Design and Implementation of Flight Inquiry and Booking System Based on Microservice Architecture

Wei Li

College of Information Wuhan Vocational College of Software and Engineering Wuhan, China

Abstract. This paper presents the design and implementation of Flight Inquiry and Booking System Based on Microservice Architecture. The system built on the Spring Cloud Alibaba and Kubernetes framework, not only provides basic flight inquiry and ticket booking functions, but also performs well under high concurrency and responds within 1 second. The system can expand or shrink its service scale according to traffic needs during actual deployment.

Keywords: design; implementation; System; Microservice; framework.

1. Preface

Flight inquiry and ticket booking are one of the core businesses of the civil aviation industry. Since the establishment of the computerized ticketing system in civil aviation, the core business of flight inquiry and ticket booking has always been implemented through mainframes, but large mainframes are expensive and have limited capacity, unable to meet the rapid development needs of the civil aviation industry. Therefore, the design goal of this system is to develop a highly available, scalable, and flexible deployment flight inquiry and booking system, based on the microservice architecture of Spring Cloud Alibaba and Kubernetes. The system not only implements the basic functions of flight inquiry and ticket booking, such as ticket modification and refund, but also can adapt to high concurrency in performance and respond within 1 second. At the same time, by using Kubernetes for resource scheduling, the system can expand or shrink its service scale according to traffic needs during actual deployment.

2. System Development Technology Specification

2.1 development of technology options

The following technologies are used:
1) Frontend
- HTML+CSS+JavaScript: Frontend Three-Piece Suit
- Vue2.0: Progressive framework
- ElementUI: Component library based on Vue
- Vue Router: Vue routing component
- VueX: Vue state management
- Axios: Promise-based HTTP library
- Node.js+webpack: Project build tool
- Nginx: Web reverse proxy server

2) Backend
- SpringBoot: Project rapid development framework.
- SpringCloud: Microservice framework
- SpringBoot Admin: Microservice state and log management
- XXL-JOB: Distributed task scheduler

3) Middleware
- MybatisPlus: Persistence layer framework
- RocketMQ: Message queue
- Mysql: Relational database
- Redis: Cache database

- MongoDB: Distributed file storage database
- Sentinel: High availability traffic protection component
- Nacos: Service configuration center

4) Operation and maintenance

- Docker: Application container engine
- Kubernetes: Container orchestration engine
- Shell Script: Automated deployment script

2.2 The architecture of the service

The system's microservice architecture adopts the Spring Cloud system, using Spring Cloud's lightweight components combined with Alibaba's mature components to build a secure and highly available microservice architecture.

3. System Requirement

3.1 Functional requirement analysis

Based on the preliminary requirement research and project background analysis, the system is divided into two business systems: the front-end flight inquiry and booking system and the back-end management system. The specific functional requirement list is as follows.

Table 1. Functional requirement

Business System	Functional Requirements	Requirement Description
Flight Inquiry and Booking System	Member Registration	Visitor users register and enter account information, login password, mobile number, passenger information, address information, and other member user information.
	Members Login	Members enter their username and password to login to the flight inquiry and booking system.
	Flight Inquiry	Members view a list of flights and search for flights using one-way, round-trip, or multi-segment modes according to their needs.
	Flight Booking	Members perform a booking operation and check the result of whether the booking is successful.
	Flight Change and Refund	Members can change or cancel their booking based on their own reasons.
Back-end Management System	Member Management	Manage member information: (1) Basic information management: maintain basic member information (2) Address information management: view member address information to assist in address information management (3) Passenger information management: view member passenger information to assist in passenger information management
	Flight Management flight information	Route management: display all flight information Hot city management: configure hot city information Connecting flight management: configure connecting flight information Connected flight management: configure connected
	Change and Refund Management	Ticket Refund Review: view and review orders that do not meet the fast refund criteria Change of flight review: view and review orders that do not meet the fast change of flight criteria.

3.2 Performance Requirements Analysis

Software quality includes attributes such as functionality, performance, security, usability, reliability, portability, compatibility, fault tolerance, and maintainability. The system is implemented in a microservice manner to provide flight inquiry and booking functions. The system should have the ability to monitor and alarm its own operating status and provide a corresponding visual interface. A visual configuration interface should be provided for the abilities of circuit breaker, flow control, and gray-scale in non-functional requirements. The non-functional attributes mainly focus on the system's ability to handle concurrency and limit the flow of requests when the access volume exceeds the system's capacity, as well as the ability to melt down abnormal microservices. The system should also support gray-scale traffic splitting strategies of at least two dimensions. The specific requirements are as follows:

a) The system should have the ability to handle concurrent inquiries from more than 500 people and concurrent bookings from more than 50 people.

b) The system should ensure that the response time in high-concurrency query and booking scenarios is within 1 second.

c) The system should have good request response time and throughput and should be tested using dedicated performance testing tools.

d) Usability: The system is user-friendly, and the pages are simple and easy to use. For example, executing a flight inquiry only requires clicking a button.

e) Compatibility: The front-end of the flight inquiry and booking system should support access from all major browsers.

f) Maintainability: Refers to the difficulty of modifying software to fix defects, add features, and improve quality. The system should be able to record operational and system logs to help troubleshoot system errors. The system runs on a kubernets environment and supports visual service instance expansion, service deployment, and other capabilities.

3.3 Other Functional Requirements

a) Some functional requirements of the system are not explicit, and there are also some implicit requirements. Other functional requirements are as follows:

b) The system page should provide input validity check, validate incorrect or non-compliant inputs with business logic, such as dates, times, names, ID numbers, passwords, etc. Illegal inputs should have error prompt, instead of causing system page crash.

c) During flight booking, if the user stays in the booking success but not yet payment success, the flight ticket inventory should be reasonably reduced.

d) To avoid users to place duplicate orders for the same flight, to avoid overselling of flight tickets, etc.

4. Database Design

Database design refers to the process of creating and organizing a database to store, manage and retrieve data effectively. It involves defining the data structure, relationships between tables, data constraints, and database security measures. The goal of database design is to create a database that is flexible, scalable, and easy to maintain, while ensuring data integrity and security. Effective database design is critical for the efficient and effective operation of a system.

The system designs the data tables in different fields in different databases.

4.1 MySQL database

MySQL database can be expanded or contracted according to the specific business pressure to achieve high availability of the service.

1) Database Connection Pool

The performance of accessing the MySQL database is improved by using the database connection pool HikariCP.

2) Relational Database Table Design

a) System Database: Contains tables related to the backend system.

b) Gray Service Database: Gray service configuration table.

c) Nacos Database: Tables related to the Nacos service dependencies.

d) Distributed Scheduling Task Database: Tables related to the dependencies of the distributed scheduling task service.

e) Flight Database: Tables related to flight routes, cities, flight plans, and flight inventory.

f) Member Database: Tables related to member basic information, passenger information, and address information.

g) Order Database: Tables related to orders and order-passenger information.

4.2 MongoDB database

MongoDB database can provide faster query services for users and avoid performance sacrifice from multi-table connection queries in MySQL, making it more suitable for high-concurrency query services. In this system, because flight query is the main function and its frequency of use is relatively high, MongoDB database is used to provide query services. Flight data in MongoDB is generally generated by the flight data manager based on flight route data during daily scheduled tasks.Related Document Design:

a) Airport Information Document: Stores airport information for front-end system fuzzy queries.

b) Flight Number Document: Stores existing flight number information for front-end system fuzzy queries.

c) Flight Plan Document: Stores existing flight plan information for front-end system queries.

5. Function Module Design and Implementation

5.1 Function Module Architecture

The system's functional structure is divided into modules based on the system's functional requirements analysis, and then integrated into various sub-modules. The main sub-modules include the Member Basic Data Management Module, the Flight Query and Booking Module, the Order Rescheduling Application Module, and the Order Refund Module.

5.2 Design and Implementation of Main Function Modules

a) Design and Implementation of Member Basic Data Management Module

The system's Member Basic Data Management Module includes sub-modules such as Member Basic Information Management, Member Passenger Information Management, and Member Address Information Management.

b) Design and Implementation of Flight Query and Booking Module

The system's Flight Query and Booking Module includes sub-modules such as One-Way Flight Query, Round-Trip Flight Query, Multi-Segment Flight Query, Flight Number Query, and Flight Booking.

c) Design and Implementation of Order Rescheduling Application Module

The system's Order Rescheduling Application Module includes sub-modules such as Order Information Query, Flight Information Query, Order Rescheduling Application, and Air Ticket Order Audit Management.

d) Design and Implementation of Order Refund Module

The system's Order Refund Module includes sub-modules such as Order Information Query and Order Refund Application.

6. Business Analysis

Table 2. Functional requirement

Serial number,	Business process	Process description
1	Login and registration process:	Members perform login, the system verifies and successfully logs in to the personal center page.
2	Flight Inquiry Process	Members can search for flights under three modes (one-way, round trip, multiple segments), if the system has information that meets the criteria, it will display the relevant information for users to view details and make a reservation.
3	Flight Booking Process	Users can book the flights they have searched for, and can select multiple passengers during the booking process. The process ends when the booking is successful and payment is completed.
4	Member Refund Process	Orders that have not been checked in can be refunded, and service fees will be charged to the user during this process.
5	Member Rescheduling Process	Orders that have not been checked in can be rescheduled, and service fees will be charged to the user during this process.

Functional requirements are specifications of what a system, software, or product should do. They describe the functions, features, and behaviors of the system, and provide a basis for testing to determine if the system meets the desired requirements. They are typically expressed as statements of what the system must do in order to fulfill its intended purpose and meet the needs of the stakeholders. The analysis of the functional requirements of the system is as follows:

6.1 The overall software architecture.

The system adopts a front-end and back-end separation structure, with the front-end using the Vue framework and the back-end using the Spring framework. After the front-end sends the request, the Controller receives the request and calls the Service for processing, and the processed data is persisted by calling the Mapper or Repository. The architecture is shown in Figure 1.

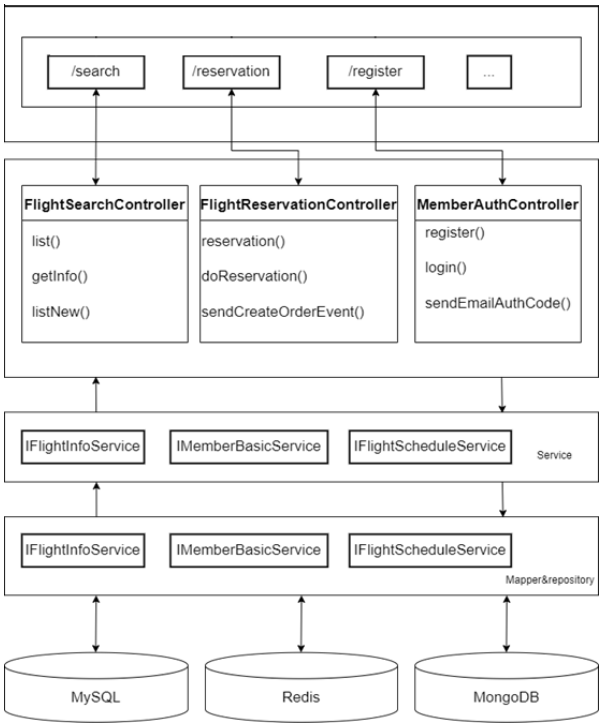


Figure 1. Overall Software Architecture

7. System implementation

The following is part the logic code.

```
/**
 * Agree to the flight itinerary change request
 */
@RequiresPermissions("order:change:agree")
@Log(title = " Agree to the flight itinerary change ", businessType = BusinessType.UPDATE)
@PutMapping("/agree")
public AjaxResult agree(@RequestBody ChangeBody changeBody) {
    OrderInfo orderInfo = orderInfoService.getById(changeBody.getOrderDataId());
    orderInfo.setChangeDepartureTime(null);
    orderInfo.setChangeArrivalTime(null);
    orderInfo.setChangeClass(null);
    orderInfo.setChangePrice(null);
    orderInfo.setChangeTime(null);
    orderInfo.setRemark(null);
    orderInfo.setChangeStatus("C");
    orderInfoService.updateById(orderInfo);
    MGReservation reservation =
mgReservationRepository.findById(orderInfo.getSerialNumber()).get();
  FlightSchedule flightSchedule = reservation.getFlightSchedule().getFlightSchedule();
      flightSchedule.setStartTime(changeBody.getStartTime());
    flightSchedule.setEndTime(changeBody.getEndTime());
reservation.getFlightSchedule().setFlightSchedule(flightSchedule);
reservation.setOrder(JSONObject.parseObject(JSONObject.toJSONString(orderInfo)));
    mgReservationRepository.save(reservation);
    return AjaxResult.success();
}
```

8. Summary

Flight Inquiry and Booking System Based on Microservices Architecture is implemented using Spring Cloud Alibaba and K8s. The system built on the Spring Cloud Alibaba and K8s framework, not only provides basic flight inquiry and ticket booking functions, but also performs well under high concurrency and responds within 1 second. Using Kubernetes for resource orchestration, the system can scale service according to traffic needs and is a highly available and flexible deployment solution for flight inquiry and booking of various sizes.

References

[1] Barua B, Whaiduzzaman M, Mesbahuddin Sarker M, et al. Designing and Implementing a Distributed Database for Microservices Cloud-Based Online Travel Portal[M]//Sentiment Analysis and Deep Learning: Proceedings of ICSADL 2022. Singapore: Springer Nature Singapore, 2023.

[2] Debski A, Szczepanik B, Malawski M, et al. A scalable, reactive architecture for cloud applications[J]. IEEE Software, 2017, 35(2).

[3] Kookarinrat P, Temtanapat Y. Design and implementation of a decentralized message bus for microservices[C]//2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE). IEEE, 2016.

[4] Alshuqayran N, Ali N, Evans R. A systematic mapping study in microservice architecture[C]//2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA). IEEE, 2016.

[5] Wizenty P N, Rademacher F, Sorgalla J, et al. Design and implementation of a remote care application based on microservice architecture[C]//Software Technologies: Applications and Foundations: STAF 2018 Collocated Workshops, Toulouse, France, June 25-29, 2018, Revised Selected Papers. Cham: Springer International Publishing, 2018.