CSIT6910D - Independent Project

The Final Report of Project

Development of Real Time Location Service System Based on Kafka

Student: XUE Hongbin

Student ID: 20922852

Supervisor: Professor Gary S H CHAN

Date: 2023/12/13

TABLE OF CONTENT

1. Acknowledgment

Firstly, I would like to express my deepest gratitude to my project supervisor Professor Gary S H CHAN for giving me this opportunity to carry out this independent project and for his continuous support throughout the project.

Secondly, I would like to thank Professor Gary S H CHANi's doctoral student HU Siyan for her assistance and constructive feedback on this independent project. Under her guidance, I confirmed my experimental direction and significantly accelerated the progress of my experiment. Meanwhile, her patient guidance and supervision ensured the quality of my project.

Finally, I would like to thank my parents and grandpa, who suffered from cancer last year and this year respectively. I should have taken more care of them and accompanied them, but they have kept me focused on my studies and always supported me.

Your support and guidance have played an essential role in completing this project, and I am deeply grateful.

2. Abstract

In the digital age, real-time location services are crucial for various applications, from navigation and location search to logistics and mobile advertising. This project has developed a real-time location service system based on Kafka, aimed at processing and responding to a large amount of real-time location data. This project is developed using Java and Spring Boot frameworks, integrating MySQL and Redis as data storage solutions and implementing efficient data stream processing through Kafka. This project has successfully received and processed various types of real-time data, such as GPS, Wi-Fi SSID+RSSI, and IMU data, from multiple sources, providing real-time map services and real-time response to user location data. Tests have shown that the system has high concurrency processing capabilities and can meet the needs of commercial-level applications.

3. Introduction

With the development of mobile technology and location-aware devices, real-time location services have become essential to the contemporary technology ecosystem. These services play a crucial role in multiple fields, including but not limited to logistics tracking, personal navigation, and location-based advertising. A system that can efficiently process and respond to large-scale real-time data is needed to support these applications.

This project aims to develop a real-time location service system based on Kafka, which can not only process a large amount of location data from various devices but also respond to user requests in real-time, such as providing the latest map information. By integrating Kafka, the system can efficiently manage data streams, ensuring real-time data processing and transmission. In addition, this project focuses on scalability and maintainability in design and implementation, adopting a modular architecture that combines popular technology stacks such as Spring Boot, MyBatis, MySQL, and Redis. This report will provide a detailed introduction to the design, implementation, and testing process of this project.

4. Technology Overview

(1) Spring Boot

Spring Boot is an open-source Java infrastructure framework that significantly simplifies Spring-based application development. By providing a quick set of default settings for configuration, Spring Boot makes creating production-level Spring applications simple and straightforward. Spring Boot is the main framework in this project, providing convenient dependency management and automatic configuration functions, significantly reducing configuration code and development time. Its automatic configuration function is suitable for database connections, message queue management, and RESTful service development, making the entire system structure clear and easy to test and deploy.

(2) MyBatis-Plus

MyBatis Plus is an enhancement tool for MyBatis, used to simplify and enhance the use of MyBatis. It provides powerful features such as automatic code generation, pagination, and performance analysis. In our project, MyBatis Plus played a core role, not only simplifying database operations but also improving development efficiency and data query performance. Especially its dynamic SQL capability makes implementing complex queries simpler and more intuitive.

(3) MySQL

MySQL is a widely used relational database management system known for its stability and efficiency. In this project, MySQL is responsible for storing core data such as location data and user information. Combining MySQL with MyBatis Plus makes MySQL's data operations more efficient and flexible.

(4) Redis

Redis is a high-performance key-value pair database commonly used for data caching and message queuing. In this project, Redis is used to cache frequently accessed data, such as real-time location information of users, to alleviate database pressure and improve system response speed. The high-speed data reading and writing

capability of Redis is crucial for processing large-scale real-time data, especially in high concurrency scenarios.

(5) Kafka

Apache Kafka is a distributed streaming platform primarily used for building real-time data pipelines and streaming applications. It can efficiently process large amounts of data and support multiple consumers. Kafka plays a crucial role in this project as the main channel for data flow, responsible for processing real-time location data from different sources. Kafka's high throughput and scalability enable the system to collect, process, and distribute large amounts of location data in real-time, ensuring the real-time and reliability of the system.

Through the comprehensive application of these technologies, our project can efficiently process large-scale real-time data and maintain the system's scalability and maintainability. Each technology has played its unique role in jointly building a stable and efficient real-time location service system.

5. Project Architecture Overview

This project has been layered based on the hierarchical architecture of *Alibaba's coding standards* (Figure 1), with each layer having precise functions. The following is a detailed description of the project architecture:

(1) Entity Layer

Entity Definition: The entity classes in the project (such as the Location class located in the “lbs-shared” module) are directly mapped to the database's table structure.

Function: The entity layer is mainly responsible for defining models corresponding to database tables, including fields, data types, etc.

(2) DAO Layer

Data interaction: In the “lbs-service” and “lbs-shared(Mapper)” modules, data access objects (DAO) are implemented through MyBatis-Plus integration.

Function: This layer focuses on performing database operations such as querying, inserting, updating, and deleting data.

(3) Service Layer

Business logic: The “lbs-service” module contains service interfaces and implementations, such as LocationService and LocationServiceImpl.

Function: The service layer includes the core business logic of the project, handles requests from the control layer, and interacts with the data access layer.

(4) Controller Layer

Request processing: In the “ lbs-admin ” module, controller classes such as LocationController are responsible for handling external HTTP requests.

Function: The control layer is responsible for coordinating the interaction between user input and the service layer and is the front end of user-system interaction.

(5) Message queue layer (Kafka)

Data flow management: Kafka is a middleware to handle real-time data flows from multiple sources.

Function: Kafka consumers in lbs admin are responsible for extracting data from Kafka and passing it on to the service layer for further processing.

(6) Redis caching layer

Data caching: Redis is used to cache frequently accessed data, such as hotspot location information.

Function: Improve data retrieval speed and reduce access pressure to the central database.

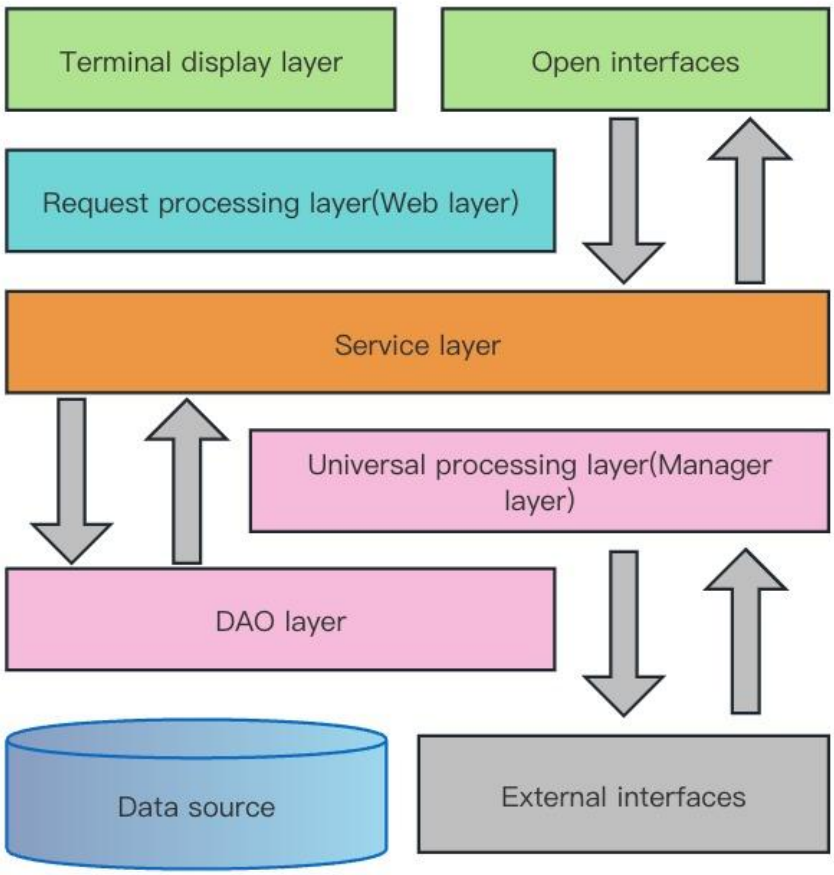


Figure 1 Alibaba's coding standards

6. Core Business Processes

This section will introduce the core business process of this project, which is how real-time location services are implemented. The flowchart is shown in Figure 2.

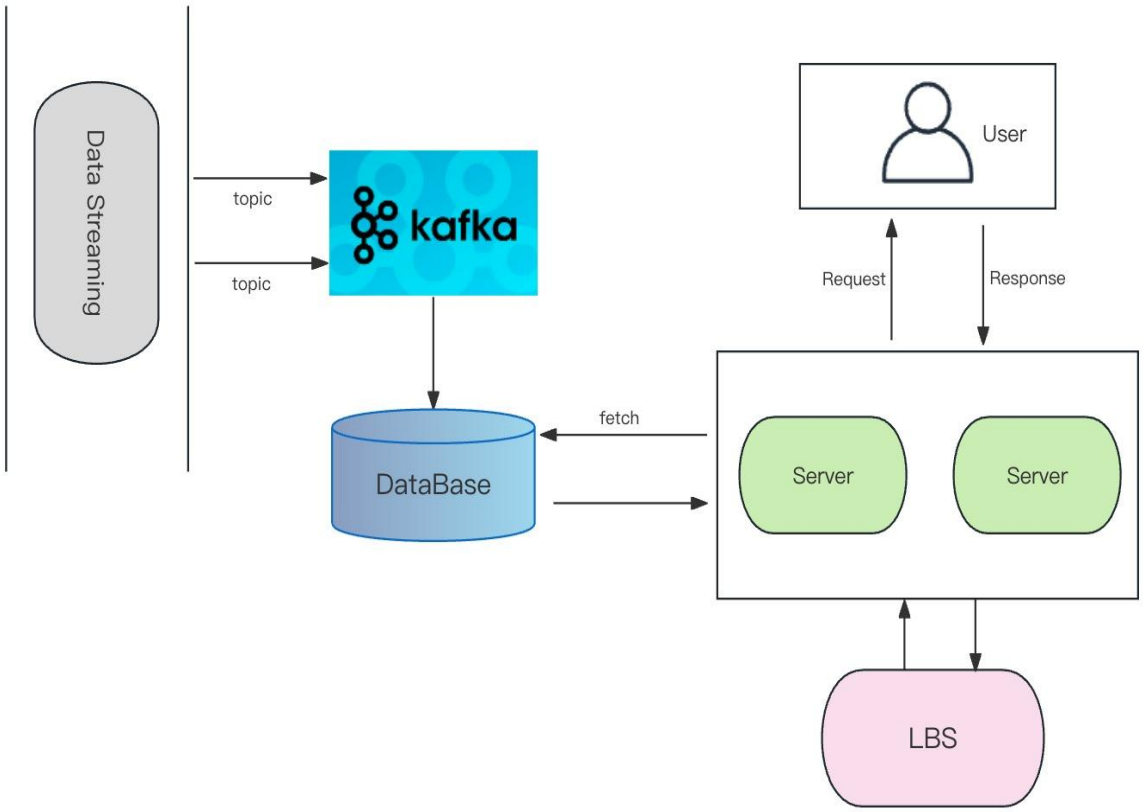


Figure 2 The flowchart of core business

Its detailed implementation can be divided into the following stages:

Stage 1: Data Production

(1) Data generation: The user's location data will be generated, which will be implemented by various client devices (such as smartphones, GPS devices, etc.), responsible for collecting real-time location data of the user, such as longitude and latitude, timestamps, etc.

(2) Data release: As Kafka producers, these devices will publish the collected location data to Kafka. Kafka acts as a message queue, responsible for receiving data streams from various sources (such as devices of different users).

(3) Data Streaming: For the convenience of testing, this project uses data streaming (DataStreaming class) in the "lbs-admin" module to simulate the generation of user location data. As shown in the code in Figure 3 and Figure 4, we randomly generate 50 position data, then specify that the longitude and latitude increase by 0.01 each time, and finally send it to Kafka to simulate changes in the user's position.

```java
// 生成一个随机的用户ID
String userId = UUID.randomUUID().toString();
// 初始经纬度，仅为测试所任意提供的一个合理数据
double longitude = 116.407;
double latitude = 39.9042;
// 生成50个位置数据并发送到Kafka
for (int index = 0; index < 50; index++) {
    IosLocationDto locationDto = new IosLocationDto();
    locationDto.setUserId(userId);
    locationDto.setLongitude(longitude);
    locationDto.setLatitude(latitude);
    locationDto.setSsid("WIFI");
    locationDto.setRssi(70.0);

    ImuDto imuDto = new ImuDto();
    imuDto.setX(1.0);
    imuDto.setY(1.0);
    imuDto.setZ(1.0);
    locationDto.setImu(imuDto);
    locationDto.setTimestamp(new Date());

    // 创建Kafka的生产者记录
    // 第一个参数是Kafka主题的名称，消息将被发送到名为"ios-location"的Kafka主题。
    // 第二个参数是消息的值，即locationDto。这个对象包含了要发送的实际数据。
    ProducerRecord<String, IosLocationDto> producerRecord =
            new ProducerRecord<>( topic: "ios-location", locationDto);
    // 发送记录到Kafka
    producer.send(producerRecord);

    // 更新经纬度，每次各自增加0.01
    longitude += 0.01;
    latitude += 0.01;
}
```

Figure 3 Simulate generation of location data

对象 location@kafka_lbs (res...

id	user_id	device	longitude	latitude	ssid	rssi	imu	timestamp
65	07738498-afa0-4408·	IOS	116.407	39.9042	WIFI	70	{"x":1.0,"y":	2023-11-2
66	07738498-afa0-4408·	IOS	116.417	39.9142	WIFI	70	{"x":1.0,"y":	2023-11-2
67	07738498-afa0-4408·	IOS	116.427	39.9242	WIFI	70	{"x":1.0,"y":	2023-11-2
68	07738498-afa0-4408·	IOS	116.4370000	39.9342	WIFI	70	{"x":1.0,"y":	2023-11-2
69	07738498-afa0-4408·	IOS	116.4470000	39.9441999	WIFI	70	{"x":1.0,"y":	2023-11-2
70	07738498-afa0-4408·	IOS	116.4570000	39.9541999	WIFI	70	{"x":1.0,"y":	2023-11-2
71	07738498-afa0-4408·	IOS	116.4670000	39.9641999	WIFI	70	{"x":1.0,"y":	2023-11-2
72	07738498-afa0-4408·	IOS	116.4770000	39.9741999	WIFI	70	{"x":1.0,"y":	2023-11-2
73	07738498-afa0-4408·	IOS	116.4870000	39.9841999	WIFI	70	{"x":1.0,"y":	2023-11-2
74	07738498-afa0-4408·	IOS	116.4970000	39.9941999	WIFI	70	{"x":1.0,"y":	2023-11-2
75	07738498-afa0-4408·	IOS	116.5070000	40.0041999	WIFI	70	{"x":1.0,"y":	2023-11-2
76	07738498-afa0-4408·	IOS	116.5170000	40.0141999	WIFI	70	{"x":1.0,"y":	2023-11-2
77	07738498-afa0-4408·	IOS	116.5270000	40.0241999	WIFI	70	{"x":1.0,"y":	2023-11-2
78	07738498-afa0-4408·	IOS	116.5370000	40.0341999	WIFI	70	{"x":1.0,"y":	2023-11-2
79	07738498-afa0-4408·	IOS	116.5470000	40.0441999	WIFI	70	{"x":1.0,"y":	2023-11-2
80	07738498-afa0-4408·	IOS	116.5570000	40.0541999	WIFI	70	{"x":1.0,"y":	2023-11-2
81	07738498-afa0-4408·	IOS	116.5670000	40.0641999	WIFI	70	{"x":1.0,"y":	2023-11-2

Figure 4 Randomly generated data

Stage 2: Message queue (Kafka processing)

(1) Kafka pipeline: Kafka has topics set up to organize and manage these data flows. Producers (user devices) publish data to designated themes, while Kafka maintains the stability and management of the data flow.

Stage 3: Backend processing

(1) Consumer service: The Kafka consumer implemented in the "lbs-admin" module is responsible for reading data from the Kafka queue.

(2) Data processing: The received data is parsed, validated, and may be converted into a standard format. This step is completed at the service layer in the "lbs-service" module. In this project, we can specify the topic as "ios location" or "android location", as shown in Figure 5.

(3) Business logic: The service layer is also responsible for implementing the core business logic of the project, such as data aggregation, analysis, and storage. As shown in Figure 6.

```
@KafkaListener(
        topics = "ios-location",
        groupId = "ios-location",
        containerFactory = "iosLocationContainerFactory")
```

Figure 5 Specify topic

```
public interface LocationService {
    1 usage  1 implementation
    RespLocationVo getLatestLocation(String userId);

    1 usage  1 implementation
    RespMapVo getLatestMapByLocation(String userId, String device);

    1 usage  1 implementation
    List<RespLocationVo> listLocationsByUserId(String userId);
}
```

Figure 6 Service layer

Stage 4: Data Persistence

(1) Data storage: Use MySQL database for data persistence. Structured location data is stored in a database for querying and analysis. As shown in Figure 7.

(2) Data access: The MyBatis-Plus framework optimizes the interaction between backend services and databases.

Stage 5: Cache Stage

(1) Redis: To improve data retrieval speed and reduce database burden, frequently accessed data, such as hotspot location information, will be cached in Redis. As shown in Figure 7.

```java
public void ios(IosLocationDto iosLocationDto) {
    // 创建Location实体，用于保存位置信息
    Location location = new Location();
    // 设置Location实体的属性
    location.setUserId(iosLocationDto.getUserId());
    location.setDevice(DeviceEnum.IOS);
    location.setLongitude(iosLocationDto.getLongitude());
    location.setLatitude(iosLocationDto.getLatitude());
    location.setSsid(iosLocationDto.getSsid());
    location.setRssi(iosLocationDto.getRssi());
    location.setImu(JsonUtils.toJson(iosLocationDto.getImu()));
    location.setTimestamp(iosLocationDto.getTimestamp());
    // 将位置信息插入数据库
    locationMapper.insert(location);

    // 构建Redis中的缓存key
    String cacheKey = StrUtil.format( template: "location:ios:{}", iosLocationDto.getUserId());
    // 将位置信息缓存到Redis
    stringRedisTemplate.opsForValue().set(cacheKey, JsonUtils.toJson(location));
}
```

Figure 7 Store data in MySQL or Redis

Stage 6: Front-end display

(1) Map display: The backend system calls the AMap API, which can display the user's specific location on the map based on the stored location data.

(2) API integration: The front-end communicates with the back-end through RESTful APIs provided by the controller in the lbs admin module.

Simply put, the entire process involves generating location data from user devices, transmitting this data through Kafka, processing and storing the data in the backend system, and finally displaying location information in the frontend application using the Amap service API.

7. Experimental Result

This section will run the project and present and analyze the results.

Firstly, we need to ensure that the Docker images related to Kafka have successfully run and started the project, as shown in Figure 8. Then, we run the Data Streaming script (DataStream. java) to simulate the generation of user location data (as shown in Figure 9) and store it in the database, where the latest location information will be stored in Redis. As shown in Figure 10, the data flow script continuously generates position data and sends it to Kafka.

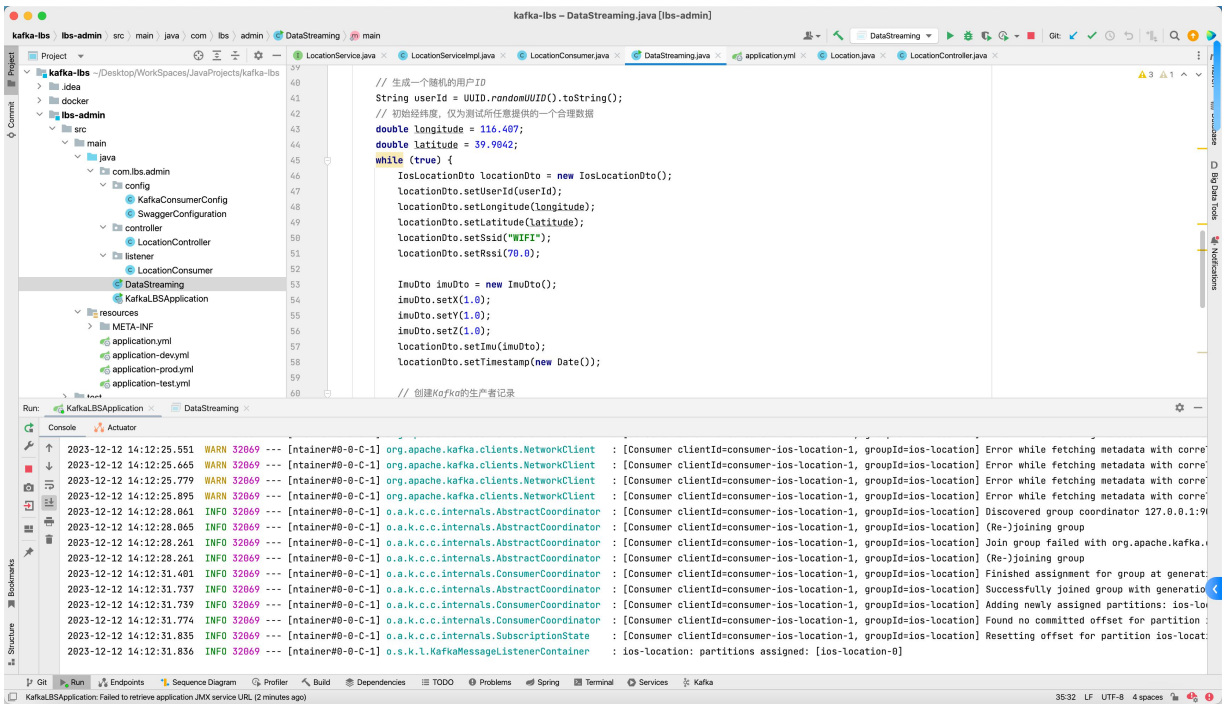


Figure 8 Running the program

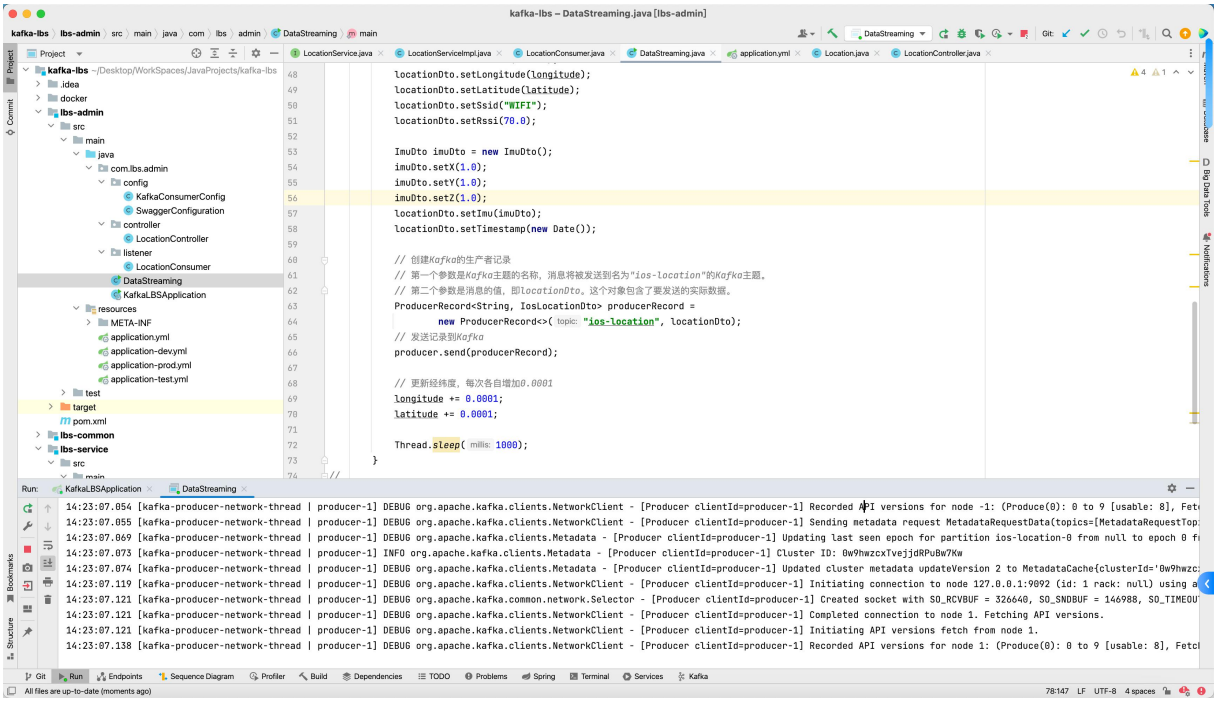


Figure 9 Running the script(DataStreaming.java)

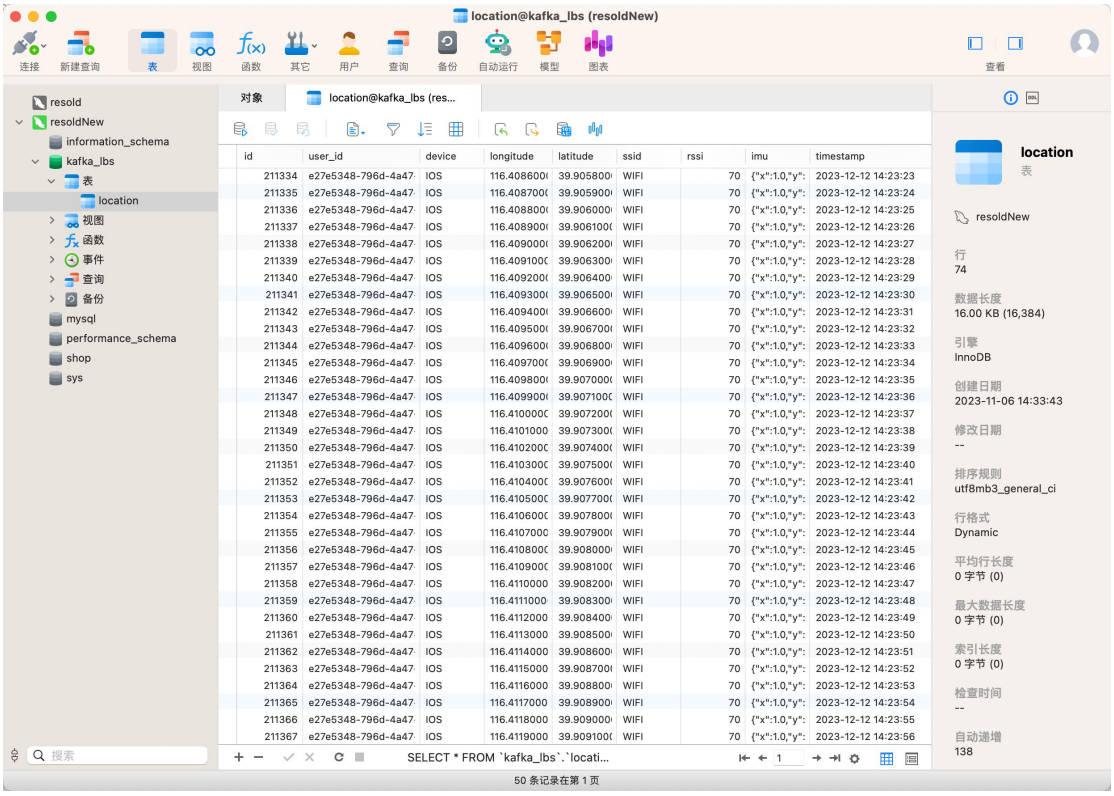


Figure 10 The generated random position data

Then, we copy the user ID in the database, successfully access the interface in the browser, and pass the user ID, and the backend calls the AMap API to generate the mapUrl. As shown in Figure 11.

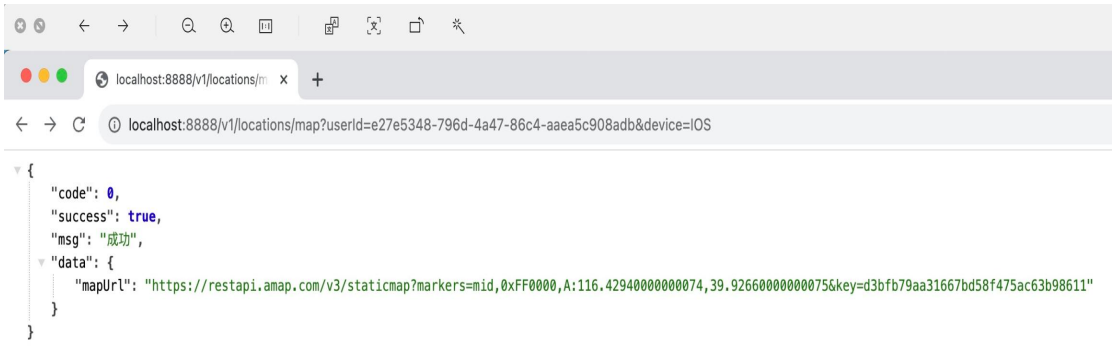


Figure 11 Get the mapUrl

By accessing the mapUrl, we can obtain the user's specific location in the map, as shown in Figure 12. When we refresh the webpage, the mapUrl has been updated with changes in user location (latitude and longitude) and moved to another location, as shown in Figure 13.

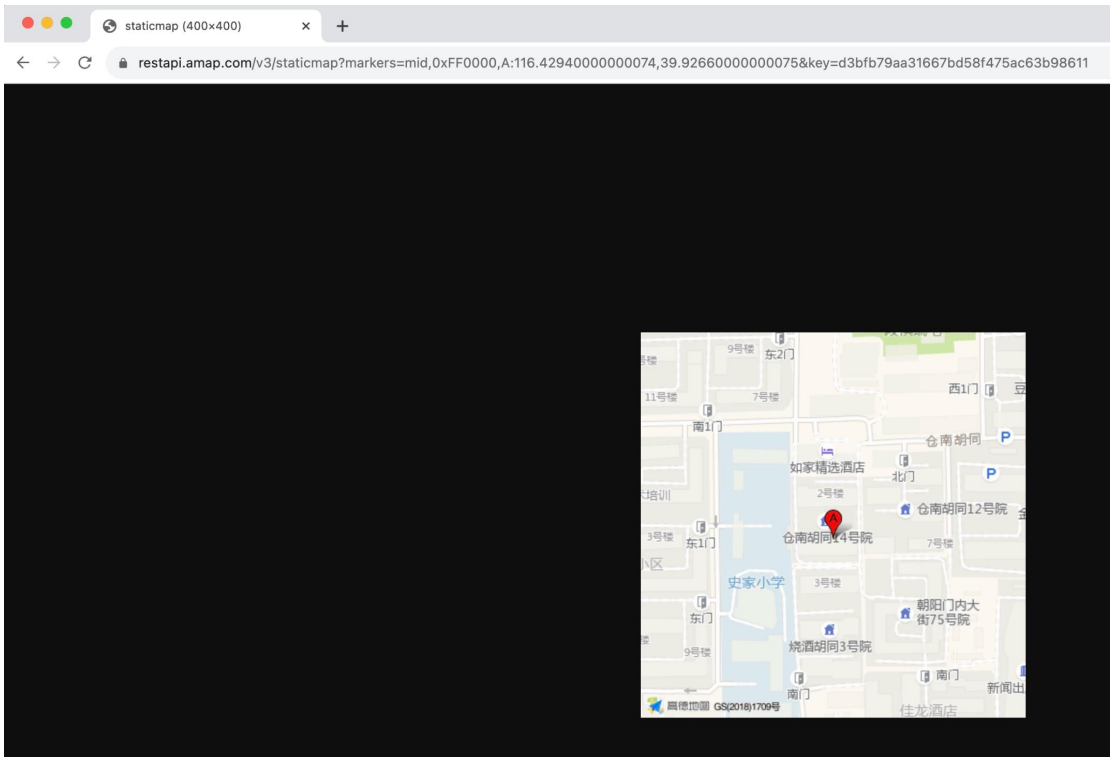


Figure 12 Location before user movement

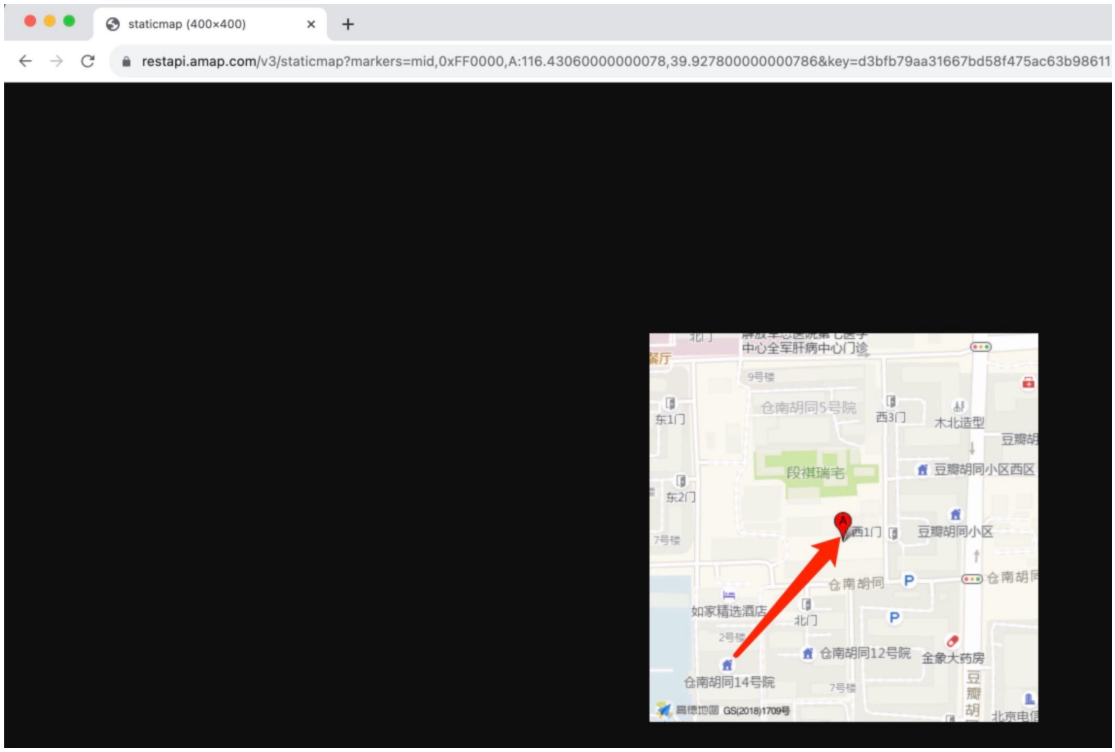
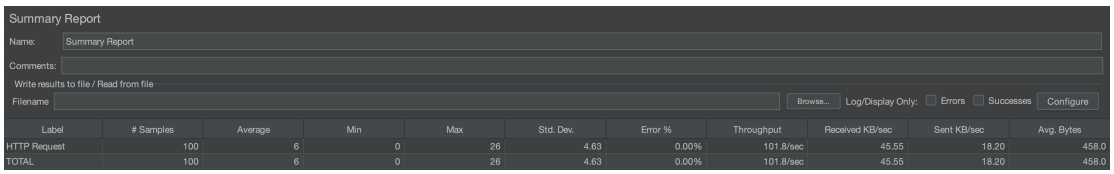


Figure 13 Location after user movement

Finally, we conducted stress testing on the program. The number of threads is 10, and the number of loops is 10, which means a total of 100 samples. In this case, the error rate of the program running is 0%, as shown in Figure 14.



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	100	6	0	26	4.63	0.00%	101.8/sec	45.55	18.20	458.0
TOTAL	100	6	0	26	4.63	0.00%	101.8/sec	45.55	18.20	458.0

Figure 14 Stress testing

8. Conclusion

Based on this project's previous description and demonstration, we can conclude that it has successfully implemented an efficient and reliable real-time location service system, achieving all the initial goals set. Meanwhile, by utilizing rich technology and a standardized hierarchical architecture, this project has demonstrated excellent data processing capabilities, passed stress testing, and ensured system stability and scalability.

By participating in this independent project, I have learned and mastered the concepts and applications related to Kafka and how to apply them in practice to a project. At the same time, I also learned how to effectively integrate multiple technologies (Kafka, Spring Boot, MyBatis-Plus, Redis, and MySQL) to build a complex system. In addition, I have gained a deeper understanding of how to design and implement a layered system architecture, especially when following the Alibaba coding standards. In short, this project improved my technical skills and enhanced my ability to solve complex problems, which will be very valuable in my future career.

9. Meeting minutes

(1) First project meeting

Date: 25/08; Duration: approximately 30 minutes; Meeting content: The main purpose of this meeting is to assign me a project supervisor. At the beginning of the meeting, I gave a detailed introduction to my technology stack and future career plan to Professor CHAN Shung Han Gary, and discussed my desired research direction. After understanding my situation, the professor designated his doctoral student Hu Siyan, a software engineer, as my project manager.

(2) Second project meeting

Date: 27/08; Duration: approximately 30 minutes; Meeting content: This meeting has determined the topic of my independent project. Firstly, I introduced my situation to my project supervisor, Siyan, and explained to her my desired research direction: Java backend development. Subsequently, we determined the topic of my independent

project as "Development of Real Time Location Service System Based on Kafka", and we also established the requirements and some details of the project, such as its ability to respond to real-time data requests from different users simultaneously. Finally, I completed the project proposal and submitted it to Siyan and the Professor for review.

(3) Third project meeting

Recently: 28/09; Duration: approximately 40 minutes; Meeting content: The topic of this meeting is the interim work report. At the beginning of the meeting, I reported to Shiyan on the progress of my project, that is, I have completed learning about Kafka and understood how to integrate Kafka with actual assignments. Meanwhile, I have also completed the overall project framework of this project. Then, Shiyan raised some questions about my project and asked me to provide an explanation of architecture. Then, I drew an architectural diagram by hand for Shiyan's reference. This meeting accelerated the progress of my project and gave me some new thoughts on my project.

(4) Fourth project meeting

Date: 16/11; Duration: approximately 20 minutes; Meeting content: I reported the implementation progress of this project to the professor at this meeting. Firstly, we discussed the architecture design and timeline of this project, as well as the technologies involved. Then, I introduced the requirements of this project to the professor and provided a requirement report to clarify the functions that this project aims to achieve. Meanwhile, after more than two months of development, this project has been largely completed, so I showed the professor some of the core code of this project and demonstrated its functionality, that is, how to obtain the real-time location of users on the map.