心部分是逻辑层，功能是提供对外接口供业务层调用。由于后期随着用户数量的增多，系统面临着很大的并发压力，采用 Nginx 负载均衡系统能够很好地提升系统的性能。这些对外接口均采用了 MVC 的设计思想，接口的设计使用了模型层和控制层两个部分。控制层负责对用户从业务层传来的请求参数进行验证处理，待验证通过以后把数据传递给模型层，模型层对数据进行逻辑处理以后把处理结果返回给业务层。模型层是系统中与数据进行交互的部分，主要根据用户的请求对缓存、数据库中的数据进行逻辑处理，是系统中直接处理数据的部分，同时也是系统中最为重要的一层。

数据层：数据层主要由 Redis 和 MySQL 存储两部分组成。

本系统为了提高逻辑层接口的访问速度，提升用户体验，采用 Redis 集群作为缓存系统。缓存系统的主要目的是为了提升本系统中高频数据和热点数据的访问效率[26]。在股票交易系统中，由于用户会经常性地访问股票行情信息，浏览热点新闻等热点数据，同时考虑到本系统用户基数庞大，还必须要考虑到带来的高并发的问题。采用 Redis 对数据进行缓存，用户在访问的时候可以直接读取缓存，由于 Redis 是内存数据库，所有的数据都是存储在内存当中，数据的读取效率相当迅速，可以有效地应对高并发访问，提升系统整体的访问性能[27]。

另外，系统采用 MySQL 进行数据存储，由于单一的数据库在面对日益增长的数据量时很容易就会出现存储瓶颈问题，与此同时，基于交易模块的特性，用户进行读操作的频率要远远大于写操作的频率，所以系统采用两台数据库进行读写分离，一台作为主数据库，用来进行数据的写操作，另外一台作为从数据库，用来数据的读操作[28]，两者互不影响，主库出现故障并不会影响从库的读操作，从库出现故障也并不会影响主库的写操作，这样可以有效地减轻主数据库的访问压力，避免单点故障提升系统整体的稳定性。

随着用户量的增多，与之相关的交易数据也随之增长，系统会面临行情交易数据

在分布式结构中的存储以及一致性问题。在行情模块中，由于所有的行情数据均是只读的，不存在用户写入的情况，因此系统只需要定时的把万德行情数据服务提供商的行情数据存储在 Redis 中，供用户访问调用即可。在交易模块中，由于会涉及到用户登录，持仓市值的计算以及委托上报和成交回报等操作，系统采取把 Session 和常量计算数据存储在 Redis 中，以提高系统的响应速率，数据对象如委托数据、成交汇报数据则存储在数据库中。在交易管理模块中，为了避免缓存中出现脏数据，对数据的每一次操作都会先存储在数据库中，待数据写入成功后，再把数据写入到缓存中，以保障数据的一致性。

4.2 系统功能结构设计

本小节旨在对股票交易系统的各个模块功能进行细化分析。通过上一章节对系统的需求分析，系统可以为用户提供股票行情的查询以及股票交易等功能，管理员可以对用户的交易信息进行管理，因此系统主要是由行情模块、交易模块和交易管理模块三大模块组成。

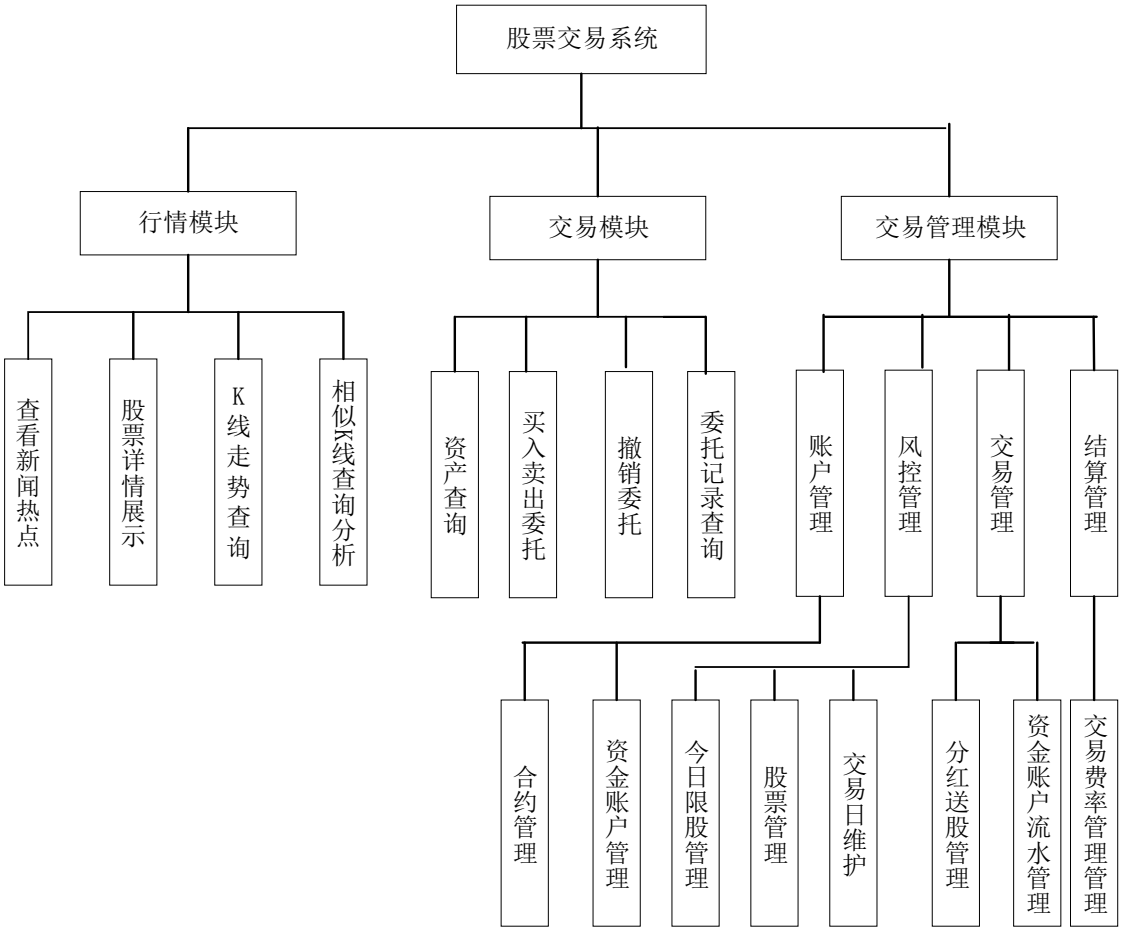


图4.2 系统功能模块设计图

如图 4.2 所示，当用户在使用该股票软件时，系统要实现行情模块、交易模块和交易管理模块三大部分，每个模块下面又包含了详尽的子功能。用户在行情模块能够浏览新闻热点、查看股票详情、查询 K 线走势以及查询分析相似 K 线操作。用户在交易模块可以供注册用户进行资产查询、买入卖出委托、委托记录查询以及撤销委托操作。交易管理模块是供系统管理员对账户、交易、风控以及结算进行管理。以上各个模块共同构成了股票交易系统。

4.3 系统数据库设计

本系统的数据库采用的是 MySQL，基于其体积小、成本低以及速度快等特性[29]，是目前最为受欢迎的关系型数据库，具备非常稳定的性能，能够满足公司对数据库的大部分需求。由于后期股票交易系统的注册用户会逐渐增多，单一的数据库很容易就遇到性能瓶颈，因此采用主从配置、读写分离的方式对数据库进行管理[30]，基于对股票交易系统的需求分析，设计出具备良好的读取速度和高容灾性的数据库对系统的性能提升有不可忽视的作用。

4.3.1 实体关系图设计

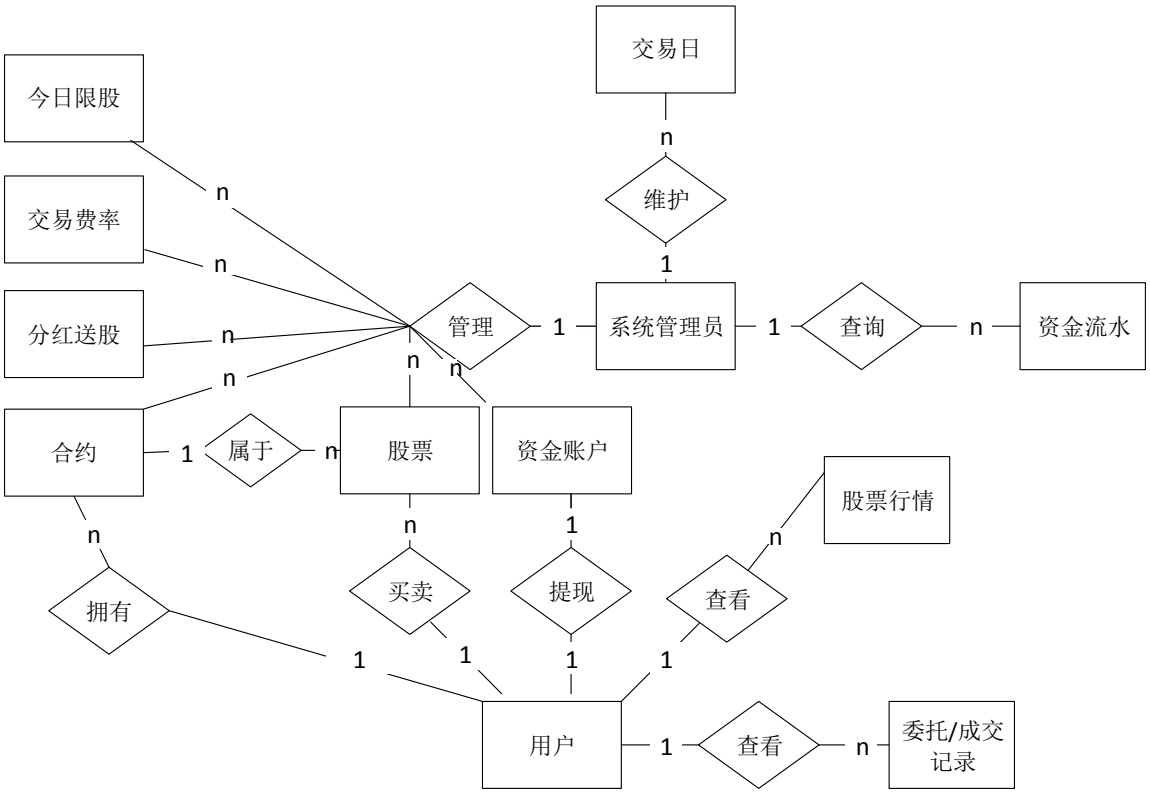


图4.3 实体关系图

如图 4.3 所示，系统中有合约、股票、资金账户等多个实体，图中已经表明了各个实体之间的联系，实体的属性将会在数据库表的设计中体现出来。用户与合约是一对多的关系，一个用户可以拥有多个合约，用户对股票的买卖均是基于合约进行交易，用户拥有的合约可以进行多只股票的买卖。每个用户都拥有自己独立的资金账户，用户与委托/成交记录是一对多的关系，每个用户都可以查看当日或者历史的所有委托成交记录。由于系统管理员是对系统所有的用户信息和股票信息进行管理，所以管理员与其相互联系的实体均是一对多的关系。

4.3.2 数据库表设计

由于系统中的数据库表结构规模过于庞大，本论文只列举部分主要的数据库表，具体的表结构如下所示。

（1）股票市场详情表：用于存储股票的市场买卖信息等，详情如表4.1所示。

表4.1 股票市场详情表

字段名	数据类型	非空	默认值	备注
Stk_Cd	varchar	否	-	股票代码（主键）
Stk_Nm	varchar	否	-	股票名称
Cur_Prc	decimal(18,2)	否	0	最新价格
By_Prc_1	decimal(18,2)	否	0	买价1
By_Prc_2	decimal(18,2)	否	0	买价2
By_Qty_1	int	是	-	买数量1
By_Qty_2	int	是	-	买数量2
Sel_Prc_1	decimal(18,2)	否	0	卖价1
Sel_Prc_2	decimal(18,2)	否	0	卖价2
Sel_Qty_1	int	是	-	卖数量1
Sel_Qty_2	int	是	-	卖数量2
Trans_Qty	int	是	-	成交数量
Opn_Prc	decimal(18,2)	否	0	开盘价
Yesterday_Cls_Prc	decimal(18,2)	否	0	昨日收盘价
Td_Hst_Prc	decimal(18,2)	否	0	当日最高价
Td_Lst_Prc	decimal(18,2)	否	0	当日最低价
LmtUp_Prc	decimal(18,2)	否	0	涨停价
LmtDn_Prc	decimal(18,2)	否	0	跌停价

（2）交易日表：用于存储交易日信息，详情如表4.2所示。

表4.2　交易日表

字段名	数据类型	非空	默认值	备注
Hldy	date	否	–	节假日
Nxt_Trd_Dy	date	否	–	后续交易日
Lst_Upd_Tm	datetime	是	–	最后修改时间
Lst_Upd_Psn	bigint	是	–	最后修改人

（3）股票表：用于存储股票信息，详情如表4.3所示。

表4.3　股票表

字段名	数据类型	非空	默认值	备注
Stk_Cd	varchar(20)	是	-	股票代码（主键）
Stk_Nm	varchar(20)	是	-	股票名称
Exch_Id	smallint	否	-	交易所编号
Lst_Upd_Tm	datetime	否	-	最后修改时间
Lst_Upd_Psn	bigint	否	-	最后修改人

（4）交易费率表：用于存储交易时的费率数据，详情如表4.4所示。

表4.4　交易费率表

字段名	数据类型	非空	默认值	备注
Trd_Fee_Rt_Id	bigint	否	–	交易费率编号（主键）
Exch_Id	smallint	是	–	交易所编号
By_Stamp_Duty_Pct	decimal(9,6)	是	–	买入印花税比例
Sel_Stamp_Duty_Pct	decimal(9,6)	是	–	卖出印花税比例
By_Trnfr_Fee_Pct	decimal(9,6)	是	–	买入过户费比例
Sel_Trnfr_Fee_Pct	decimal(9,6)	是	–	卖出过户费比例
By_Lst_Trnfr_Fee	decimal(18,2)	否	0	买入最低过户费
Sel_Lst_Trnfr_Fee	decimal(18,2)	否	0	卖出最低过户费
Lst_Upd_Psn	bigint	是	–	最后修改人

（5）合约表：主要用于存储用户持仓相关信息，详情如表4.5所示。

表4.5 合约表

字段名	数据类型	非空	默认值	备注
Cntr_Pos_Id	bigint	否	-	合约持仓编号（主键）
Usr_Id	bigint	否	-	用户编号
Usr_Nm	varchar(20)	否	-	用户名称
Cap_Acct_Id	bigint	否	-	资金账户编号
Stk_Cd	varchar(20)	否	-	股票代码
Stk_Nm	varchar(20)	否	-	股票名称
Sec_Qty	int	是	-	证券数量
Aval_Sel_Qty	int	是	-	可卖数量
Cst_Prc	decimal(18,2)	否	0	成本价
Flot_PL_Amt	decimal(18,2)	否	0	浮动盈亏金额
PL_Pct	decimal(9,6)	是	-	盈亏比例
Cur_Mkt_Val_Amt	decimal(18,2)	否	0	最新市值金额
Cur_Prc	decimal(18,2)	否	0	当前价
Td_By_Qty	int	是	-	今买数量
Td_Sel_Qty	int	是	-	今卖数量
Cst_Amt	decimal(18,2)	否	0	成本金额
Shrhdr_Cd	varchar(20)	是	-	股东代码
Cmt	varchar(1000)	是	-	备注
Lst_Upd_Tm	datetime	是	-	最后修改时间

（6）历史成交单表：用于存储用户历史成交的股票信息。详情如表4.6所示。

表4.6 历史成交单表

字段名	数据类型	非空	默认值	备注
Trans_Id	bigint	否	-	成交编号（主键）
Trans_Cd	varchar(20)	是	-	成交代码
Usr_Nm	varchar(20)	是	-	用户名称

表 4.6（续） 历史成交单表

Cap_Acct_Id	bigint	是	-	资金账户编号
Stk_Cd	varchar(20)	是	-	股票代码
Stk_Nm	varchar(20)	是	-	股票名称
Delgt_Id	bigint	是	-	委托编号
Delgt_Seq	tinyint	是	-	委托序号
By_Sel_Flg_Id	smallint	是	-	买卖标志编号
Trans_Prc	decimal(18,2)	否	0	成交价格
Trans_Qty	int	是	-	成交数量
Trans_Amt	decimal(18,2)	否	0	成交金额
Trans_Tm	datetime	是	-	成交时间
Stamp_Duty_Amt	decimal(18,2)	否	0	印花税金额
Trnfr_Fee_Amt	decimal(18,2)	否	0	过户费金额
Stl_Fee_Amt	decimal(18,2)	否	0	结算费金额

4.4 行情模块的实现

行情模块是面向用户的，凡是下载 APP 的用户均可以在未登陆的状态下浏览新闻热点、查询股票详情、查看股票 K 线以及分析查询相似 K 线，为用户的投资选择提供多种维度的帮助。整个行情模块的数据落地实现借鉴了软件设计模式中的适配器模式的思想，利用 JavaScript 脚本预先将万德行情数据服务提供商的相关数据进行一定的整合处理定时地存储在 Redis 中，通过访问存储在 Redis 中的缓存数据完成查询功能。新闻热点、股票详情、K 线走势的查询功能比较类似，这里以查看股票 K 线功能为例进行说明。

4.4.1 查看股票 K 线

具体的流程图如图 4.4 所示，进入到行情页面后，用户可以通过点击自己所收藏的股票列表查看自己所关心股票的 K 线走势，请求发出后，后台系统会先从 Redis 中查询最近两天的股票详情信息，并判断用户请求的当日是否是交易日（交易日为周一到周五，法定节假日除外[31]），如果是非交易日，系统会首先获取离当前日期最近的交易日期，并判断从 Redis 中获取股票详情的最新日期是否与最近的交易日期相同，相同即说明缓存命中，直接从 Redis 中获取用户所需的 K 线数据，并利用渲染引擎进行 K 线的绘制，否则向万德行情数据服务提供商发起请求，获取用户所需的数据并存储在 Redis 中供下次使用。若是交易日，系统便会对当前时间是否在交易时间（股

票的交易时间为周一到周五的上午 9:30-11:30，下午 13:00-15:00[30]）进行再次判断，如果用户请求的时间处于交易时间段之内，系统会获取到当前时间的前一天日期并与从 Redis 中获取的最新日期进行比对，若处于交易时间段之外，系统则会将当前日期与 Redis 中获取的日期进行比较，缓存命中与否的处理方式与非交易日的处理方式一致。至此用户对股票 K 线的查询过程结束。

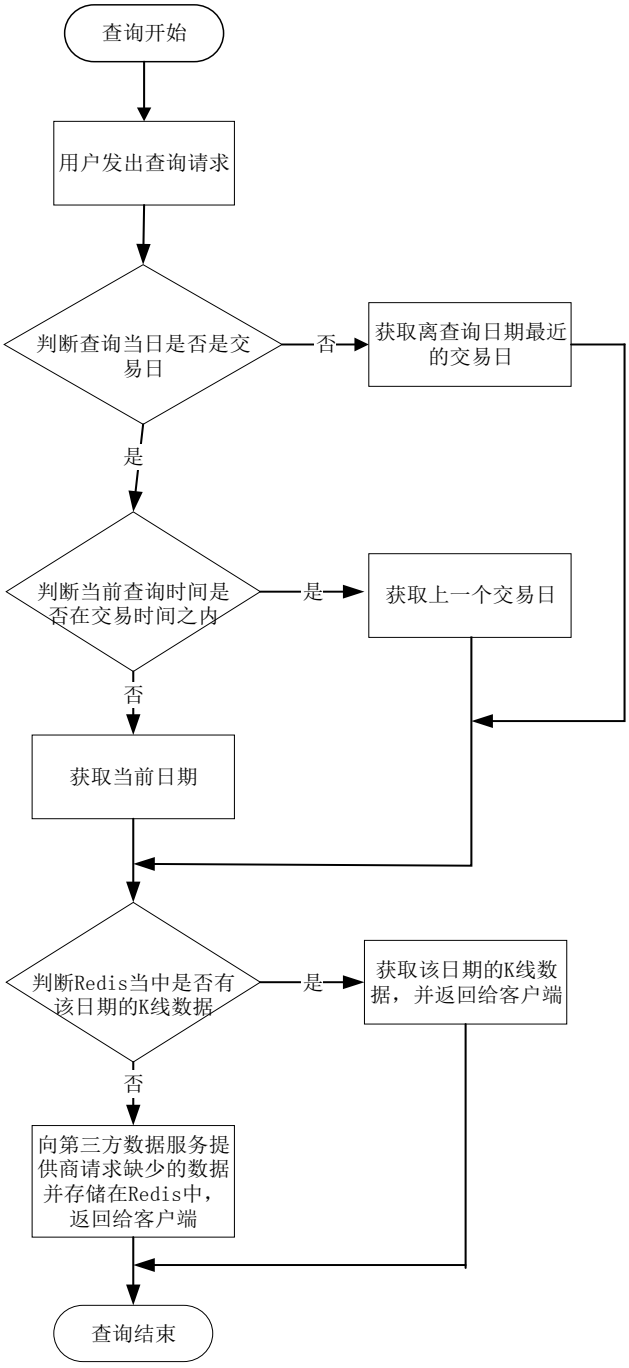


图4.4 查询股票 K 线流程图

4.4.2 相似 K 线查询分析

在对股票走势的预测上，可以从历史数据库中寻找同当前股票状态相似的历史片段，根据这些历史片段此后的股票走势，判断当前股票的未来走势，相似 K 线查询分析就是针对预测股票未来走势开发出的一种功能。利用 DTW 算法动态规划的特点计算两个时间序列的对应关系，系统通过对数据处理过程的改进，以期达到更好的预测效果。

（1）数据选择和预处理

在股票分析中，影响股票走势的因子有很多，如成交量、股票收盘价、换手率等，如果只选取股票价格单变量的时间序列作股票状态预测，一方面会因为选取因子的单一化造成预测结果的偏差，另一方面由于不同的股票价格之间差异范围较大，也会对结果造成一定的影响。由于成交量和收盘价之间是相互影响和验证的，两者在反映用户对股票的投资选择上具有相似的权重。因此本系统选取由成交量和收盘价格组成的多变量时间序列对股票的状态进行描述。假设第 t 个交易日的观测样本用收盘价格和日成交额表示为 $x_t=(p(t),v(t))$，L 个交易日的收盘价格和日成交序列表示为 $X_L=\{x_{t-L+1},...,x_{t-1},x_t\}$。为了消除行情价格和成交量序列的非平稳性造成的偏差，对观测样本 $x_t=(p(t),v(t))$ 中的成交量均要除以最新的收盘价格 $p(t)$。

（2）DTW 交易策略实现过程

DTW 算法在对样本序列进行计算分析时，需要通过模式识别对持仓至下一个交易日的收益率 $r_{t+1}=p(t+1)/p(t)-1$ 进行估计，以决定当日收盘时的建仓方向。具体的实现过程如下。

首先，采用长度为 L 的移动窗口，将历史的行情划分为不同的行情片段，每一个片段为 L 个交易日的行情时间序列，片段划分示意图如图 4.5 所示。

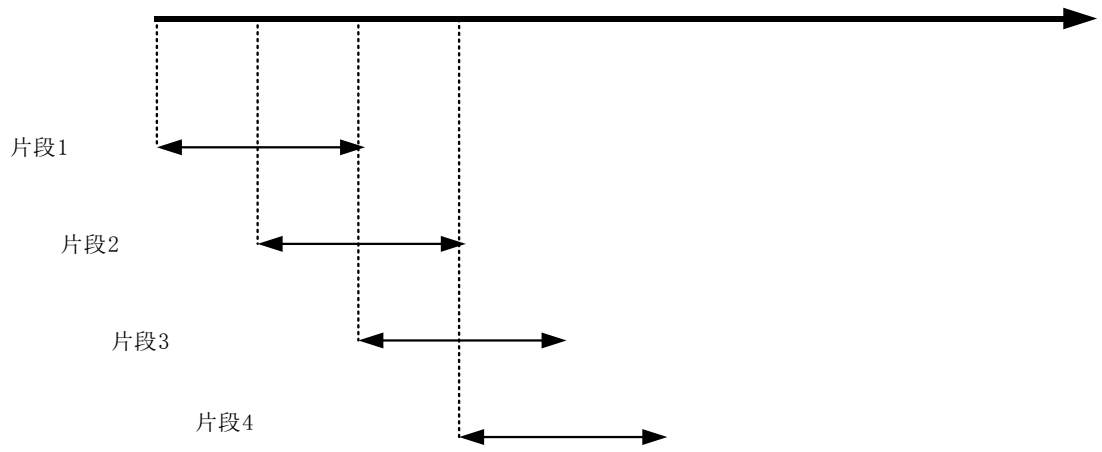


图4.5 历史行情的片段划分示意图

然后，通过动态规整计算公式（4-1）在历史样本中寻找与 X_L 距离最小的前 n 个序列，其中 $D_{base}(head(X),head(Y))$ 用欧式距离进行计算。

$$\begin{aligned} D(X,Y) &= D_{base}(head(X),head(Y)) + D_{\min} \\ D_{\min} &= \min\{D(X,rest(Y)), D(rest(X),Y), D(rest(X),rest(Y))\} \end{aligned} \tag{4-1}$$

最后，对计算出的 n 个动态时间规整距离按照数据的大小进行排序，数据值最小的便是所要查询的相似 K 线。

考虑到用户使用相似 K 线的最终目的是为了对交易次日的股票收益进行预测，因此，系统对所得结果进行了进一步的改进处理。查找计算得到的 n 个序列对应的交易次日的收益率，记为 $r_1,r_2,r_3,...,r_n$，则可获得对样本 K 线交易次日的收益率 $\hat{r}_{t+1}$ 的估计公式如（4-2）所示，其中 w_i 如公式（4-3）所示。

$$\hat{r}_{t+1} = \frac{w_1r_1 + w_2r_2 + ... + w_nr_n}{w_1 + w_2 + ... + w_n} = \frac{\sum_{i=1}^{n} w_ir_i}{\sum_{i=1}^{n} w_i} \tag{4-2}$$

$$w_i = \frac{1}{D_i} \tag{4-3}$$

这种加权估计的方式使得距离小的历史样本在预测中占有比较大的权重，因此，能够减小 n 的取值对预测结果的影响。

相似 K 线分析界面如图 4.6 所示。

图4.6 相似 K 线分析界面

4.5 交易模块的实现

交易模块给注册用户提供了可交易的平台，用户可以在此模块买卖委托、查询撤销委托以及查询资产等，下面将通过类图、序列图的方式对此模块的功能实现进行详细地介绍。

4.5.1 买入卖出委托

股票交易市场上常见的交易类型包括普通买入股票、市价买入股票、普通卖出股票、市价卖出股票四种类型，普通买入和市价买入的区别在于普通买入必须在买入指令中规定买入的价格和数量，系统会根据价格优先和时间优先的原则在卖出委托中来选择对用户有利的来成交，此时的成交价格不会高于指定的价格；而市价买入委托指的是即时地在卖出委托中选择最低的价格成交，此时的买入价格是当时的股票交易市场价格。卖出和市价卖出同理，本系统只实现普通买入卖出功能，具体的序列图如图4.7 所示。

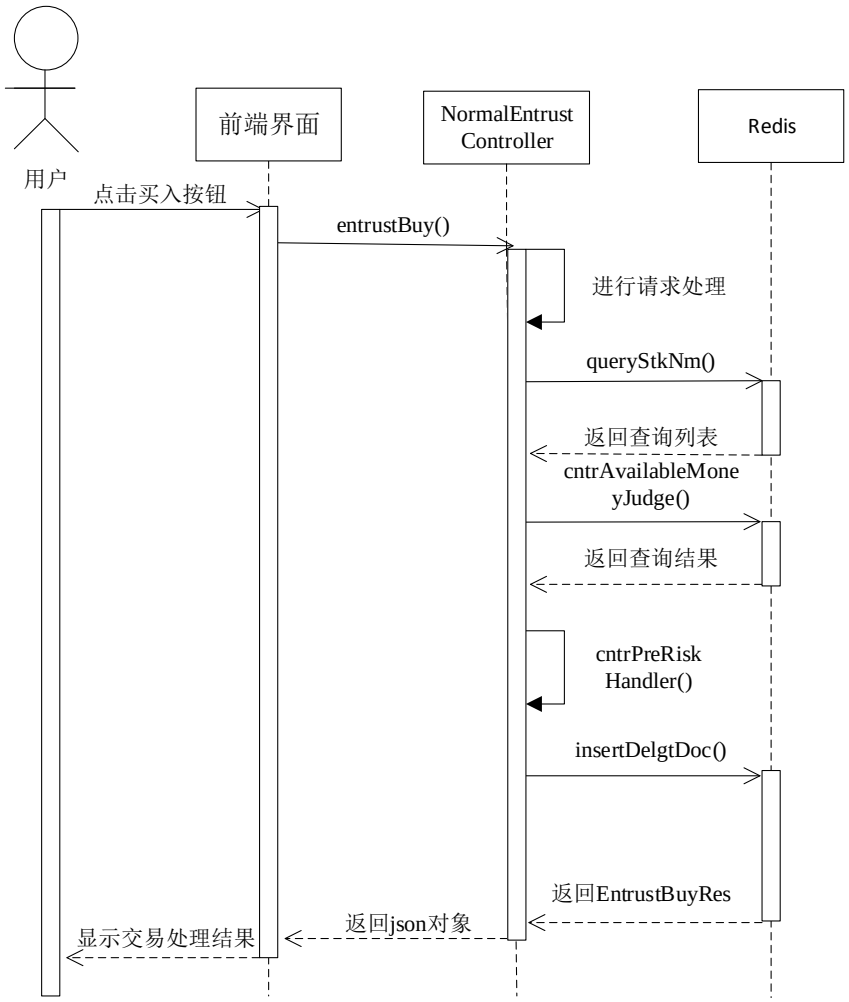


图4.7 用户市价买入序列图

如图 4.7 所示，注册用户在买入界面中指定要买入的股票价格以及股票数量，点击买入按钮将股票委托数据通过 http 请求提交至交易客户端，后台系统会首先通过 queryStkNm 方法查询要买入的股票信息，然后会判断其提交的价格参数是否位于跌停价格和涨停价格之间，所要委托的股票数量是否是 100 的整数倍，如果上述的委托数据均符合系统的要求，接下来便会读取 Redis 服务器中的印花税系数、过户费系数以及佣金系数，计算此次委托的最终交易金额通过 cntrPreRiskHandler 方法进行合约的风控处理，比如单只股票持仓比例、交易日判断、今日限股判断等，风控的失败将会直接导致此次委托失败，风控成功后会扣除相关的金额并把变动后的数据通过 insertDelgtDoc 方法存储在 Redis 中，通过 json 对象的形式向客户端返回委托成功的提示信息。整个后台系统采用的是开源的 Spring MVC 框架，能够有效地降低代码之间的耦合，方便进行开发维护。

系统买入功能实现类图如图 4.8 所示，类图说明如下。

NormalEntrustController 继承自 BaseJsonController，提供买入委托的接口，entrustBuy方法完成普通买入委托的请求，首先会对用户发送的委托数据进行合法验证，然后再调用BuyEntrustService里面的entrustBuyDeal方法，完成买入委托接口的逻辑实现。

NormalEntrustController还提供了很多查询接口，比如queryStkNm方法用于根据股票代码查询股票名称，通过调用StockInfoService里面的findStkByStkCd方法在数据库中查询股票名称。queryCntrAva方法根据合约编号查询合约信息，查询到的结果均会以JSON格式返回给页面。

BuyEntrustService类提供买入功能的逻辑实现，entrustBuyDeal方法用于普通委托买入的逻辑实现，对用户的委托数据进行合法验证之后，会判断用户的合约可用金额，如果可用金额大于所要买入股票金额，便会划转相应的资金并进行风控处理，生成一条委托记录，同时生成一条委托成功日志流水记录，否则返回委托失败结果。judegeStockIsBuy方法用于判断用户所委托股票是否可允许买入。cntrPreRiskHandler方法用于进行交易事前风控处理。updateCapAccountMoney方法用于根据用户的股票委托情况修改数据库中的资金账户信息。insertDelgtDoc方法用于在委托记录表中生成一条委托记录。managContractPosJudge方法用于对合约中的持仓股票进行判断。

EntrustRes类用于将处理结果进行实例化，方便将用户的查询结果JSON化处理返回给客户端。

LogService提供的是日志记录服务，通过logFlwTypCd日志类型、logFlwLvlId日志级别、time日志记录时间查询特定时间段的日志信息。系统中的每个功能实现均有日志记录，后续类图说明将不再赘述。

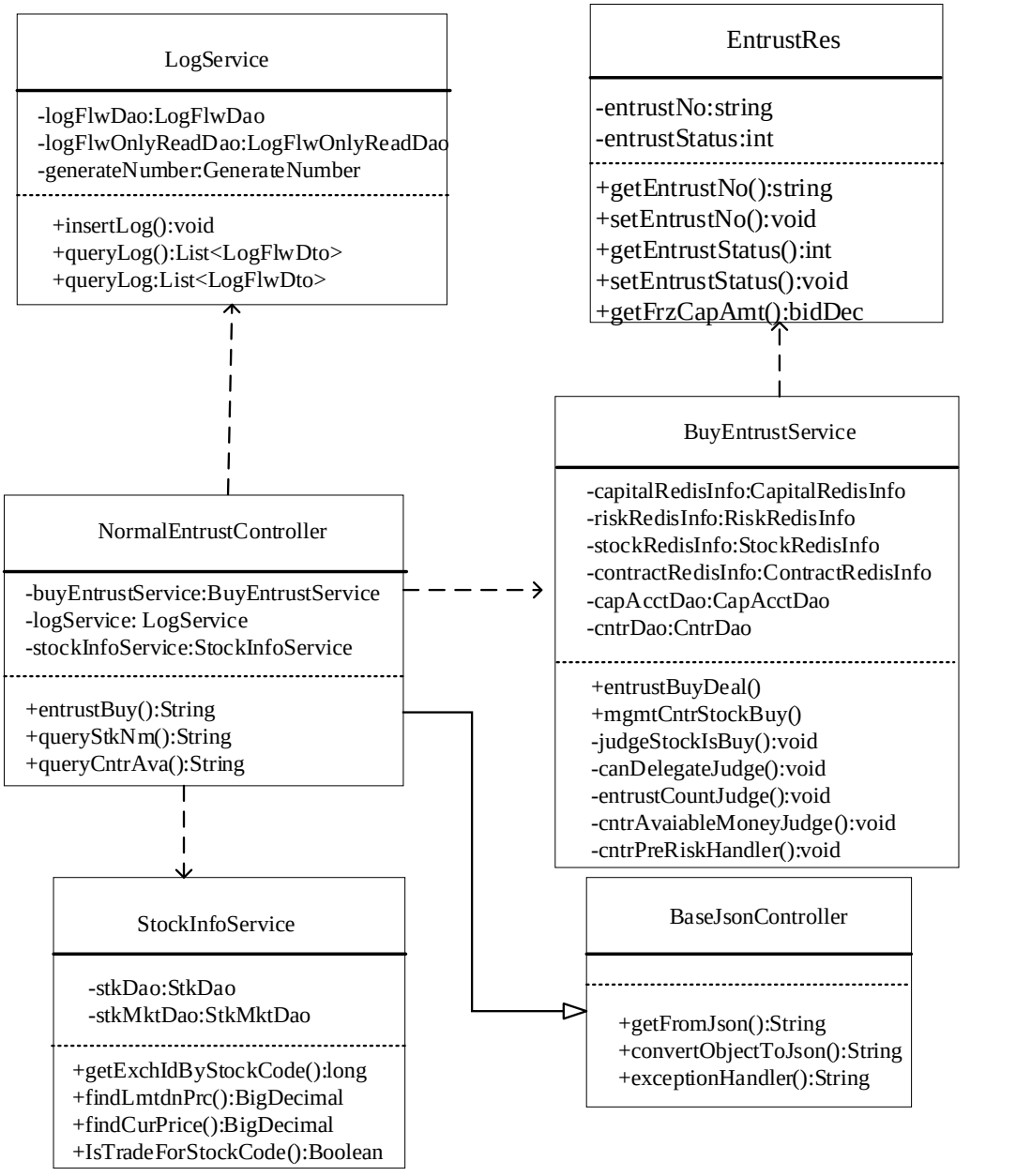


图4.8 买入功能实现类图

卖出委托的序列图如图 4.9 所示。注册用户在卖出界面中指定要卖出的股票价格以及股票数量，点击卖出按钮通过 entrustSell 将股票委托数据提交至交易后台系统，后台系统会首先读取 Redis 中卖出常量（即是否在交易日内的交易时间）通过 checkNum 方法判断用户所要委托的股票是可否可以卖出，然后再会判断其价格是否位于跌停价格和涨停价格之间，如果上述的委托数据均符合系统的规定要求，便会通过 canDelegateJudge 方法判断用户可持仓数量是否可以满足，如果满足便会生成委托数据并把变动后的数据存储在 Redis 中，并以 json 对象的方式向客户端返回委托成功的提示信息。

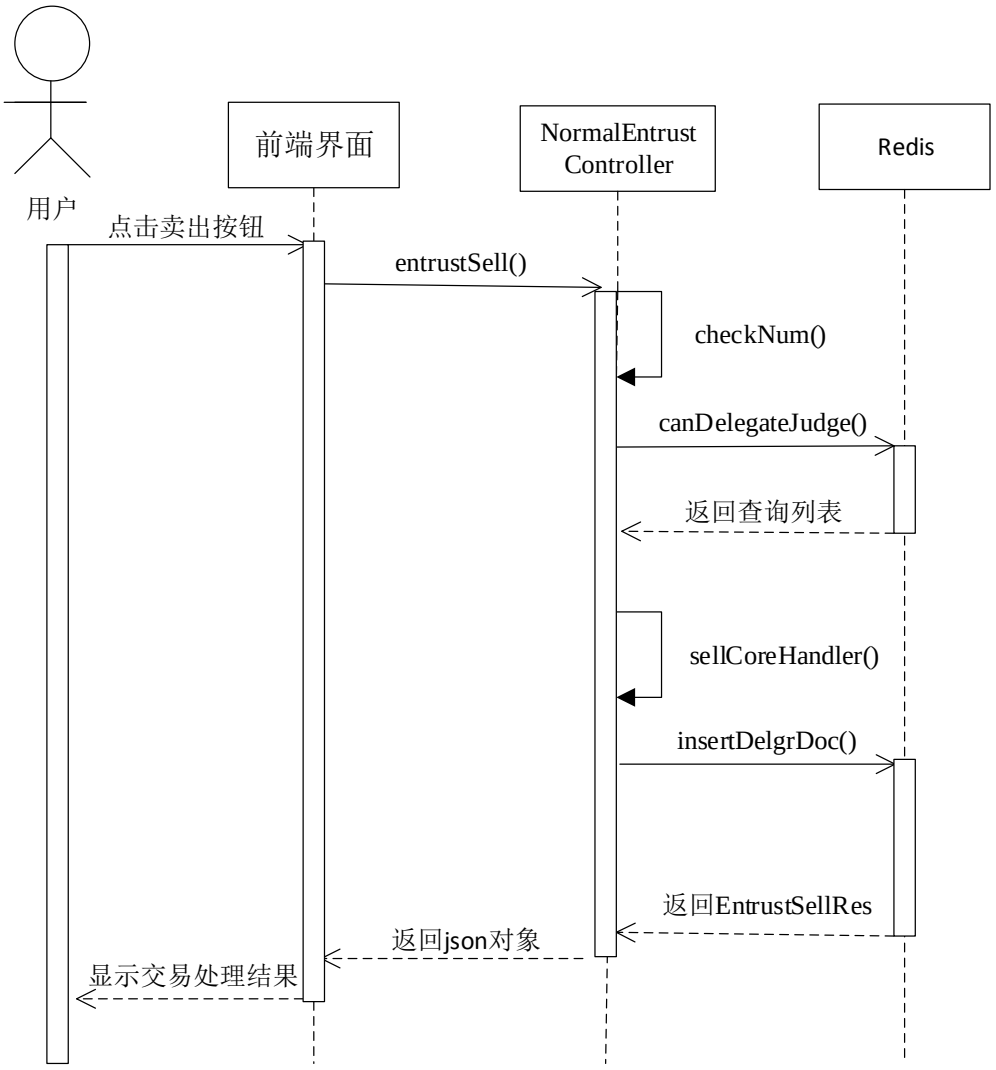


图4.9 卖出功能实序列图

用户在发出买入卖出交易委托请求之后，用户的委托数据便会存储在系统的Redis 服务器中，然后系统会对接券商系统进行委托上报和成交回报，委托上报是指把用户的委托请求上报给券商系统，成交回报是指券商系统在撮合交易成功之后，会把成交结果数据返回给系统。整个交易过程中的用户委托的数量会随着用户请求不断增大，如何及时地委托上报和成交回报便成为影响系统的瓶颈所在。在此本系统采用Quartz 的调度 Job，根据预先配置的定时表达式，每秒读取 Redis 中的委托数据提交给券商系统，当用户量增大导致 Redis 中未报的委托数据增多时，可在 Job 内开启多线程不断地把委托数据提交给券商端，提高委托上报的效率，防止委托数据积压。整个过程中委托上报和成交回报都是异步进行，两者互不影响，大大的加快了交易的速度，当委托上报结束后，券商系统撮合交易成功，便会把成交结果通过回报把数据返回给系统，至此一次交易过程成功结束。

委托上报的流程图如图 4.10 所示。首先系统会把 Redis 中所有未报委托记录的委

托状态更改为待报状态，接下来查询所有的待报委托并调用券商端提供的接口进行委托上报，券商端收到委托上报的记录后会返回委托记录接收是否成功的反馈信息，若反馈信息为接收成功，说明委托数据已经成功上报，更改该委托状态为已报状态，若反馈信息为接受失败，则此次 Job 任务结束，待下一秒 Job 开启后，继续进行委托上报，直至上报成功。

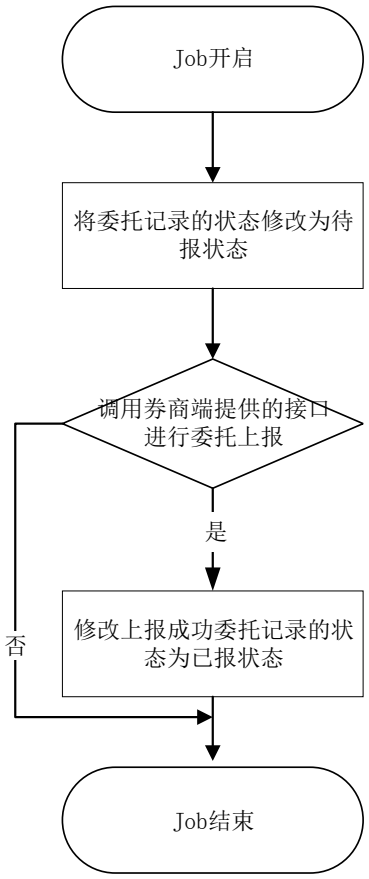


图4.10 委托上报流程图

在成交回报中，券商端系统会调用股票交易系统提供的接口进行成交回馈，为了避免重复的回报记录入库，系统首先会判断数据库中是否有该条回报成交记录，若存在相同的回报记录，直接结束。若不存在，则根据成交回报记录中的委托 ID 查询原始的委托记录，根据委托记录信息修改用户相应的资金账户、持仓列表等并更改该委托记录的状态为已成交状态。

整个买入卖出委托的完整过程如上所述。在委托买入界面中注册用户只需要输入相应的股票代码和股票数量即可下单委托，委托买入股票的数量必须为 100 的整数倍。委托卖出界面中用户首先在持仓列表中选取自己想要委托卖出股票，如果输入的委托数据不合法，只需要根据页面提示修改即可，最后点击卖出按钮即可成功委托。

委托买入界面如图 4.11 所示。

图4.11 委托买入界面

4.5.2 撤销委托

用户在发出买卖委托的指令后，委托数据便会存储在系统的数据库中，与券商系统进行对接之后进行委托上报和成交回报。委托上报和成交回报均属于金融交易系统与券商系统的对接环节，由于在进行对接之前，金融交易系统与券商系统会预先约定接口，委托上报可以通过接口上传用户所提交的委托数据，待券商进行交易成功之后，券商系统同样也可以通过接口把成交成功后的结果回传给金融交易系统。伴随着股市市场行情波动频繁，股票价格很有可能在几秒之内发生比较大的变化，因此撤单操作成为用户经常使用的功能，具体的序列图如图 4.12 所示。

如图 4.12 所示，用户进行撤单操作，用户在撤单列表中选择想要撤单的股票并点击撤单按钮后，通过 entrustCancel 将请求数据提交至后台系统，通过 getEntrust 方法查询出当前股票的委托状态，若股票的委托状态是未报状态，则直接进行撤单操作，即直接在数据库列表中新增一条委托状态为已撤的撤单委托记录，退回用户所用资金或者所持股票，在客户端显示撤单成功。若股票的委托状态是已报状态，则首先需要更改股票的委托状态为已报待撤，在数据库中新增一条委托状态为未报的委托记录，此后委托上报的 Job 通过 withProcessOne 方法上报此委托记录到券商系统进行撤单操作，根据成交回报的具体的结果在客户端显示相应的信息。

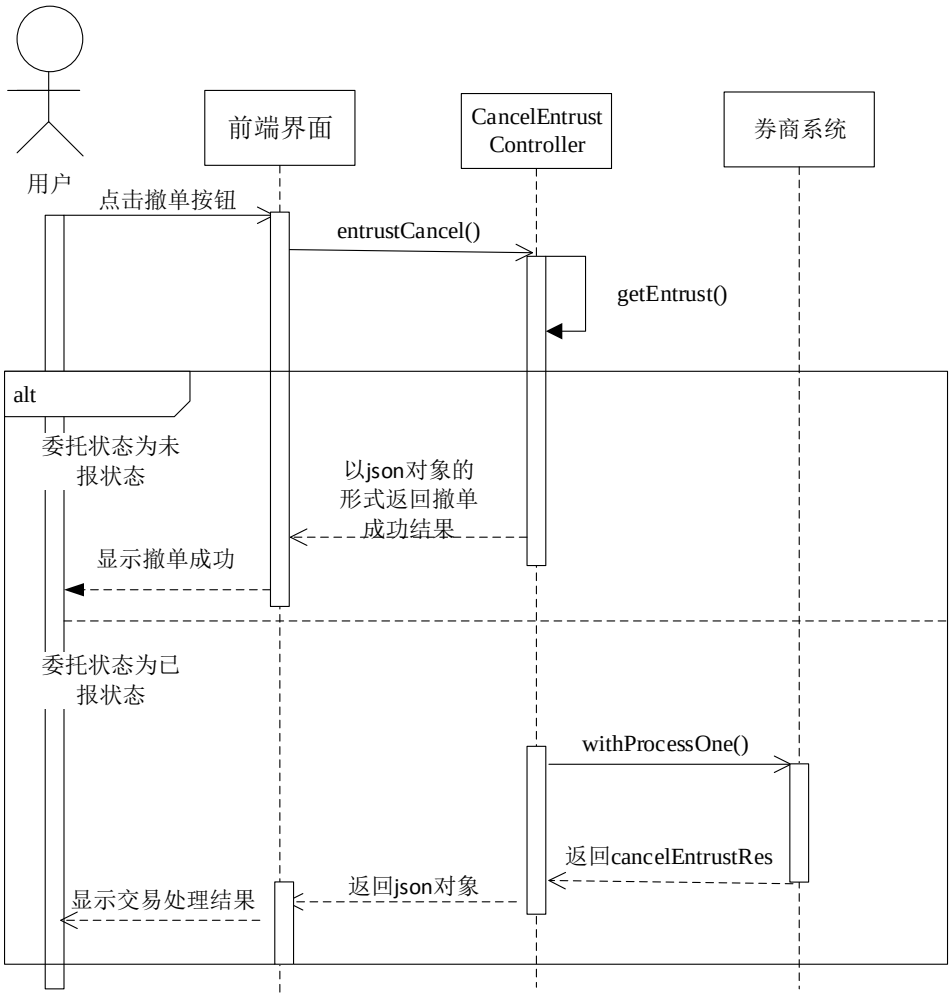


图4.12 撤单功能序列图

具体实现的功能类图如图 4.13 所示，类图说明如下。

CancelEntrustController 继承自 BaseJsonController，提供撤销委托的接口。entrustCancel方法用于完成用户撤销委托的请求，在获取到用户的委托编号之后，通过调用cancelEntrustService服务类的entrustCancelDeal方法，在entrustCancelDeal方法中，调用withdrawProcessOne方法，由CapitalRedisInfo类里面的updateCuBalAmt方法修改Redis中的资金账户余额。

CancelEntrustService类主要完成撤销委托的逻辑实现。entrustCancelDeal方法用于实现撤销委托接口的逻辑实现，transFormObject用于返回数据库中新增的委托对象，entrustCancelManual方法用于完成手工撤单操作。

CancelEntrustRes类用于将处理结果进行实例化，方便将用户的查询结果JSON化处理返回给客户端。

CapitalRedisInfo类用于操作Redis数据库里面的相关数据。

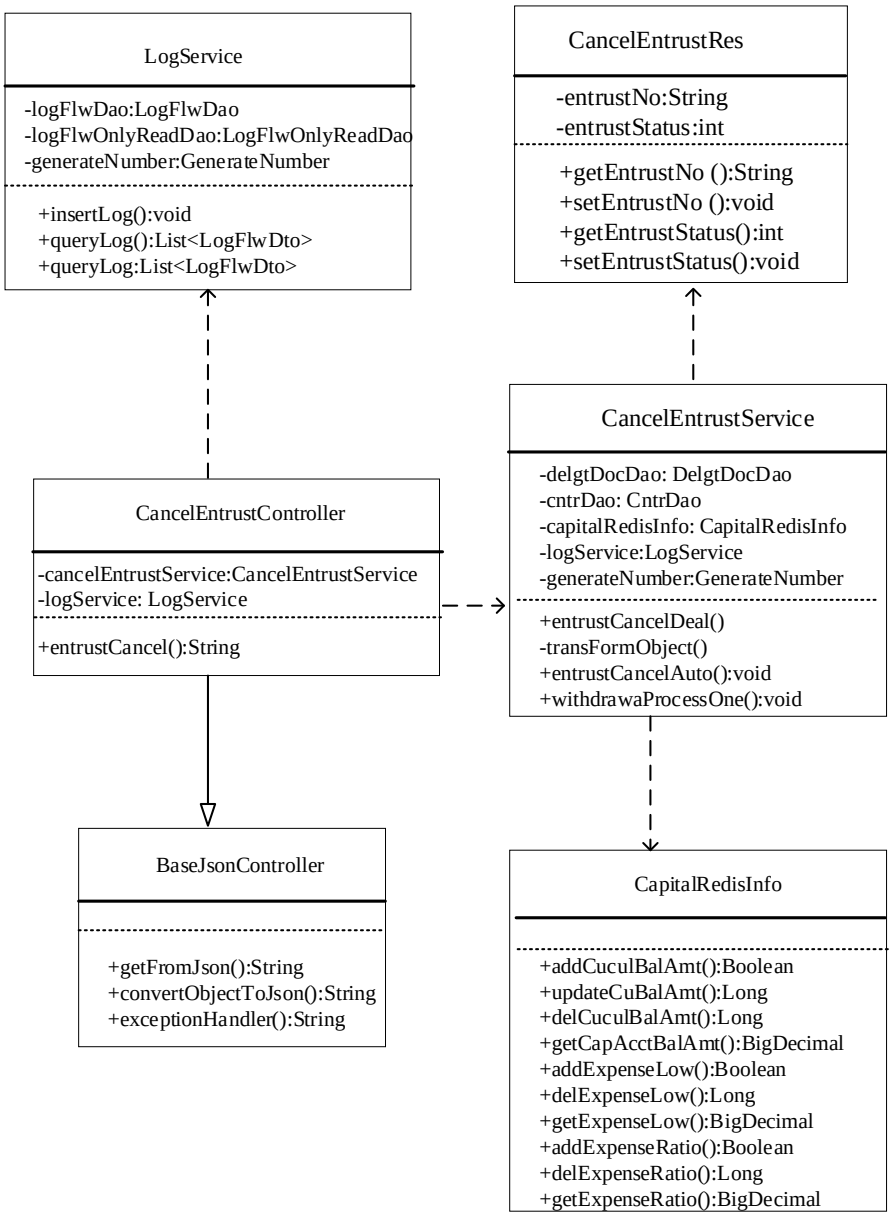


图4.13 撤单功能类图

4.5.3 查询功能

交易客户端的查询功能包括资产查询和委托记录查询，注册用户可以在资产查询界面中及时地了解个人资金账号金额变动、盈亏收益以及所持仓股票价格的变动情况，若因为股票价格下降导致用户资金总额变少，总盈亏以及浮动收益金额会以绿色显示，反之则会用红色表示用户资金增加。在委托记录中可以查看历史成交、历史委托、当日成交、当日委托，每一项均会以股票列表的形式显示，用户可以方便的查看自己所买股票的名称、价格、状态以及成交时间。由于查询功能均是前端页面通过调用后台系统所提供的接口来实现的，具体的实现流程相对比较简单，在此不再详细介绍。资产查询记录界面如图 4.14 所示。

图4.14 资产查询界面

4.6 交易管理模块的实现

4.6.1 账户管理

账户管理模块包含合约管理和资金账户管理，账户管理就是配置角色账户之间的关系。用户注册成功时会自动生成一个合约，用户的所有股票交易都是基于合约的，每个合约里面都会有相应的风险保证金。用户盈利之后便可提现到自己的资金账户，合约和资金账户的管理模块大致类似，都是最基本的查询更新操作，在此以资金账户为例进行说明。类图实现如图 4.15 所示，具体说明如下。

CapAcctController继承自BaseJsonController，为管理端提供管理账户的接口。通过调用CapAcctController类里面的add方法、deleteCapAcct方法、update方法、findAllCapAcct方法完成股票的增删改查功能,逻辑实现均需要调用CapAcctService类里面的insertCapAcct方法、deleteCapAcct方法、updateCapAcct方法、findAllCapAcct方法完成相应的逻辑实现。

CapitalRedisInfo服务类用于对Redis中跟资金账户有关的数据表进行处理。包括增加资金账户的可用余额、更新资金账户的可用余额等。

CapAcctResponse类用于将处理结果进行实例化，方便将用户的查询结果JSON化处理返回给客户端。

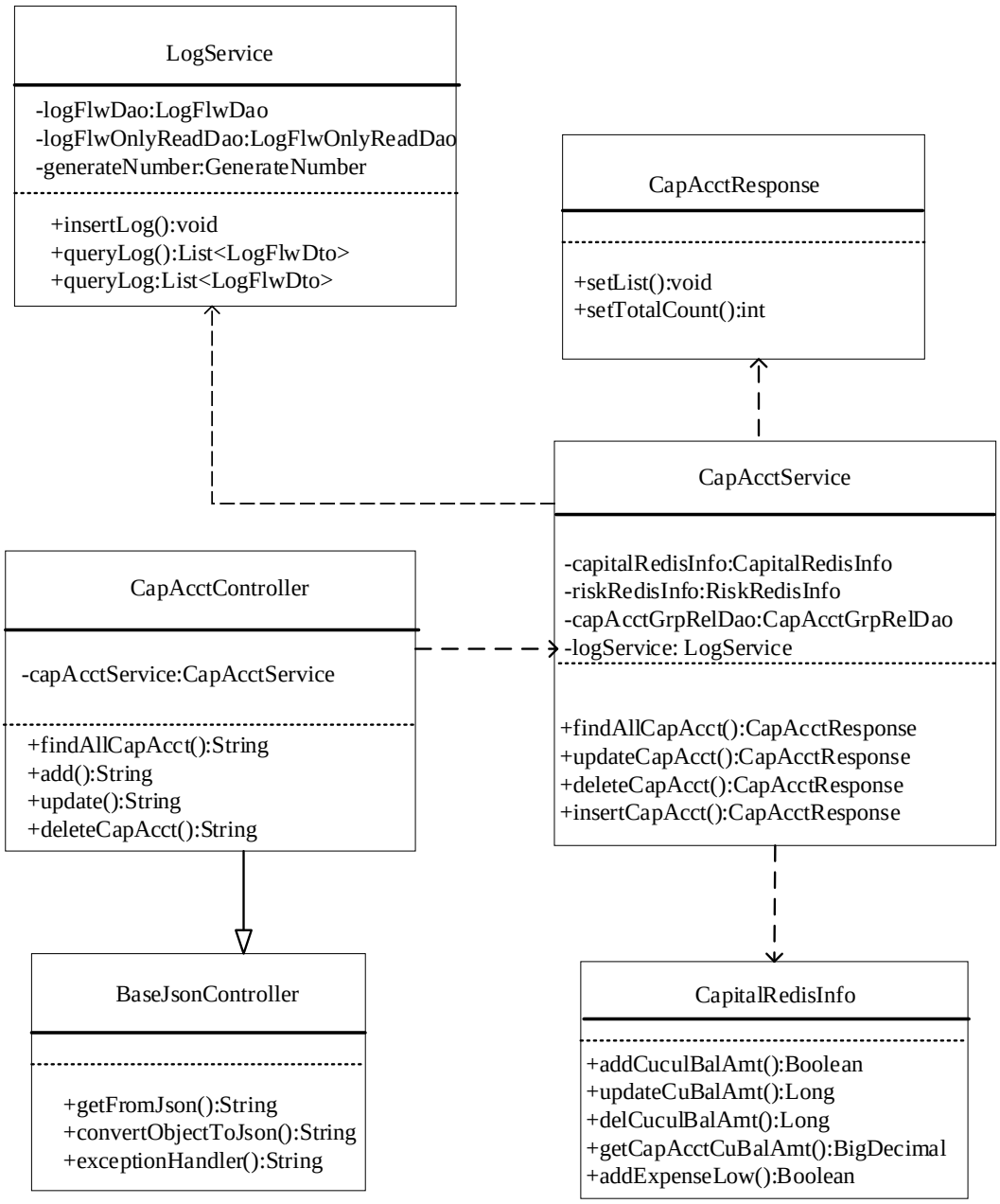


图4.15 账户管理类图

4.6.2 风控管理

国内股票市场起伏不定，为了尽可能保障平台和用户的经济利益，避免不必要的损失，需要对用户的交易过程施加一定的风险控制，风控的本质是尊重市场的不确定性，尽可能的防范由此带来的风险。本系统的风控管理模块主要包含股票管理、今日限股以及交易日管理。

股票管理主要是对股票的信息进行基本的维护，股票来源包括上深交易所的所有股票，会对每只股票的股票代码、所属板块、所属交易市场以及股票名称进行统一的管理。系统的管理员通过股票管理界面可以进行股票的模糊查询以及对股票列表进行

基本的增删改查操作。具体的类图如图 4.16 所示，类图说明如下。

StkManageController继承自BaseJsonController，提供供管理端进行股票管理的接口。通过调用StkManageController类里面的addStk方法、deleteStk方法、updateStk方法、findStk方法完成股票的增删改查功能,逻辑实现均需要调用StkManageService类里面的addStk方法、deleteStk方法、updateStk方法、findStk方法完成相应的逻辑实现，在增改删三种方法里面由stockRedisInfo类的secStockSector对Redis数据库的数据进行操作。

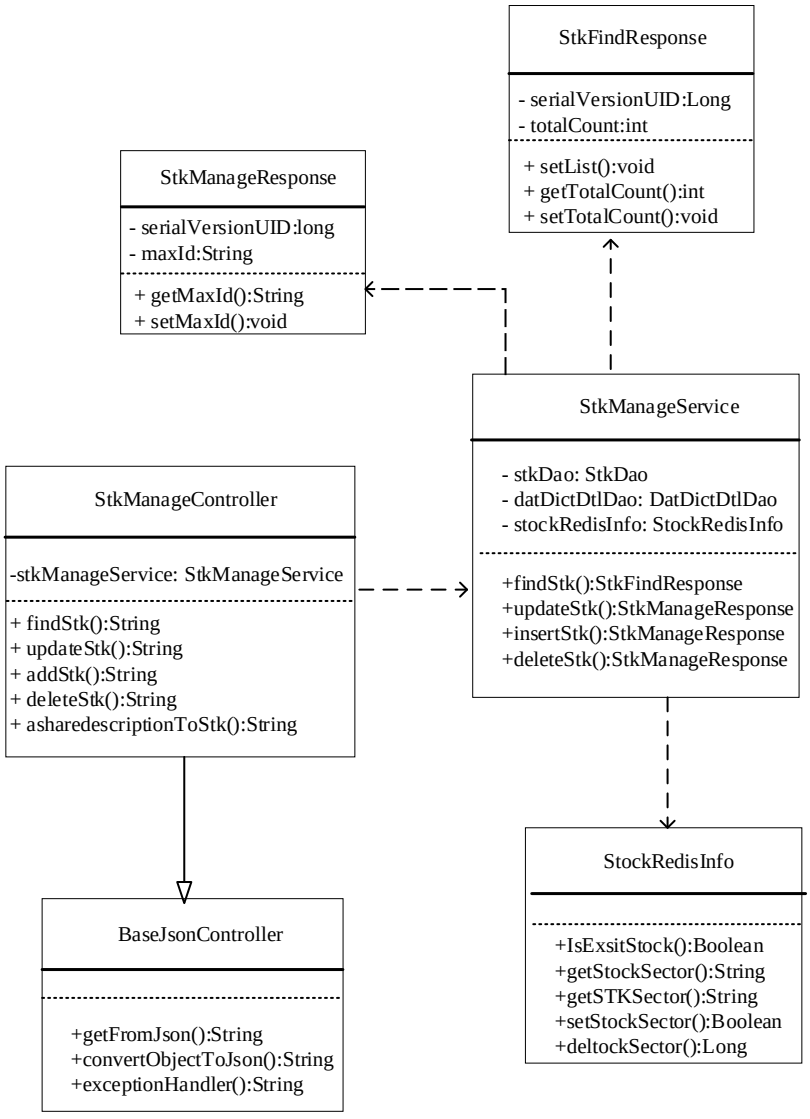


图4.16 股票管理类图

股票管理的界面如图 4.17 所示。管理员通过点击查询按钮可以对股票列表进行模糊查询，点击修改或者删除按钮可以队选中的股票进行相应的操作。

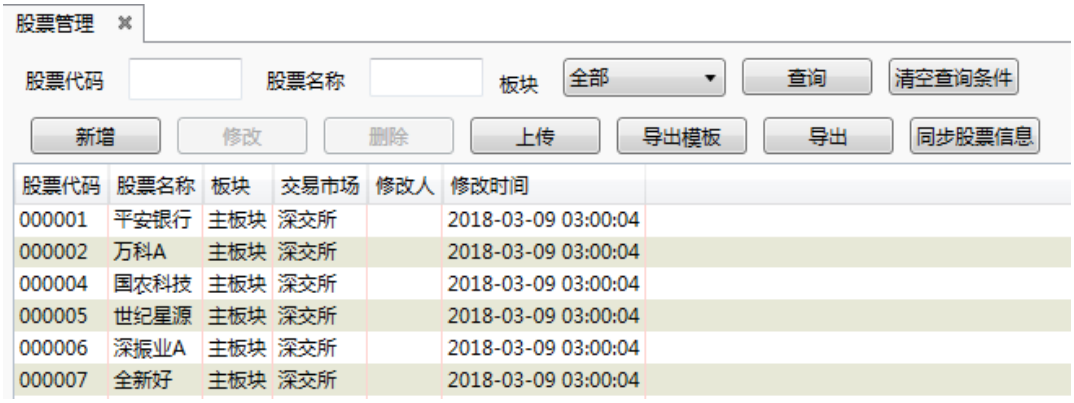


图4.17 股票管理界面

股票市场瞬息万变，一个小小的操作失误就可能会给用户带来损失。今日限股的主要作用是帮助用户提前筛选出停牌股票和高风险股票，限制此类股票的买卖，最大程度的保障用户和平台的利益。今日限股的界面如图 4.18 所示，系统管理员可以对限制股票进行模糊查询，并可以对特定的限制股票进行删除、修改等操作。

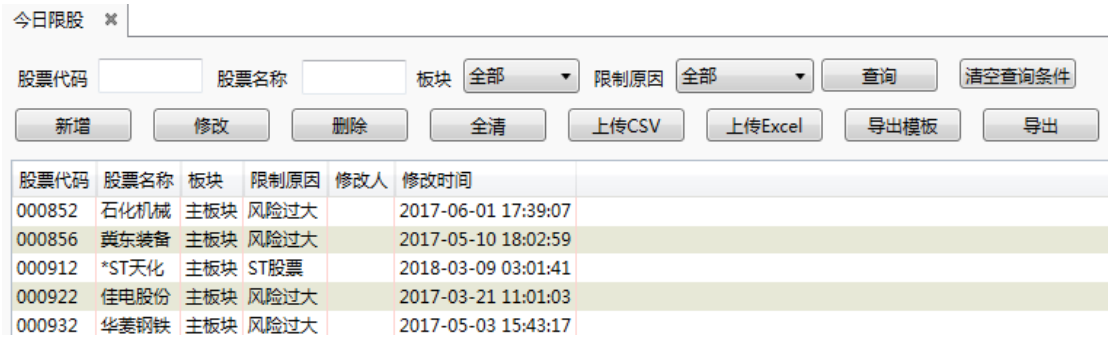


图4.18 今日限股管理界面

交易日管是对股票的交易日期进行管理，所有的股票交易都必须在交易日内才可以进行股票的买卖，在休市时期内是无法进行股票交易的。我国股市的交易日为除节假日外的周一至周五（AM9;30-11;30 , PM13;00-15;00）。对交易日的管理是保障用户可以进行有效交易的前提，系统管理员通过管理界面可以通过设置起始时间和截止时间查询特定时间段内的节假日和后续交易日，点击新增、修改、删除按钮可以对单个日期进行相应的操作。具体的界面如图 4.19 所示。

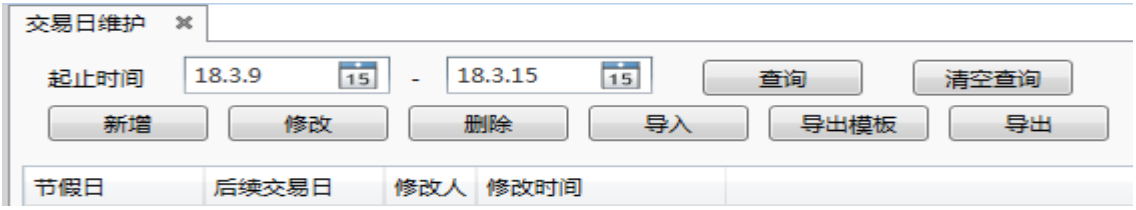


图4.19 交易日管理界面

4.6.3 交易管理

交易管理模块是系统交易端的重要组成部分，是用户能够进行正常交易的重要保障，模块主要提供了分红送股、资金账户流水查询以及资金流水查询等功能，有效地保证了用户的经济利益不受损害，尽可能的使用户的资金安全不受到损失。

分红送股主要是针对用户所得收益的一种管理。分红和送股是投资者获得利润回报的两种不同的方式，一种是上市公司按照股票的一定份额支付给投资者，一种是上市公司采取送股票的形式给投资者分配利润。用户在股票交易中所得的利润主要是通过这两种方式得来的。功能实现的类图如图 4.20 所示，类图说明如下。

BnsAllotController继承自BaseJsonController，给管理端提供分红送股的接口。sentShareBns完成分红派股的功能，通过convertJsonToObject把参数转换成对象之后，调用bonusDistributeService类的bonusDistribute完成逻辑实现，其中capitalRedisInfo类的updateCuBalAmt处理Redis中资金账户的可用资金

CapitalRedisInfo服务类用于对redis中跟资金账户有关的数据表进行处理。包括增加资金账户的可用余额、更新资金账户的可用余额等。

图4.20 分红送股类图

分红送股管理界面如图 4.21 所示，系统管理员可以通过设置分红送股类型、起始时间和截止时间、分红送股状态和审核状态进行分红送股的查询，点击清空查询按钮可以恢复设置初始化。点击新增按钮，可以在弹出的窗口中进行信息的添加，通过修改、删除按钮可以对选中的条目进行相应的操作。

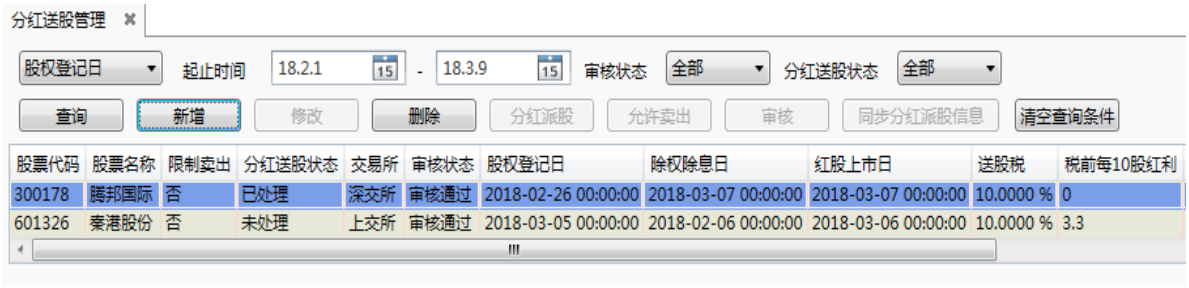


图4.21 分红送股管理界面

流水管理就是对用户的资金流动情况以流水的形式进行记录，对整个系统的注册用户的资金状况有一个宏观上的把控，通过对资金流水的分析，不仅可以制定出更加完善的管理机制，还可以作为与券商系统进行对账的重要依据。系统管理员可以通过设置流水时间的起止时间、流水类型以及进出标记根据实际的查询需求进行精准查询，点击清空查询条件可以初始化查询条件。在查询结果列表中通过选择资金账户类型和单证类型即可查看特定种类的资金账户流水信息。具体的资金账户流水界面如图 4.22 所示。

图4.22 资金账户流水记录

4.6.4 结算管理

作为一个以交易为主的金融交易系统，平台每天都要处理巨量的买入卖出请求，所涉及的每笔交易都会发生资金的变动，因此结算模块便成为系统中不可缺少的一环。本系统中结算模块是对用户交易费率的管理，用户在进行交易委托的时候必须缴纳一定的手续费才可以完成交易，由印花税、佣金和过户费组成。无论是买入还是卖出均需要收取，具体的实现类图如图 4.23 所示，类图的说明如下。

TrdFeeRtController继承自BaseJsonController，给管理端提供交易费率管理的接口。insertTrdFeeRt、delTrdFeeRt、queryTrdFeeRt、queryTrdFeeRt完成交易费率增删

改查的功能，通过调用trdFeeRtService的方法完成相应的逻辑功能。

CommonRedisInfo用来对系统存储在Redis中的常量数据进行管理，比如佣金系数、印花税、过户费等。

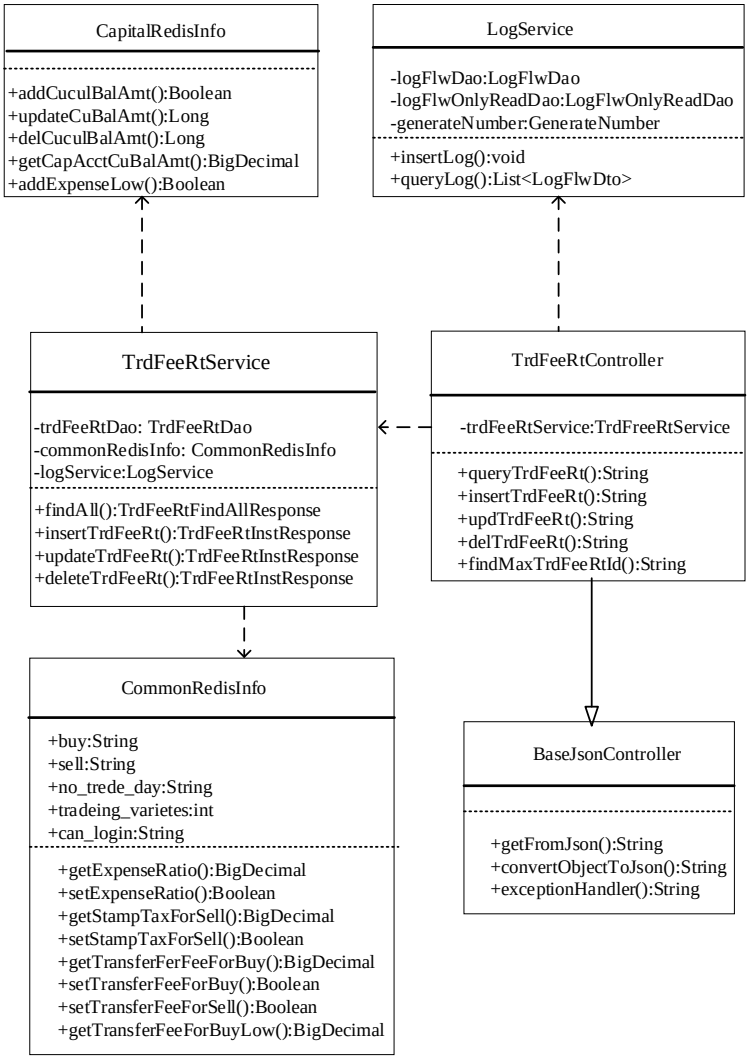


图4.23 交易管理费率类图

交易费率管理界面如图 4.24 所示。管理员通过点击查询按钮可以查询到所有交易所的交易费率情况，点击修改或者删除按钮可以对选中的交易所费率进行操作。点击新增按钮可以添加新的交易所。

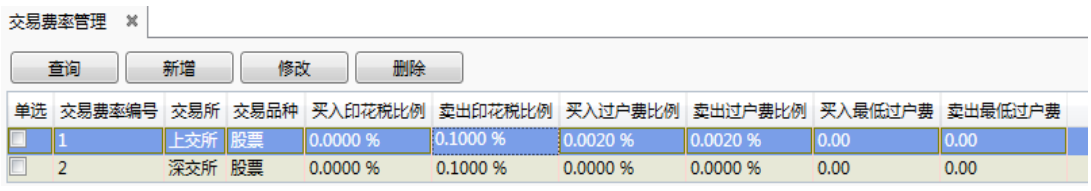


图4.24 交易费率管理界面

4.7 Redis 模块的实现

4.7.1 session 管理

Session 在网络应用中称为“会话控制”，主要用于对用户会话期间所需的配置信息及属性进行存储[32]。当用户在 Web 页面之间相互跳转时，Session 中存储的信息将会在会话期间一直保持下去。其工作原理如图 4.25 所示，当用户从客户端发起页面请求时，服务器会首先在 Cookies 里面检测是否存在 Session_ID，如果不存在，服务器就会自动创建一个 Session 对象并放在 Session 集合中，并在此次响应中返回给客户端[33]。

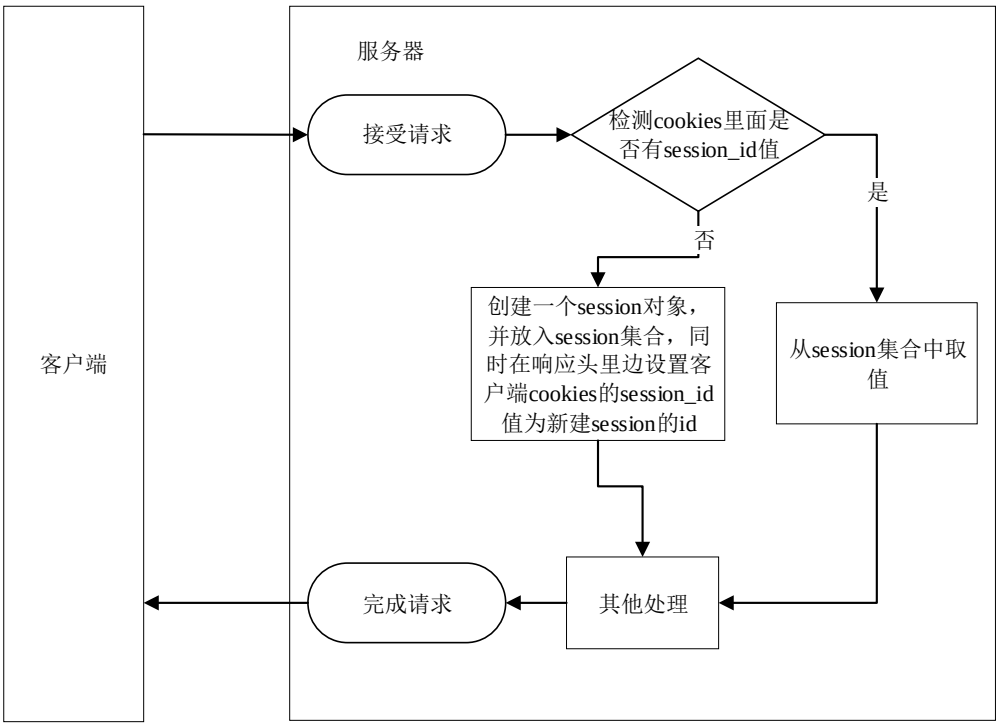


图4.25 session 工作原理图

作为一款大型的应用系统，用户的数量和访问量与日俱增，需要管理的 Session 也会随之变得庞大。为了应对用户的访问量所带来的性能上的压力，需要架设多台服务器，如何对共有的 Session 对象进行管理就成为系统设计初期必须要考虑的问题。由于 Session 存储介质需要具备较高的读写性能，所以应该选择缓存服务器作为存储介质，目前业内常用的两种缓存服务器是 Redis 和 Memcache，两者的对比如表 4.7 所示。

表4.7　Redis 和 Memcache 对照表

内存数据库	支持的数据类型	持久性	分布式存储	数据一致性	读写 10 万个值用时
Redis	支持的类型丰富，能够支持 set 和 list	支持	可以使用一致性 Hash 做分布式	使用单线程模型，保证数据提交顺序	17s/16s
Memcache	支持的类型简单，需要客户端自己处理复杂的对象	不支持	支持 Master-slave 模式	需要使用 CAS 保证一致性	15s/15s

从表中的对比可以看出，虽然 Redis 在读写性能方面会稍稍的弱于 Memcache，但是 Redis 支持的数据类型比较多，而且支持数据的持久化[34]。因此本系统采用 Redis 对 Session 进行管理，其实现机制如下：在客户端的 Cookie 中存储一个 SessionID,当客户端向服务器发起一个请求之后，服务器会将其作为 Redis 的 key 值进行检索查找找到对应的 value 数据。如果用户在一定的时间内没有再次进行请求，Redis 会根据提前设定好的 Expire 关键字对 Session 进行过期处理。在此方案中使用 Redis 的好处在于 Redis 提供了良好的主从复制和集群能力，能够很好地保证 Session 存储的高可用性，而且还提供了数据失效和订阅通知的能力，能为 Session 共享提供良好的支撑。Redis 中存储 Session 采用的是 String 类型，key 是用户在系统中登录的唯一标识，value 用于存储用户的登录信息，如账户 ID，密码等，Expire 是针对于 key 设置的，用于设置过期时间，具体结构如表 4.8 所示。

表4.8 Session 存储结构

key	value	Expire
session:m:sessionId	系统管理员的账户信息	8h
session:c:sessionId	注册用户的账户信息	1h

4.7.2　行情数据存储

（1）行情数据介绍

实时的行情数据提供服务是证券市场赖以运转的基础，它不仅以数据的形式为广大的用户描绘出瞬息万变的市场动态，而且也是从事量化投资的用户赖以建模的基础。

由于证券市场瞬息万变，秒级的数据差异都有可能为用户带来不可估量的损失，因此，能够提供实时的行情数据服务就成为系统能否提供良好用户体验的基础。行情数据主要包括股票的开盘价、收盘价、成交量、换手率等，这是一种无结构的数据，在 Redis 中这些数据被设置成 String 类型，key 为每只股票的代码，value 是股票的行情数据，过期时间设置为永不过期。具体的存储结构如表 4.9 所示。

表4.9 行情数据结构

key	value	Expire
股票代码	开盘价|收盘价|最高价|最低价|成交额|市盈率|总市值|成交量	永不过期

（2）行情数据的获取存储流程

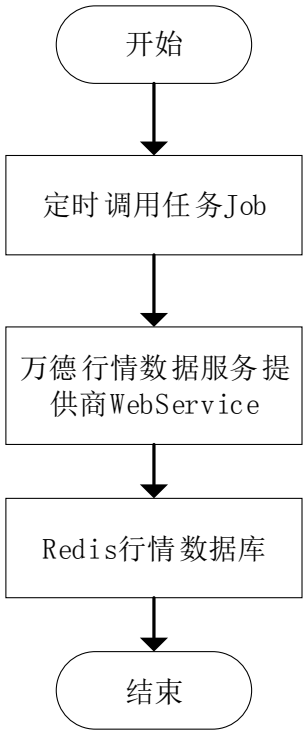


图4.26 存储行情数据流程图

如图 4.26 所示，系统通过 Quartz 框架的定时调度 Job 实时地调用万德行情数据服务提供商所提供的 WebSocket 接口，请求最新的股票行情数据，万德行情数据服务提供商在接收到请求之后，会把最近的股票行情数据实时地推送给 Redis，平台接收到返回的数据之后会把行情数据按照指定的格式存入在 Redis 中，供系统调用使用。

4.7.3 指令队列

股票交易系统中的指令队列主要是利用 Redis 中的 List 队列，key 为合约所在的队列编号，value 为合约编号，过期时间为永不过期。指令队列是由操作队列和平仓队列两个部分组成，用户在发出买入卖出请求后生成的合约会放在操作队列中，通过 Quartz 调度 Job 实时的计算用户合约金额的变化，如果合约触发用户预先设置的平仓线，系统就会把合约从操作队列移到平仓队列中，平仓队列中的股票就会被系统自动卖出。

（1）持仓市值的计算

如图 4.27 所示，系统通过配置的 crontab 表达式“0/5 * 9,10,11,13,14,15 * * ?”会每隔 5s 计算合约中的持仓市值以及相关金额，并把计算结果同步在 Redis 中，由于合约的编号是唯一的，每次重新计算的数据会覆盖原先的旧数据，使用 Redis 的管道目的是为了保障数据的一致性。

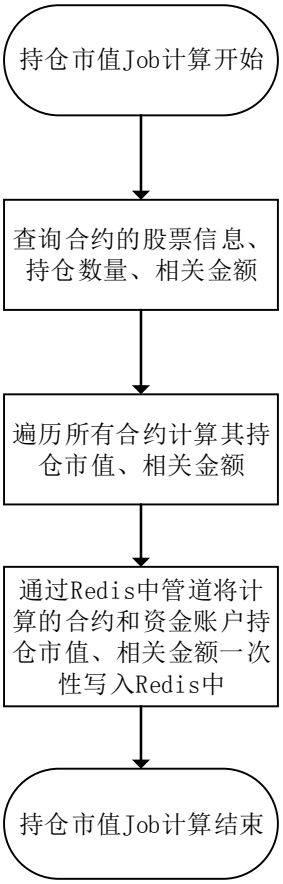


图4.27 持仓市值计算流程图

（2）操作队列和平仓队列

系统在用户发出买入卖出交易请求之后会将生成的合约放入普通队列中，合约在

操作队列的存储结构中的 key 为 contract:[0-9](系统会默认存在 0-9 个队列，会根据合约编号 mod 10 的结果将相应的合约放入相对应的队列中)，value 的值为合约编号，过期时间为永不过期。Quartz 调度 Job 会不断的计算合约的持仓市值，如果合约触发了用户设置的平仓线，系统就会把达到平仓线的合约放入到平仓队列中，平仓队列在 Redis 中的存储结构 key 为 contract_close,value 为合约编号，过期时间为永不过期。存储结构如表 4.10 所示。

表4.10 队列存储结构

key	Value	Expire	含义
Contract_active:n: [0-9]	合约编号	永不过期	普通队列
Contract_for_close:[0-9]	合约编号	永不过期	平仓队列

4.8 本章小结

本章是对金融交易系统的实现章节，从宏观角度上对系统的整体架构和功能模块进行了概述。系统结构设计一节从系统架构层面把系统分为了业务层、逻辑层、数据层三大部分，并对每一个部分所采用的技术及原因进行了详细的阐述；系统数据库设计介绍了系统的具体表格设计；系统功能设计和实现章节根据需求分析对系统的功能模块进行了分类设计，并概述了每一个功能模块所起的作用，通过序列图、类图的方式对功能模块的实现进行了详尽的说明。

第五章 系统测试

随着互联网浪潮的袭来，股票证券交易已经与移动互联技术深度融合，相对于没有互联网的时代，股票类软件在其交易数量、交易频率上已经发生了质的改变。由于股票类软件与用户的资金安全息息相关，在正式上线之前必须进行完备的测试，以保障系统各个功能模块能够按照预期正常运行[35]。本章主要介绍功能测试和性能测试。功能测试指测试验证系统中的各个模块，找出不易被发现的系统漏洞，检查系统能否达到用户的要求。性能测试是测试系统中接口的性能，检查其是否能够达到系统的设计要求，通过并发测试观察系统能否在高负载的情况下保持稳定性。

5.1 测试方案

5.1.1 测试目标

股票交易系统的测试主要分为两个部分：功能测试和性能测试。功能测试的主要目的是尽可能的覆盖系统所有的功能点，确保达到系统需求分析的设计要求。在完成系统的功能测试之后，还需要对系统的性能进行一定的测试。由于 APP 的注册用户众多，在特定的时间点会遇到一定的峰值请求，因此提供一个高效稳定且能够应对高并发的接口访问不仅可以保障系统的稳定性，还可以提升用户的使用体验。

5.1.2 测试范围

本系统的测试范围包括行情模块和交易模块，针对这两个模块进行功能点的逐一测试，确保模块之间能够协同工作。在功能测试的基础上，在对系统的各个接口进行性能测试，主要是针对接口的响应速度和并发能力，验证是否能够达到系统的性能要求。

5.1.3 测试环境

测试环境分为测试手机（包括安卓手机和苹果手机）和后台测试服务器。测试手机包括手机的硬件配置以及所使用的 APP，后台测试服务器也包括服务器的硬件配置和软件配置。其中安卓测试机的配置如表 5.1 所示。

表5.1 安卓测试机配置

名称	配置条件
手机	华为 nova 2S
操作系统	Android

表 5.1（续） 安卓测试机配置

中央处理器(CPU)	海思(Hisilicon)
ROM 容量	64GB
RAM 容量	4GB
APP 软件	量加股票 android 版

苹果测试机配置如表 5.2 所示。

表5.2 苹果测试机配置

名称	配置条件
手机	苹果 6s
操作系统	iOS 11.2.6
中央处理器(CPU)	苹果 A10 2.23GHz(4 核)
ROM 容量	64GB
RAM 容量	2GB
APP 软件	量加股票 ios 版

测试服务器的配置如表 5.3 所示。

表5.3 测试服务器配置

名称	配置条件
操作系统	Window Server 2008R2
中央处理器(CPU)	Intel
内存	32G
硬盘	300GB
数据库	mysql
Redis	redis 3.2.1

5.2 系统功能测试

功能测试是指系统测试人员按照产品业务流程进行相应的操作，通过模拟不同的用户，对系统的各个模块功能进行检测，确保系统能够按照规定的流程正常使用，与此同时，通过设定边界值，测试系统在边界条件下是否会出现预想之外的响应，并及时记录异常情况提交给开发人员，保障系统的正常运行。

5.2.1 行情模块功能测试

行情模块主要测试与股票相关的新闻热点、股票详情、K 线走势能不能正确的在 APP 页面上面显示。行情模块平台主要是针对用户使用，具体的测试用例如表 5.4 所示，用户可以通过相应的板块查询到自己所需要的信息，由于三者的功能比较相似，仅仅是显示内容的不同，这里以查询股票新闻热点为例。

表5.4 查询新闻热点功能测试用例表

测试用例及说明	预期结果	实际结果
使用已经注册的手机号或者微信号登录系统	能够成功登陆系统	正常
使用未注册的微信号登录系统	使用微信号登录系统时，会提示此微信号首次登陆，请绑定手机号。	正常
在行情菜单中的热门资讯板块获取最新热门新闻	能够获取系统推送的所有热点资讯，并且支持按照时间先后排序	正常
下拉页面，刷新热门资讯信息	缓存中有新的热门资讯时会在页面上按照时间先后的顺序予以显示，若缓存中不存在，则保持原来的新闻列表不变	正常
获取某个特定的新闻详情	能够获取特定的新闻详情	正常
点击分享按钮，分享到微信指定好友、朋友圈等	成功的分享到朋友圈，点击可以浏览新闻资讯	正常
点击返回按钮	能够返回到热门资讯页面	正常

针对行情模块的预期结果进行测试，具体测试结果如下。

用户登录 APP 分为微信登录和手机登录两种方式，界面如图 5.1 所示。

图5.1 登录界面

首次使用未注册过的微信登录 APP，系统会提示“该微信账户在系统中不存在，

请使用微信注册”信息，测试界面如图 5.2 所示。

图5.2 热点新闻测试界面

用户在首页点击热门资讯栏目，APP 会引导用户进入行情页面，行情页面会显示所有热门资讯，测试界面如图 5.3 所示。

图5.3 热门资讯界面

用户点击新闻标题，APP 会跳转到资讯详情页面，测试界面如图 5.4 所示。

图5.4 新闻详情页面

用户点击右上角的分享按钮，可以将新闻分享至指定的好友或者朋友圈，测试界面如图 5.5 所示。

(a)APP 分享页面 (b)微信朋友圈分享页面

图5.5 新闻分享界面

5.2.2 交易模块功能测试

交易模块是为用户提供可在线交易股票的平台，主要包括资产查询、委托记录查询、买入卖出委托、撤销委托四大功能。

（1）查询功能测试

查询功能包括用户的资产查询和委托成交记录查询，在委托成交记录查询中包含当日成交、当日委托、历史成交、历史委托四个部分，主要的功能是供用户了解自己的资产及委托流水情况。具体的测试用例表如表 5.5 所示。

表5.5 查询功能测试用例表

测试用例及说明	预期结果	实际结果
用户登录，进入委托交易，点击资产选项卡	用户可以看到自己的总资产、总盈亏、持仓市值、浮动收益、可用余额。用户所持股票会以列表的形式在页面下方显示，如果用户盈利，总盈亏、浮动收益以及股票的盈亏数值颜色为红色，反之则为绿色。	正常
用户登录，点击查询选项卡，点击当日成交	展示当日成交的股票信息，包括股票的名称/代码、价格/数量、状态/类型、成交时间	正常
用户登录，点击当日委托	展示当日委托的股票信息，包括股票的名称/代码、价格/数量、状态/类型、成交时间	正常
用户登录，点击历史成交	展示历史成交的股票信息，包括股票的名称/代码、价格/数量、状态/类型、成交时间	正常
用户登录，点击历史委托	展示历史委托的股票信息，包括股票的名称/代码、价格/数量、状态/类型、成交时间	正常
用户登录，点击返回按钮	返回到股票详情页面	正常

查询功能包括资产查询和委托记录查询，测试界面如下所示。

用户点击查询选项卡，可以看到当日成交、当日委托、历史成交、历史委托四个选项，点击历史成交可以看到用户所有买卖成功的股票信息。测试界面如图 5.6 所示。

(a)委托记录查询 (b)历史成交列表

图5.6 委托查询界面

用户选择资产选项卡，可以查看自己的资产详情，测试界面如图 5.7 所示。

图5.7 资产查询界面

（2）买入卖出委托功能测试

买入卖出委托是供注册用户使用的，具体的测试用例表如表 5.6 所示，非注册用户不能进行股票的委托买卖操作。由于买卖的界面基本一致，区别在于一个是买入一个是卖出，这里以买入为例。

表5.6 买入委托功能测试用例表

测试用例及说明	预期结果	实际结果
用户登录，进入委托交易页面，点击买入选项卡，输入股票代码	在股票代码输入正确后显示股票名称，在价格文本框显示当前的股票单价及涨跌价	正常

表 5.6（续） 买入委托功能测试用例表

用户登录，点击股票单价文本框的减号或者加号按钮	股票单价会相应的减少或者增加	正常
用户登录，在股票单价文本框中输入低于跌停价的数字或者高于涨停价的数字	股票单价会默认显示跌停价格或者涨停价格	正常
用户登录，在委卖股数文本框中输入合理的股票数量，点击买入按钮	页面会弹出委托买入确认弹框，在弹框显示所要委托的股票代码、股票名称以及委托的价格和数量	正常
用户登录，在非交易时段内点击确认买入按钮	页面会提示委托失败	正常
用户登录，在委托买入确认弹框上点击确认买入按钮	页面显示委托成功提示，在查询选项卡中的当日委托中可以查询到刚刚所委托的股票信息，状态显示未成交	正常

买卖委托的界面比较相似，在这里以买入委托为例，测试结果如下所示。

用户在股票代码框中可以输入股票代码，APP 会自动显示股票的价格。测试界面如图 5.8 所示。

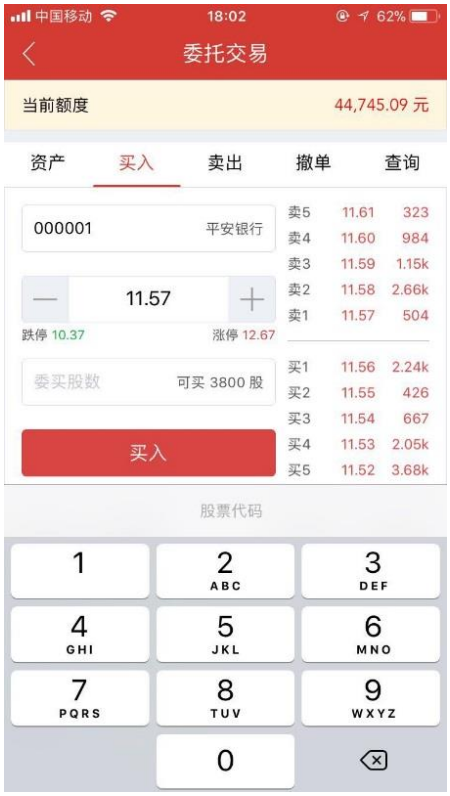
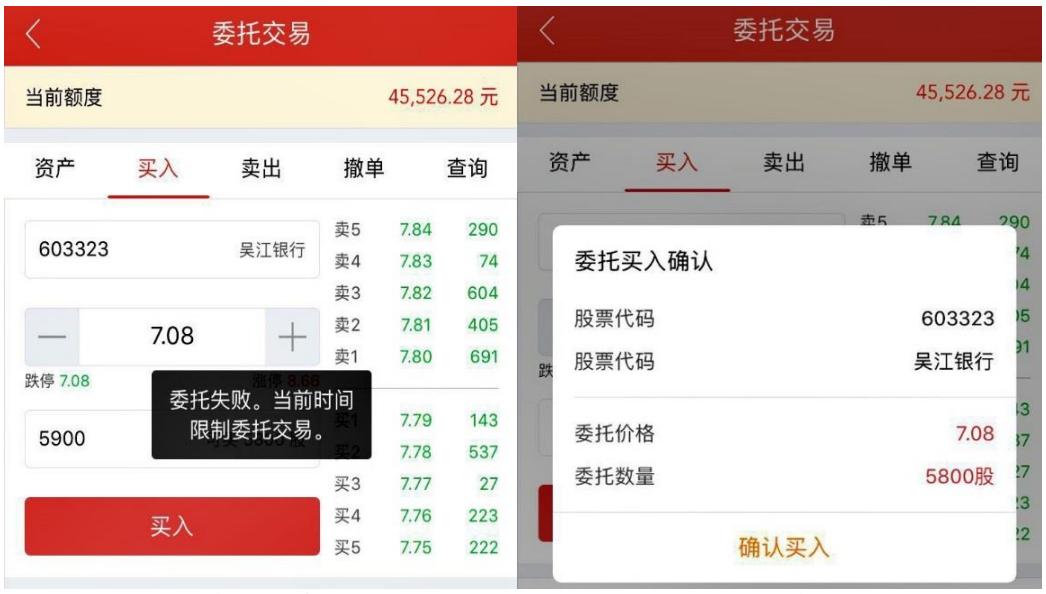


图5.8 股票代码输入测试界面

用户选择买入选项卡，如果在非交易时间内进行委托请求，用户在交易时间内进

行委托交易请求，系统会弹出委托买入确认弹框，测试界面如图 5.9 所示。



(a)委托交易失败　　(b)委托交易成功

图5.9 买入委托测试界面

（3）撤销委托功能测试

当用户因为误操作或者没有以心仪的价格委托股票时，可以在撤单界面进行撤销操作，具体的用例表如表 5.7 所示。

表5.7 撤销委托功能测试用例表

测试用例及说明	预期结果	实际结果
用户登录，在买入界面成功进行一笔委托交易，点击撤单选项卡	页面显示刚成功委托的股票信息	正常
用户登录，点击所要撤单股票前面的小圆圈按钮	小圆圈按钮里面会显示对号图标，表示用户已经选中此股票	正常
用户登录，点击撤单按钮	页面会显示撤单成功提示，下拉页面，被撤单的股票信息不会在页面中显示。在查询选项卡中的当日委托中可以可以查询到被撤销股票的信息，状态显示为已撤销	正常
用户登录，点击返回按钮	页面返回到股票详情页面	正常

撤销委托测试界面如图 5.10 所示。用户选中想要撤单的委托数据，点击撤单按钮，如果委托数据还没有被券商撮合交易，系统就会返回撤单成功提示。

图5.10 撤销委托测试界面

5.2.3 交易管理模块功能测试

（1）账户管理功能测试

账户管理用于配置角色账户之间的关系，对账户进行一定的管理。具体的测试用例表如表 5.8 所示。

表5.8 账户管理功能测试表

测试用例及说明	预期结果	实际结果
管理员登录，选择资金账户管理选项卡，点击查询按钮	页面显示出所有的资金账户信息	正常
管理员登录，设置起止时间，点击查询按钮	页面显示管理员所设定时间下所有的资金账户信息	正常
管理员登录，点击清空查询条件按钮	页面上管理员所设置的全部查询条件全部恢复初始化	正常
管理员登录，点击导出查询结果按钮	页面会弹出选择存储路径页面，并以 Excel 表的形式存储到指定的路径	正常
管理员登录，选中一条资金账户信息，点击删除按钮	页面弹出提示是否删除当前资金账户	正常
管理员登录，点击新增按钮	页面弹新增限制股票弹框	正常
管理员登录，在新增资金账户弹框上输入相应的合法信息，并点击提交按钮	页面显示操作成功提示，数据库中表的信息同步修改	正常

账户管理测试界面如图 5.11 所示。当管理员点击新增按钮时，管理界面弹出新增资金账户弹框，若填入的数据有误或者为空，系统弹出错误提示信息。

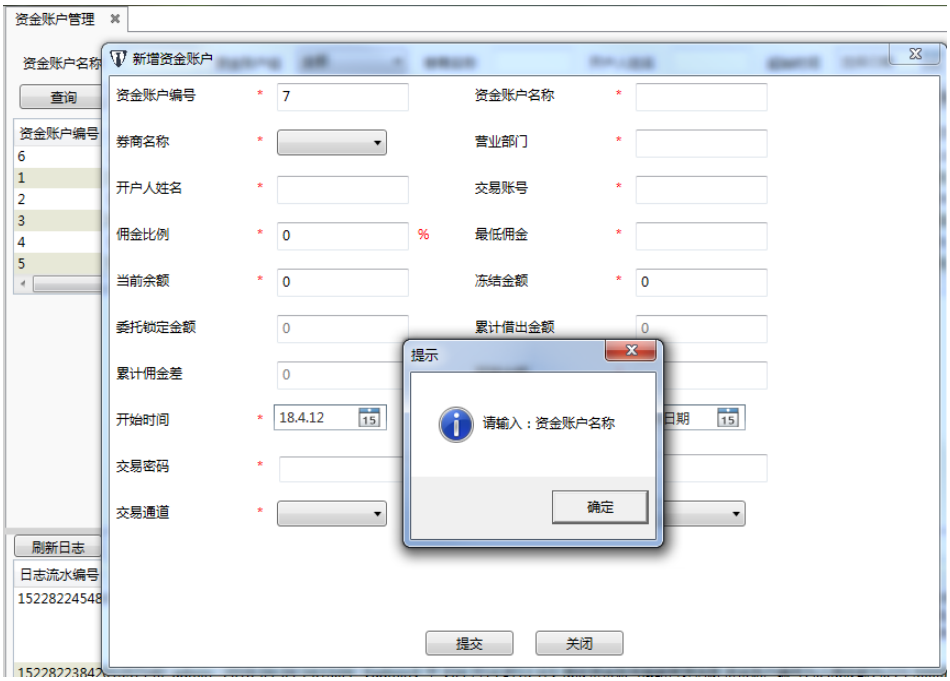


图5.11 账户管理测试界面

（2）风控管理功能测试

风控管理模块的功能是为用户的交易提供一定的风险控制，包括今日限股、股票管理、交易日管理三大部分。具体的测试用例表如表 5.9 所示。

表5.9 风控管理功能测试用例表

测试用例及说明	预期结果	实际结果
使用具有管理员权限的账号登录交易管理端	能够成功登录交易管理端	正常
使用不具备管理员权限的账号登录交易管理端	不能够登陆成功，页面显示登录失败提示	正常
管理员登录，选择今日限股选项卡，点击查询按钮	页面显示符合初始条件的所有股票列表	正常
管理员登录，输入正确的股票代码、股票名称，选择所属板块和限制原因	页面显示符合条件的股票信息	正常
管理员登录，点击清空查询条件按钮	初始化所有的查询条件，清空股票代码文本框和股票名称文本框	正常
管理员登录，点击新增按钮	页面弹出新增限制股票弹框	正常
管理员登录，在新增限制股票弹框上输入相应的合法信息，并点击提交按钮	页面显示操作成功提示信息，数据库中表的信息同步修改	正常

表 5.9（续） 风控管理功能测试用例表

管理员登录，选中一条股票信息，点击修改或删除按钮	点击修改按钮，页面会弹出修改限制股票弹框。点击删除按钮，页面会删除所选股票信息，数据库同步刷新	正常
管理员登录，在修改限制股票页面修改限制原因，并点击提交按钮	页面显示修改成功提示信息，数据库中的数据也会同步更改	正常
管理员登录，点击导出按钮	页面会弹出选择存储路径页面，并以 Excel 表的形式存储到指定的路径	正常
管理员登录，点击上传 Excel 按钮	页面会弹出选择指定 Excel 文件页面，管理端界面会同步刷新股票列表，数据库同步刷新	正常
管理员登录，点击全清按钮	删除页面中所有的股票列表，数据库中的数据同步删除	正常
管理员登录，设置起止时间，点击查询按钮	页面显示指定时间段内的节假日和后续交易日	正常

交易日维护、股票管理和今日限股界面类似，在此以今日限股为例。管理员点击今日限股选项卡，点击查询按钮可以看到所有的今日限股信息，点击新增按钮，管理界面弹出新增限制股票弹框，若输入数据有误或者为空，弹出相应提示信息。

今日限股测试界面如图 5.12 所示。

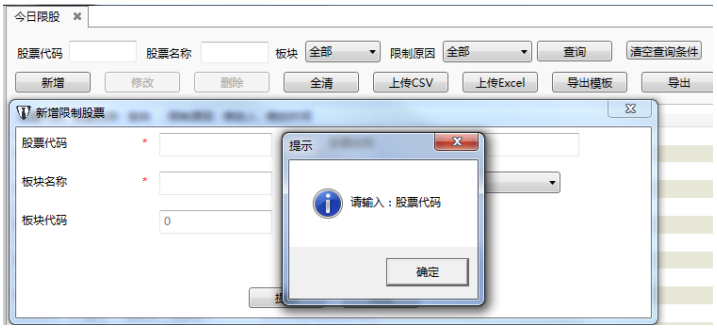


图5.12 今日限股测试界面

（3）交易管理功能测试

交易管理模块包含分红配股管理和账户流水查询两大部分。具体的测试用例表如表 5.10 所示。

表5.10 交易管理功能测试用例表

测试用例及说明	预期结果	实际结果
管理员登录，选择资金账户流水记录选项卡，点击查询按钮	页面显示出所有账户的所有流水信息	正常
管理员登录，设置流水起止时间，选择流水类型、选择进出标记，点击查询按钮	页面显示出管理员所设置条件下的所有流水记录	正常
管理员登录，点击清空查询条件按钮	页面上管理员所设置的全部查询条件全部恢复初始化	正常
管理员登录，点击导出查询结果按钮	页面会弹出选择存储路径页面，并以 Excel 表的形式存储到指定的路径	正常
管理员登录，设置分红送股类型、起止时间、审核状态、分红送股状态，点击查询按钮	页面显示所有符合条件的股票信息	正常

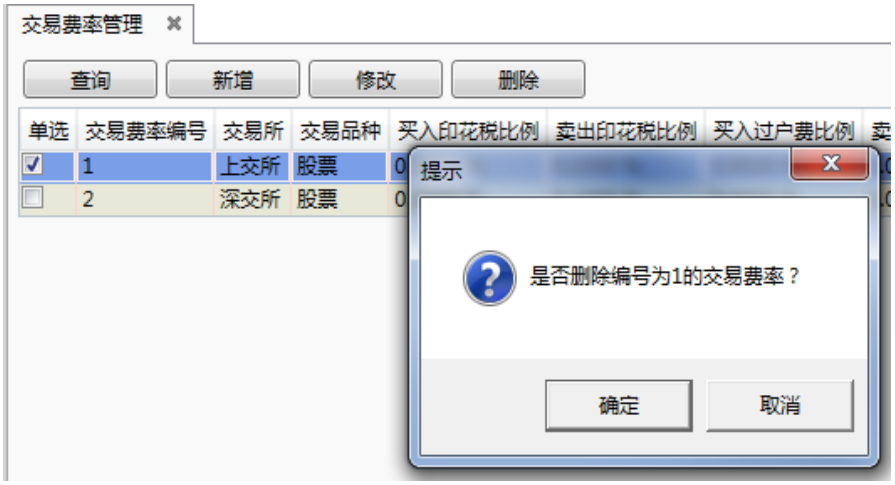
（4）结算管理功能测试

结算管理模块是对交易费率的管理。具体的测试用例表如表 5.11 所示。

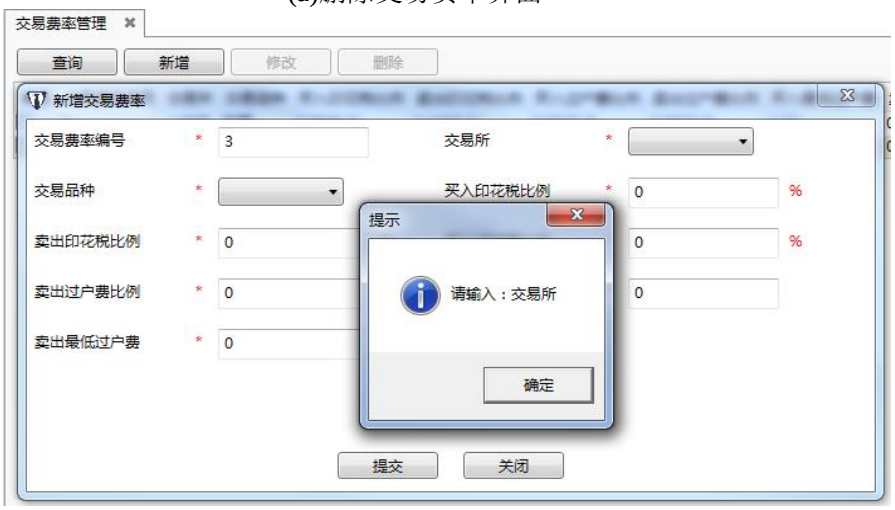
表5.11 结算管理功能测试用例表

测试用例及说明	预期结果	实际结果
管理员登录，选择交易费率选项卡，点击查询按钮	页面显示深交所和上交所的交易费率信息	正常
管理员登录，点击新增按钮	页面弹出新增交易费率弹框	正常
管理员登录，在新增交易费率弹框上填写合理的信息，点击提交按钮	页面上会显示新增交易所的交易费率信息，并显示新增成功提示，数据库中也会同步增加	正常
管理员登录，选中特定的交易费率信息，点击修改按钮	页面弹出修改交易费率弹框	正常
管理员登录，在修改交易费率弹框中输入所需要修改的信息，点击提交按钮	页面提示修改成功，数据库中的信息也会同步刷新	正常
管理员登录，选中特定的交易费率信息，点击删除按钮	页面提示删除成功，数据库中也会同步删除	正常

交易管理和结算管理的测试过程类似，均是对相关数据的增删改查，这里以交易费率测试为例，管理员点击删除按钮，界面会提示是否删除当前选中数据。管理员点击新增按钮，界面会弹出新增交易费率弹框，若输入数据有误，界面弹出相应提示。交易费率测试界面如图 5.13 所示。



(a)删除交易费率界面



(b)新增交易费率弹框

图5.13 交易费率测试界面

5.3 系统性能测试

系统的整个性能测试是都是在测试环境中进行的，由于系统的所有功能都是在APP 上显示的，而每日使用 APP 的用户数量巨大，因此对系统的并发能力提出了很高的要求，为了保证用户的使用体验，必须对系统提供的接口进行完备的性能测试。

测试接口性能选用的是 ApacheBench。ApachBench 是 Apach 服务器自带的一个Web 压力测试工具，简称 ab。ab 是一个命令行工具，根据设置参数的差异模拟数量不同的访问者对同一个接口地址进行访问。ab 工具小巧简单，可以提供需要的指标，因此成为并发测试的首选工具。命令格式如表 5.12 所示。

表5.12 ApacheBench 命令格式

ab -n 总请求数 -c 模拟并发数 测试的接口地址

测试接口的平均延时，使用 ab 对系统的常用接口进行测试，每一个接口都会执行 1 万次请求，每次测试 10 次，总计 10 万次请求，具体的测试结果如表 5.13 所示。

表5.13 接口延时测试结果表

测试接口名称	平均延时(ms)
获取新闻热点	20
获取股票行情	13
获取 K 线走势	25
获取资产详情	40
撤销委托	31
买卖请求	43
平均	28.6

测试接口的并发能力主要是通过 ab 来测试每个接口的每秒查询率（Query Per Second，QPS）。对于每个接口执行 1 万次请求，每次测试 10 次，总计 10 万次请求，并发数量分为 100、200、500、800，对应的测试结果如表 5.14 所示。

表5.14 接口并发数测试结果表

QPS 并发数 / 接口	100	200	500	800
获取新闻热点	500	660	780	988
获取股票行情	410	616	682	696
获取 k 线走势	400	580	622	768
获取资产详情	386	562	596	604
撤销委托	338	472	544	479
买卖请求	304	410	502	470

5.4 测试结论

通过对系统的各个模块进行测试，没有发现重大的功能缺陷。由接口延时测试结果表和接口并发数量测试结果表中可以看出，股票行情的平均延时为 13ms，系统主要接口的平均延时为 28.6ms，符合时间性能要求。在对接口使用 ab 工具进行并发压力测试的时候，单机每秒处理请求也均在 300 以上，以此同时，并没有出现接口无法

返回数据、服务器宕机等问题，能够正常的提供服务，保持了优秀的稳定性。

5.5 本章小结

本章主要对系统的测试方案进行描述。在功能测试中对系统的各个功能模块进行测试，保障各个功能点不会出现偏离系统要求的情况出现。在性能测试中使用 ab 工具对系统的主要接口进行了延时测试和并发测试，确定系统满足设计要求，在功能和性能都具有正常的表现，符合验收条件。

第六章 结束语

6.1 论文工作总结

本文在对股票交易平台的功能需求进行全面研讨的基础上，在充分的调研了市面上主流的软件架构的前提下对本系统进行设计研发。系统既能够满足用户在交易客户端的委托交易，还能够为用户提供热点新闻浏览、相似 K 线查询等功能，与此同时，也保证了平台能够对用户的委托过程进行有效管理。下面对本文的研究内容做简单的分析总结：

首先，对行业内相关的公司进行研讨和调研，充分的吸收在系统结构和功能设计上的优点，在对股票交易平台进行功能需求设计的基础上，从系统可行性和技术可行性等多个角度对系统进行可行性分析，从而制定出适合本系统的需求说明书和可行性说明书。

其次，确定系统的技术选型和整体架构，这部分也是本论文最为核心的内容。在系统框架的选型上面，选取了当下最为稳定且技术十分成熟的 SpringMVC 框架，前台对于界面的展示和用户的交互性要求较高，因此在技术上采取了 MVVM 模式。考虑到平台后期用户量的增长以及交易量的增加，为了保障系统的稳定性和可靠性，使用了分布式任务调度框架 Quartz。在股票交易系统里面包含了大量的非结构化数据和结构化数据，同时为了达到系统对于实时性的要求，系统在存储万德行情数据服务提供商提供的股票数据方面采用了 WebSocket 长连接请求，数据存储方面采用了 Redis 和 MySQL 相结合的设计方案。

最后，是对股票交易系统的实现和测试。在充分理解每一个功能模块的具体业务逻辑的基础上，结合序列图和类图对具体功能的代码实现过程进行阐述。然后通过功能测试和性能测试，保证系统在功能上不会出现业务逻辑错误，在性能上能够保障系统在上线之后的要求。本文的测试结果表明股票交易系统性能稳定，功能良好，符合大多数用户的要求，达到了系统最初的测试预想。

6.2 展望

本文虽然为了保证系统实时性的要求完成了基于 Redis 的股票交易系统，但是考虑到未来用户的数量增长的不可预测性，无疑会对系统的业务逻辑和系统响应方面提出更高的要求，因此系统不可避免仍有一些不足之处，结合本文的工作，还需要从以下几个方面进行改进完善：

（1）用户体验对于一款互联网产品是十分重要的，产品应该在用户的使用感受上投入更多地关注。在系统的交易客户端，在界面和表现形式上面应该更加的多样化，

当出现系统提示的时候，提示信息应该更友好。

（2）项目开发周期时间较短，注重开发速度的代价就是代码的质量差强人意，没有对其进行严格的把控，致使系统中出现较多的冗余代码。接下来应该在不影响原有代码功能的前提下，对代码进行重构，提升代码的质量。

（3）在对系统进行架构设计的初期没有考虑到后来用户量的激增问题，只是对初期的用户量进行了保守的估计。后期随着用户量的增多，由于现有的系统架构承载能力有限，很可能会出现页面响应迟缓、交易失败等故障现象。系统在处理交易时，指令队列的数量采用的是合约数量 mod10 的结果，合约数量就会会随着用户量的增多而增多。接下来打算采用生产者和消费者模式对系统予以改进，提高系统的扩展性。

参考文献

[1] 吴红松, 李高平. 网上证券交易: 未来证券业的发展方向[J]. 大庆社会科学, 2003 (5): 39-41.

[2] 王吉斌, 彭盾. 互联网+[J]. 机 械 工 业 出 版 社, 2015, 4.

[3] 徐国风. 实时数据库关键技术研究[D]. 西安: 西安建筑科技大学, 2006.

[4] 李子骅. Redis 入门指南[M]. 人民邮电出版社, 2013.

[5] 黄健宏 .Redis 设计与实现 [M]. 北京 ：机械工业出版社，2014.

[6] 张胜兰. 内存数据库及其对外接口[D]. 山东大学, 2011.

[7] 李凤雨. 高频交易对证券市场的影响及监管对策[J]. 上海金融, 2012 (9): 48-52.

[8] 于明. 欧美高频交易监管对我国的启示[J]. 海南金融, 2015 (12): 55-56.

[9] 宋逢明, 江婕. 中国股票市场波动性特性的实证研究[J]. 金融研究, 2003 (4): 13-22.

[10] 任世宗, 李润知, 张茜等. 基于 Nginx 的可扩展负载均衡 Web 站点部署[J]. 中国教育网络, 2014 (8): 76-78.

[11] 文庭孝, 李维. 大数据环境下数字资源融合初探[J]. 信息资源管理学报, 2015, 5(2): 79-84.

[12] 崔伟, 周泉. 高并发重负载网站架构浅析[J]. 科技创新与应用, 2013 (12): 38-38.

[13] 谢鹏. 分布式数据库存储子系统设计与实现[D]. 电子科技大学, 2013.

[14] Shashank Tiwari. 深入 NoSQL[M]. 北京：人民邮电出版社, 2012.

[15] 徐竟州. 基于 Redis 的高并发抢红包应用的设计与实现[D]. 湖南大学, 2016.

[16] Mao Y, Kohler E, Morris R T. Cache craftiness for fast multicore key-value storage[C].Proceedings of the 7th ACM european conference on Computer Systems. ACM, 2012: 183-196.

[17] 潘洁. 基于内存数据库的分布式数据库架构[J]. 信息与电脑, 2016 (13): 168-169.

[18] 赖歆. 基于 Redis 的系统缓存容量平滑扩展方案[J]. 网络安全技术与应用, 2014 (10): 78-79.

[19] 李国平, 陈森发, 李新平. 基于 K 线理论的股票灰色预测方法研究[J]. 郑州航空工业管理学院学报: 管理科学版, 2004, 22(4): 62-64.

[20] 张闯, 王婷婷, 孙冬娇等. 基于欧氏距离图的图像边缘检测[J]. 中国图象图形学报, 2013, 18(2): 176-183.

[21] Keogh E, Ratanamahatana C A. Exact indexing of dynamic time warping[J]. Knowledge and information systems, 2005, 7(3): 358-386.

[22] 张鹏, 白朝旭, 王锟等. 基于 Quartz 的集团化调度任务分布部署研究[J]. 现代电子技术, 2014, 37(2): 80-83.

[23] 刘志鹏, 卫晨. 基于 Quartz 与 Spring 的动态任务调度系统的设计与实现[J]. 计算机光盘软件与应用, 2014, 17(13): 263-264.

[24] 赖晓京. 基于 Html5 WebSocket 的即时通讯系统[J]. 电子技术与软件工程, 2013 (17): 61-61.267-269.

[25] Pimentel V, Nickerson B G. Communicating and displaying real-time data with websocket[J]. IEEE Internet Computing, 2012, 16(4): 45-53.

[26] 杨艳, 李炜, 王纯. 内存数据库在高速缓存方面的应用[J]. 现代电信科技, 2011, 41(12): 59-64.

[27] Kumar V, Burger A. Performance measurement of main memory database recovery algorithms based on update-in-place and shadow Approaches[J]. IEEE Transactions on Knowledge and Data Engineering, 1992, 4(6): 567-571.

[28] Greenspan J, Bulger B. MySQL/PHP database applications[M]. John Wiley & Sons, Inc., 2001.

[29] 张飞, 姜进磊, 李庆虎. 利用 MySQL 构建分布式应用[J]. 计算机工程与应用, 2001, 37(18): 102-104.

[30] 李海兵. 基于 MySQL 的负载均衡的搭建与研究[J]. 信息安全与技术, 2011 (5): 44-45.

[31] 陈共, 周升业, 吴晓求. 证券市场基础知识[J]. 1998.

[32] Rosenberg J, Schulzrinne H, Camarillo G, et al. SIP: session initiation protocol[R]. 2002.

[33] He D, Göker A, Harper D J. Combining evidence for automatic web session identification[J]. Information Processing & Management, 2002, 38(5): 727-742.

[34] 王珊, 肖艳芹, 刘大为. 内存数据库关键技术研究[J]. 计算机应用, 2007, 27(10): 2353-2357.

[35] 仲, 系统工程, 君友. 系统测试性设计分析与验证[M]. 北京航空航天大学出版社, 200.

致谢

时光荏苒，白驹过隙，转眼间三年研究生的学习生涯就已经接近尾声。在这两年多的时间里，有欢笑，也有泪水，欣慰的是自己在学习上取得了长足的进步，正是有了老师、同学、亲人的陪伴，才能让我在面对困难和挫折的时候变得从容，正因为有你们我的生活才会变得精彩，在这里请允许我向你们表达最真挚的感谢，谢谢你们的一路教导和陪伴。

首先，我需要感谢我的论文指导老师鲍亮副教授，虽然老师远在美国出差，但是时时刻刻不忘关心我的论文进展，对于在论文写作中遇到的问题，老师也总能在百忙之中抽出时间予以指导。虽然与鲍老师的接触时间不是很长，但是在仅有的接触机会中能够深切地感受到老师对学生的关心。鲍老师无论是在科研方面还是为人处世方面值得我们每一个学生学习。

其次，在实习工作中，非常感谢西北工业大学张小芳老师的耐心指导和热情帮助。在张老师耐心的指导下，我不仅顺利的融入到团队开发当中，而且还受到了同事的一直好评，自己的开发水平以及解决问题的能力也得到了很大的提高。张老师一丝不苟的工作态度和深厚的技术功底深深地感染了我，成为我以后踏入社会学习的榜样。

然后，感谢我的室友，是你们，让我在生活上遇到困难的时候给予我力所能及的帮助。感谢我的家人，正因为有了你们，才能让我一路上奋勇向前，不断进步，人生的每一次进步都离不开你们无私的关爱。感谢同一项目组的同学，让我在实习期间收获了满满的友谊。

最后，由衷的感谢对我的论文进行评审的老师们，你们辛苦了。

西安電子科技大學
XIDIAN UNIVERSITY

地址：西安市太白南路2号
邮编：710071
网址：www.xidian.edu.cn