Hindawi
Mobile Information Systems
Volume 2022, Article ID 3928363, 12 pages
https://doi.org/10.1155/2022/3928363

Research Article

Design and Application of Basketball Microservice Platform

Xiangfeng Hou and Yihao Wang

School of Physical Education, Shanxi University, Taiyuan, Shanxi, China

Correspondence should be addressed to Yihao Wang; 171849040@masu.edu.cn

Received 8 January 2022; Revised 12 May 2022; Accepted 24 May 2022; Published 9 June 2022

Academic Editor: Ateeq Rehman

This study is aimed at the problems of low security, low response time, and poor functionality of the currently designed basketball microservice platform. A basketball microservice platform is designed. According to the definition and characteristics of microservice architecture, the proposed study expounds the related framework of Spring. Based on the feature vector of microservice architecture, the platform application client or other application requests are processed, the functional and nonfunctional requirements of the platform are analyzed, and the overall architecture of the basketball microservice platform is designed. According to the specific requirements of each module in the basketball microservice platform, the overall functional structure of the platform is designed based on the Spring Cloud framework and the Spring Boot framework. According to the monitored and managed entity objects of the microservice architecture infrastructure platform during operation, the relevant attributes of each object are described. Through the relational table designed by the microservice relational specification, various entity tables and relational information in the database are given to realize the design and application of the basketball microservice platform. The experimental results show that the proposed method has good functionality and can effectively improve the security and response time of the platform.

1. Introduction

With the rapid development of software technology and the continuous expansion of platform-scale applications, the software architecture has gradually evolved from a simple monolithic architecture model to a microservice architecture model. Correspondingly, framework products such as Netflix, Dubbo, and Spring Cloud are used to build microservices, or the community is gradually active and mature [1]. Microservice architecture is to vertically split a complete application from the database layer into multiple different services. Microservices are an operational structure that, in contrast to the typical monolithic architecture, divides an application into numerous service modules. Microservices' designs make it easier to expand and develop applications, allowing for more creativity and quicker response for new features. A software created with a microservice architecture is made up of distinct modules that operate each application process as a service. These services interact via lightweight APIs and a well-defined protocol. Services are designed to support core competencies, and each one serves a single purpose. Since each service runs separately, it may be modified, distributed, and expanded to match a variety of service application components. Each service runs on a separate thread, and each service can be deployed, maintained, and expanded independently. Services and services communicate with each other using API of unified style protocol [2]. Basketball is a typical representative team event. With the progress of team competition and training, all kinds of relevant data and information are increasing. How to effectively manage all kinds of information of the team, facilitate coaches to better manage the team, and improve competition results has become a common problem faced by the team [3]. The microservice platform simplifies the project construction process, reduces the configuration of a large number of configuration files, does not need to pay too much attention to the introduction of related dependent jar packages, and reduces the complexity of the entire code writing. It not only ensures the high stability, scalability, and maintainability of the platform but

also reduces the development cost and cycle. Therefore, it is of great significance to study the basketball microservice platform.

At present, the research on the basketball microservice platform has also made great progress. In [4], the authors designed a task-driven basketball teaching mode system based on ITbegin cloud platform [5]. According to the difficulty of basketball teaching and training content, it is divided into basic content and advanced content. On this basis, the curriculum setting of basketball teaching and training is proposed. It is published on the ITbegin cloud platform via the Internet. Students complete the training by downloading or watching the course online, feedback the training progress and problems encountered in the training in time, and adjust the teaching mode in time to ensure the quality of training. In the case of testers with similar physical fitness, the basketball teaching model based on the ITbegin cloud platform has a higher overall level, and the basketball teaching model design is more suitable for actual training projects. In [6], the authors designed a public service platform for university management based on microservice architecture. With the vigorous development of mobile computing, cloud computing, and complex business scale, there is an urgent need for a cloud service platform based on distributed architecture. The proposed study focuses on the problem of load balancing and its optimization. The platform has been successfully operated in many universities, significantly alleviating their business and operation problems. However, the above methods still have the problems of low platform security and response time and poor platform functionality.

To solve the above problems, a basketball microservice platform is designed. According to the definition and characteristics of microservice architecture, the proposed research paper expounds the related framework of Spring. Based on the feature vector of microservice architecture, the contributions are listed as follows:

(i) Process the platform application client or other application requests

(ii) Design the overall architecture of the basketball microservice platform by analyzing the functional and nonfunctional requirements of the platform

According to the specific requirements of each module in the basketball microservice platform and the microservice architecture, the overall functional structure of the platform is designed. Through the relational table designed by the microservice relational specification, various entity tables and relational information in the database are given to realize the design and application of the basketball microservice platform. The proposed method has good functionality and can effectively improve the security and response time of the platform.

2. Overview of Microservice Architecture

2.1. Microservice Architecture Definition. Microservice architecture is an architectural style adopted by software development for platform architecture design [7]. It is the most popular distributed platform architecture scheme at present. It is a solution with high cohesion and low coupling in software engineering. At the same time, it is also an excellent architecture style for interface development, which can well reduce the coupling between services. It is an architecture style, not the solid principle of software design.

Microservice architecture mainly decomposes complex business functions into several service centers, reduces the coupling of platform code, and provides more flexible function invocation support. Each service can be deployed and run independently without affecting each other. At the same time, it is convenient for the platform to carry out parallel development, which can improve the development efficiency. Therefore, the microservice architecture is also a business development oriented architecture. The microservice architecture improves the development efficiency and reduces the platform development cost and maintenance cost [8]. At present, major cloud computing manufacturers support the one click deployment and operation of microservice architecture, making the development, debugging, and maintenance of microservices easier.

2.2. Microservice Architecture Features. There are three characteristics of microservice architecture, which enable it to gain greater advantages in competition with traditional components [9]. The relevant classifications are as follows:

(1) Decomposing service processing: from a technical point of view, microservices can be regarded as components. The difference between these components and traditional components is that they are simple to operate and occupy less space. Traditional components isolate the business independent parts or extract the common parts in such a way that the application can be decoupled and reused after modularization. Microservice architecture directly decomposes the platform into multiple services, and there is a certain coupling relationship between services [10]. The functions in the application only need to change a single platform and then rebuild and deploy the corresponding services.

(2) Complexity reduction: the principle of microservice architecture is to decompose single applications into multiple independent services according to groups, in order to alleviate the complexity of execution data in the program and avoid useless data occupying the process and affecting the processing speed [11]. Assuming that the functions are fixed, after the application is decomposed into multiple services, the complex functions can be modularized through the microservice architecture, in order to solve the problem that the application coding method is difficult to realize due to the complexity of single application coding and simplify and facilitate the service development and maintenance process [12].

(3) Technology diversification: in the traditional development model, the application construction method is to use similar technologies, while the main

technology of the microservice architecture is a decentralized organizational structure, and the construction method has no focus. The service in the application makes corresponding judgments based on the scope of its own services and the current development of the industry, and after determining the technology type, counterpart services are provided to make the service scientific, professional, and platform-oriented, while making the service processing faster and more efficient [13].

Therefore, compared with the monolithic architecture, the microservice architecture is more flexible and pays more attention to business boundaries and business details. In actual use, the RPC protocol and HTTP protocol are used for invocation and scheduling between services, the transmission protocol is more standardized, and the coupling is small [14].

2.3. Spring-Related Framework. The Spring framework is currently the most widely used open-source framework in J2EE development. It is a lightweight and noninvasive container framework that uses Java's reflection mechanism to achieve inversion of control and aspect-oriented programming. Therefore, there are many excellent frameworks under Spring's ecological chain.

2.3.1. Spring Boot. The Spring framework rescues J2EE developers from the EJB research and development model and ecology. Spring's IOC (Inversion of Control) and AOP (Aspect-Oriented Programming), as well as its reasonable design and packaging of commonly used techniques for Java application development, make the Spring framework become the best practice for building efficient Java Web applications [15]. However, with the popularity of dynamic languages (such as Scala, Python, and Node.js), Java application heavy configuration, lengthy code, complex deployment process, and nonstandard third-party technology integration make Java development extremely cumbersome.

Spring Boot is the product of Spring framework's best practice of the concept of "Convention over Configuration." The so-called "Convention over Configuration" refers to the configuration of a large number of projects, which has many habitual configurations built in. There is no need for developers to configure [16]. With Spring Boot, a small amount of configuration can be used to easily build a Spring framework-based project that runs independently (running a jar file or an embedded web container). The position of Spring Boot in the Spring ecosystem is shown in Figure 1.

An overview of the role of Spring can be roughly summarized as follows: Spring Boot inherits the excellent genes of the Spring framework, thus simplifying the configuration process of Spring, allowing developers to quickly create a W song application. The application of Spring Boot to build microservices has the following advantages:

(1) Quickly build a W song application based on the Spring framework

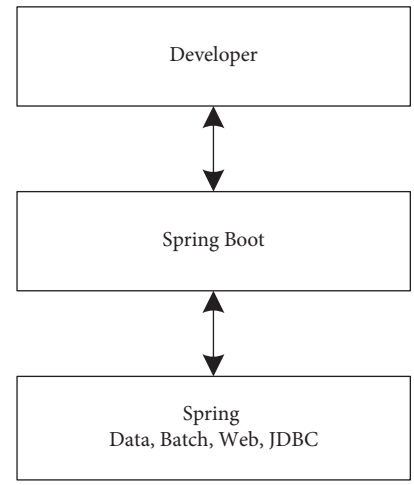

Figure 1: Location of Spring Boot in Spring ecology.

(2) With embedded WEB containers such as Tomcat and Jetty, the project can be deployed in the form of a WAR package without relying on it

(3) Using Spring-boot-start-actuator, you can obtain the performance parameters of the process running through REST, making service monitoring easier

(4) The configuration-free integration of mainstream development frameworks and tool chains can be well integrated with Docker container technology in service deployment, facilitating service deployment

2.3.2. Spring Cloud. Spring Cloud is an open-source microservice framework and a service governance framework. Spring Cloud relies on Spring Boot, which is spring boot. Spring Cloud provides many frameworks and tools for microservice development, such as service registration and discovery center Eureka, service gateway Zuul, and microservice configuration Spring Cloud Config [17].

Spring Cloud can be used simultaneously with other framework projects under the Spring ecosystem (such as Spring Framework, Spring Data, and Spring MVC), which can make the entire platform uniform, and subsequently, it can better play Spring Boot's "habit is better than configuration."It has the advantage of improving development efficiency. The technical system of Spring Cloud is shown in Figure 2.

Eureka, the service discovery and registration center, is one of the core components of Spring Cloud Netflix. Based on Netflix Eureka, it implements service registration and discovery functions and acts as a registry in the microservice architecture. At the same time, it optimizes the RESTful style of HTTP transmission by default [18].

2.4. Feature Vector Based on Microservice Architecture. When the platform application processes the request of the client or other applications, it will call the microservices and then cause the mutual calls between the microservices to form a call chain. One call chain corresponds to the use of

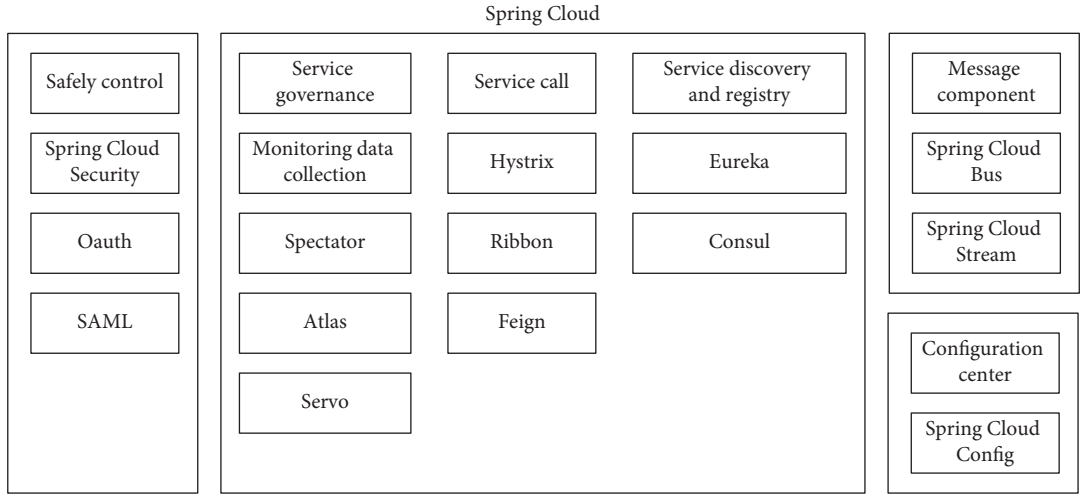

Figure 2: Technical system of Spring Cloud.

platform functions and permissions by a client in the platform. The platform functions and permission attributes in the aforementioned call chain are described by microservice application feature vectors.

Use L_{ID} to describe the call chain, and each call request in the call chain is represented by I_{ID}. The feature vector of the microservice application corresponding to one call is composed of a four-tuple:

$$Q = (I_{\mathrm{ID}}, W, E, R_L). \tag{1}$$

In formula (1), W represents the request service, E represents the calling function or method, and R_L represents the request parameter list. A call chain is composed of multiple application feature vectors, which are described by feature vector groups, namely,

$$AZ = (L_{\mathrm{ID}}, A_1, \ldots, A_i, \ldots, A_n, T_L). \tag{2}$$

In formula (2), n is the number of feature vectors, A_i is the i application feature vector, and T_L is the list of user roles that generated the feature vector set. It can be seen from formula (2) that a feature vector set consists of multiple feature vectors, for two feature vectors in a feature set AZ. If I_{ID_i} is the prefix of $I_{\mathrm{ID}_{i+1}}$, it means that there is a call path of service Z_i to method X_{i+1} in service Z_{i+1} in the platform, that is, the relationship between I_{ID_i} and $I_{\mathrm{ID}_{i+1}}$ is expressed as follows:

$$I_{\mathrm{ID}_i} = \mathrm{Parent}\left(I_{\mathrm{ID}_{i+1}}\right). \tag{3}$$

At the same time, there is a mapping relationship from $I_{I\ D}$ to $A_1, \ldots, A_i, \ldots, A_n$ and a mapping relationship from $L_{I\ D}$ to T_L in AZ. The definitions of these relationships in the feature vector set are given below. The relationship between the eigenvectors is defined as follows [19]:

$$AZC = \{L_{\mathrm{ID}}, AI, AM, AR\}. \tag{4}$$

In formula (4),

$$\begin{aligned} AI(L_{\mathrm{ID}}, Z_i) &= \left\{\langle Z_j, H_j\rangle, \ldots, \langle Z_k, H_k\rangle, \ldots, \langle Z_m, H_m\rangle\right\}, \\ &= \left\{A_{i1}, \ldots, A_{ik}, \ldots, A_{\mathrm{ip}}\right\}. \end{aligned} \tag{5}$$

Formula (5) represents the call list of A_i, that is, A_i calls $A_j, \ldots, A_k$. $AM(L_{\mathrm{ID}}) = \{A_m, \ldots, A_k, \ldots, A_n\}$ is the calling sequence list, marked as $\{Am_1, \ldots, Am_k, \ldots, Am_p\}$. $AR(L_{\mathrm{ID}}) = T_L = \{T_m, \ldots, T_k, \ldots, T_n\}$ is the list of calling roles, marked as $\{Ar_1, \ldots, Ar_k, \ldots, Ar_p\}$.

A feature vector relationship corresponds to the function and permission features in the application platform. The complete set of eigenvector relations of an application platform is represented as follows:

$$ACZ = \{AC_1, \ldots, AC_I, \ldots, AC_n\}. \tag{6}$$

3. Demand Analysis of Basketball Microservice Platform

3.1. Platform Function Requirements

3.1.1. Basketball Microservice Management and Control Module. The management and control module in the basketball microservice platform is mainly for the platform operation and maintenance personnel and carries various objects in the platform, such as cluster, server node, service, service instance, service warehouse deployment, management, monitoring, configuration management, and reliable operation ability. The operation and maintenance personnel are mainly responsible for the daily operation and maintenance control management of application services, database services, and conventional tool services in the microservice infrastructure platform, as well as the object management, service deployment, and configuration management in the corresponding platform. Object management includes providing control and management functions for clusters, warehouses, nodes, services, and instances in the microservice infrastructure platform. The configuration management module can modify and

control the fault-tolerant configuration information in the service instance and fault-tolerant module through the control module to achieve fine-grained operation. Finally, the control module is provided to the platform operation and maintenance personnel in the form of the Jar package. When using the control module, the operation and maintenance personnel only need to start the control module through the Java command line in the operating platform [20] and realize the dynamic control of the platform through the browser.

(1) Cluster creation function: it mainly provides the basic conditions for the cluster management function for the operation and maintenance personnel, in such a way that the operation and maintenance personnel can understand the operation of the platform from a macro perspective.

(2) Cluster modification function: it mainly modifies the basic information of the cluster in such a way that the operation and maintenance personnel can control the cluster. The modification content includes the description information of the cluster and the operation status of the cluster.

(3) Cluster warehouse: it mainly manages the warehouse in the project file. The operation and maintenance personnel shall specify the warehouse used in cluster deployment before deployment and determine the deployable service object through the project file in the warehouse.

(4) Service node search function: it is mainly used to more freely obtain the information of available nodes in the LAN and reduce the workload of operation and maintenance personnel.

(5) Service instance deployment function: on the basis that the cluster provides a deployment carrier for service instances, the platform shall build deployment instances according to the deployment scheme provided by the operation and maintenance personnel, deploy service instances, and display the cluster deployment status and progress to users.

(6) Service configuration modification function: it is the main means for operation and maintenance personnel to control the platform, and it is also a means for the control module to carry out fine-grained control over service instances during service operation. The module shall provide technical support for the fault-tolerant policy configuration modification module. After setting a series of cluster operations, the fault-tolerant policy configuration modification module realizes the configuration switching of the platform through fine-grained configuration modification of service instances.

3.1.2. Basketball Microservice Monitoring Module. The basketball microservice monitoring module is mainly to provide the operation and maintenance personnel with the service instances during the operation of the microservice infrastructure platform and the status collection function of the service operation node. In addition, it also includes the subsequent summary of the collected data and the storage management of the collected data. The main participants in the platform are the operation and maintenance personnel in the platform. The operation and maintenance personnel shall collect the status of service instances and service nodes running in the microservice infrastructure platform in the form of HTTP interface through the monitoring module. Different evaluation methods are provided for different upper application services. The collected data will describe the service instance, node, and cluster status with the corresponding evaluation formula. Finally, it is written into the database through the data storage function to facilitate the display function of the monitoring module for chart aggregation display. The above monitoring modules will eventually be provided to the operation and maintenance personnel in the form of service components during delivery. The operation and maintenance personnel can start the modules by command line startup according to the functional requirements of the platform and complete the collection, analysis, and storage of the platform through the cooperation between modules.

(1) Node status collection: it enables the operation and maintenance personnel to better observe the status of the service node during the operation of the microservice infrastructure platform, mainly including the static information and dynamic information of the server node.

(2) Service instance status collection: it is mainly performed through the monitoring summary module deployed by the server node.

(3) Monitoring data summary module: it is mainly to collect data of nodes and service instances.

(4) Monitoring status viewing function: it is the part related to the control module and the monitoring module. The operation and maintenance personnel locate the specific status viewing object through the control module and display and aggregate the collected data in the platform by calling the monitoring data display module.

3.1.3. Basketball Microservice Fault Tolerance Module. The basketball microservice fault-tolerant module mainly configures the fault-tolerant strategies of the cluster through the operation and maintenance personnel and provides a variety of fault-tolerant strategies to supplement the single fault-tolerant means in the current microservice infrastructure platform. On the original basis, routing fault-tolerant means, service degradation means, and flow control means are added to ensure the reliable operation ability of the microservice infrastructure platform. The main functions of the fault-tolerant module of basketball microservice platform include adding routing fault-tolerant, service degradation, flow control integration module, and cluster fault-tolerant policy configuration. A variety of fault-tolerant strategies shall be provided to the microservice infrastructure platform in the form of annotations and added to

the infrastructure platform. The control module shall combine the cluster fault-tolerant configuration function in the fault-tolerant configuration management module to realize the fault-tolerant management in the highly available microservice platform. Finally, all functions will automatically operate in the form of an interceptor filter when the service is started, provide support for the microservice infrastructure platform, and reduce the fault-tolerant control cost of operation and maintenance personnel.

(1) Add fault-tolerance strategy: it mainly integrates the components in the highly available microservice platform into the microservice infrastructure platform to provide protection means for the platform.

(2) Routing fault-tolerance strategy: it is mainly a means to deal with the request exception, which makes it impossible to obtain the correct result. The exception at the platform routing level can be solved through the integration of the retry mechanism and subsequent processing.

(3) Service degradation strategy: it is proposed to provide three forced degradation, fault-tolerant degradation, and elegant degradation schemes to degrade individual abnormal services, hence not to affect the business process of the whole platform and ensure the availability of the platform in high concurrency scenarios.

(4) Flow control strategy: it mainly deals with the uncontrollable scenario of high concurrency of the microservice infrastructure platform. In the highly available microservice platform, the platform's current limit plan is controlled through the gateway and the platform's internal services. Through the current limit strategy, the platform can switch the current limit level with one click, in such a way that the operation and maintenance personnel can better control the service access and ensure the reliability of the platform.

(5) Fault-tolerance policy configuration function: it is the core function of the microservice fault-tolerance module. During the operation of the platform, a series of operations of the control module are to make advanced environmental preparations for fault-tolerance policy configuration and provide technical support for service fault-tolerance policy configuration management. Users can realize platform level fault-tolerant control and safe and reliable operation of the platform through one click fault-tolerant configuration and fault-tolerant strategy selection.

3.2. Platform Nonfunctional Requirements. Nonfunctional requirements refer to software platform features other than specific behavioral requirements for the platform. The development of the software platform itself finally needs to be attributed to the application. The nonfunctional requirements need to consider that the software should be available and easy to use on the server in addition to completing its basic function points in practical application. Therefore, the nonfunctional requirements of the platform mainly include the following aspects:

(1) Security: because the platform is developed in the style of microservice architecture and the communication between all modules is called through the interface, it needs high security. All requests with permission requirements shall be verified without unauthorized call. Users or IP with too high access frequency shall be recorded, and there shall be complete log records.

(2) High availability: the platform shall have high availability. The failure of one node shall not affect the operation of the whole platform. Redundancy shall be provided for each service, and horizontal clustering shall be carried out.

(3) Manageability: ensure that the deployment, monitoring, and optimization of the platform can be tracked and visualized. The main service management configuration, service monitoring and tracking, and service data statistics are managed with a complete user interface, and the operation of the whole platform can be understood at any time.

(4) Extensibility: the microservice platform framework will include all functional modules, allowing developers to customize the content of the framework and develop extended functional modules. The definition of service interface maintains good compatibility with old and new versions. Service deployment and version upgrade should support a smooth transition. The deployment of services should support cluster expansion.

4. Basketball Microservice Platform Design

4.1. Overall Architecture Design of Basketball Microservice Platform. According to the requirements and boundaries of the basketball sports microservice platform, the specific modules included in the basketball sports microservice platform can be derived: basketball sports microservice monitoring module, basketball sports microservice fault-tolerance module, basketball sports microservice management and control module, and so on. It can be concluded that the overall architecture of the basketball microservice platform is as shown in Figure 3.

Figure 3 describes the overall architecture of the basketball microservice platform in the case of fine granularity. The relationship between each module in the basketball microservice platform and the microservice infrastructure platform is as follows:

(1) Microservice architecture is a favorable supplement to the microservice infrastructure platform: in view of the problems existing in the microservice infrastructure platform, a microservice architecture is added outside the microservice infrastructure platform to provide management and control, fault tolerance, monitoring, and other support for the

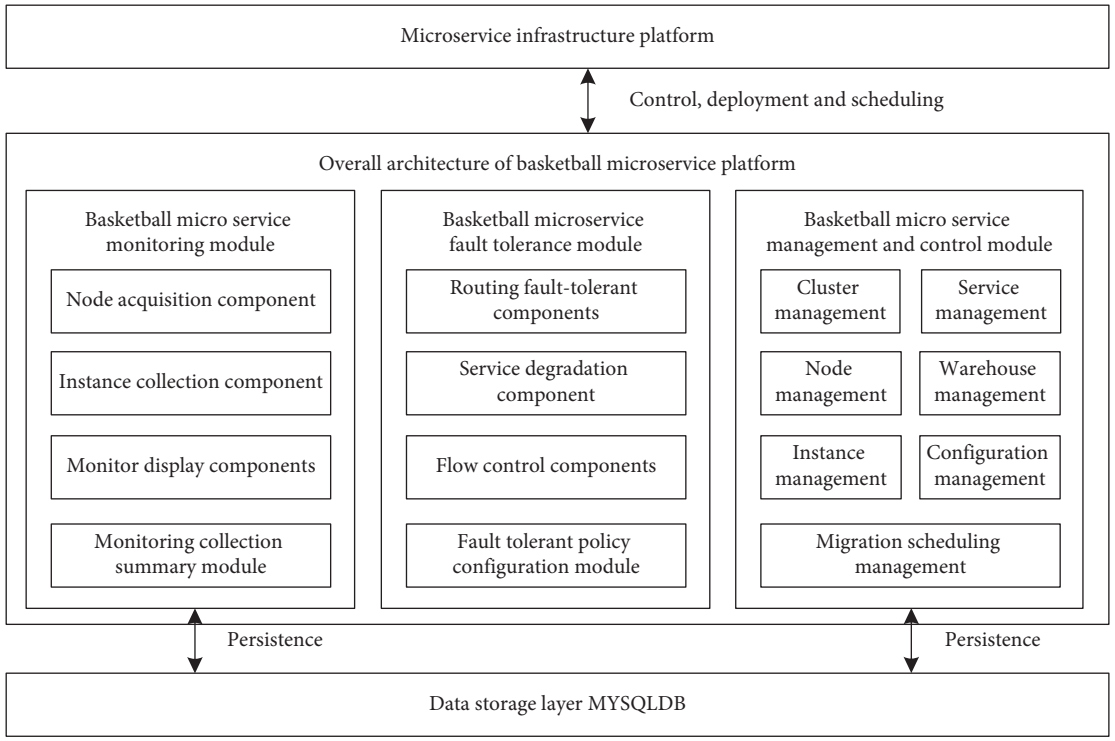

Figure 3: Overall architecture of basketball microservice platform.

operation of the microservice infrastructure platform.

(2) The basketball microservice monitoring module is responsible for improving the monitoring in the microservice infrastructure platform: the basketball microservice monitoring module provides the collection function of operation data for the microservice infrastructure platform. Based on the collected data, the objects in the infrastructure platform are evaluated, stored, and displayed persistently, which provides a one-stop platform monitoring capability for the services in the microservice infrastructure platform.

(3) The basketball microservice fault-tolerant module is responsible for improving the fault-tolerant means of the microservice infrastructure platform: the basketball microservice fault-tolerant module can inject different components into the microservice infrastructure platform, such as routing fault-tolerant component, service degradation component, and flow control component. Fault-tolerant strategies such as routing fault tolerance, service degradation, and flow control are added to the microservice infrastructure platform through components. By integrating fault-tolerant policies and controlling them with fault-tolerant policy configuration modules, the dynamic switching and configuration of fault-tolerant policies in microservice infrastructure platforms are realized.

(4) The basketball microservice management and control module is used to solve the inconvenience of management and maintenance in the microservice infrastructure platform: the basketball microservice management and control module has the functions of deployment, management, scheduling, and configuration management of the microservice infrastructure platform and stores various object data in MySQL to ensure the controllable operation of the platform [21]. In addition, the control module also provides various interfaces to cooperate with the monitoring module and fault-tolerant module to realize the microservice architecture management of the platform.

(5) Basketball microservice monitoring module, basketball microservice fault-tolerance module, and basketball microservice control module are all platform outputs: basketball microservice monitoring module and basketball microservice control module are mainly for platform operation and maintenance personnel to help them improve their work efficiency and reduce the management difficulty of microservice infrastructure platform and the actual operation of fault-tolerant policy management and control. Basketball microservice fault-tolerant module is mainly service-oriented developers, which is provided to developers in the form of annotations to reduce the development cost of developers and lay the foundation for the configuration of fault-tolerant strategy in the fault-tolerant module.

4.2. Functional Structure Design of Basketball Microservice Platform. By analyzing the overall requirements and

functional requirements of the basketball microservice platform, we studied the monitoring, fault tolerance, and control functions required by the platform in detail. According to the specific requirements of each module in the basketball microservice platform, the overall functional structure of the platform is designed and divided from the perspective of the functional structure of the platform. The functional structure of the basketball microservice platform is shown in Figure 4.

As can be seen from Figure 4, the basketball microservice platform mainly includes three modules: basketball microservice monitoring module, basketball microservice fault-tolerance module, and basketball microservice management and control module. According to the results of demand analysis, combined with the overall architecture of the basketball microservice platform, the functions are subdivided again.

The basketball microservice monitoring module mainly includes five core modules which are presented in Table 1.

The basketball microservice fault-tolerance module mainly includes two main functions:

(1) Fault-tolerant policy integration module: based on the Spring Cloud framework, design and implement the fault-tolerant policy integration module of basketball microservice platform. The fault-tolerant integration module is embedded into the service gateway and service consumers of the microservice infrastructure platform to realize the expansion of the microservice fault-tolerant strategy. The specific expansion strategies include routing fault-tolerant strategy, service degradation strategy, and flow control strategy.

(2) Fault-tolerant policy configuration module: the fault-tolerant configuration module is designed and implemented based on the Spring Boot framework, which is responsible for modifying the fault-tolerant policy configuration of the services injected by the microservice fault-tolerant policy integration module. The control module provides a fault-tolerant configuration scheme. After receiving the configuration content of the control module, the fault-tolerant policy configuration module analyzes and selects the gateway or service consumer in the microservice infrastructure platform, fills in the configuration content, and finally calls the configuration center in the microservice infrastructure platform to write and refresh the configuration of the corresponding instance.

The basketball microservice management and control module includes 9 main functions as shown in Table 2.

When the service instance in the cluster runs abnormally, the monitoring module will alert the microservice control module to call the service instance scheduling module. Through the instance scheduling function, the service instance will be reselected to migrate and deploy, in order to achieve the automatic repair of the instance.

4.3. Platform Database Design. After giving the functional structure of the platform, according to the monitored and managed entity objects of the microservice architecture infrastructure platform in the operation process, the relevant attributes of each object are described, and the entity relationship is determined according to the relationship table designed by the microservice relationship specification as shown in Table 3. The structure of each table in the database is given, including various entity tables and relationship information. The description of the table structure in the platform is given in table form, including table name, type of each field, primary and foreign keys, and notes.

5. Basketball Sports Microservice Platform Application

5.1. Setting Up the Platform Test Environment. In order to test the effectiveness of the designed basketball sports microservice platform, through platform design requirements, consider the deployment mode of the microservice architecture infrastructure platform and deploy multiple service instances in multiservice nodes. Combined with the mutual cooperation between services, complete the business functions and processing procedures of the platform and deploy the basketball microservice platform. Set the network environment as a laboratory LAN with a bandwidth of 100 Mbps. The Manager component is deployed in a node to control the deployment of the platform. The MySQL database runs on the server in the form of Docker containerization. Other services include the microservice infrastructure platform and the platform components in the basketball microservice platform. The management and control module data are stored uniformly by MySQL. Select the number of 1000 MB platform clients. The test indicators are platform functionality, platform security, and platform response time. The method in [4, 6] and the proposed method are, respectively, compared to verify the performance of the proposed method design platform.

5.2. Platform Functional Test Results. The functional test is to verify each function of the platform according to the characteristics and operation description of the platform and judge whether the platform can meet the design requirements. Functional testing is also called black box testing or data-driven testing. It only needs to consider the platform functions to be tested and does not need to consider the internal structure and code implementation of the platform. According to the functional modules determined in the requirement analysis, functional tests are carried out,

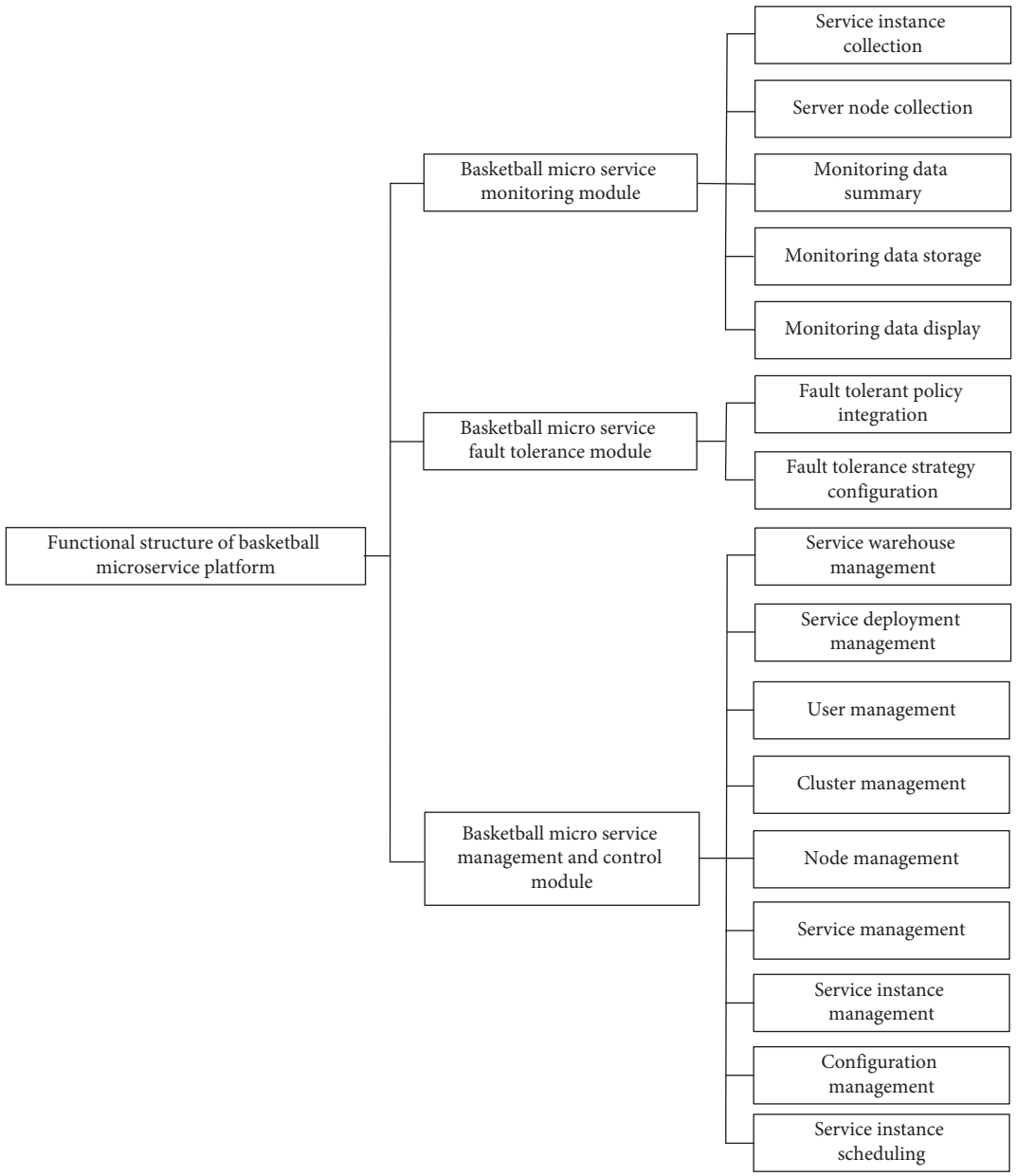

Figure 4: Functional structure diagram of basketball microservice platform.

Table 1: Basketball microservice monitoring core modules.

1	Service instance collection module	Based on the Spring Boot framework design, the collection module of the service instance running status in the microservice infrastructure platform is finally integrated into the platform service instance
2	Server node collection module	Based on the Spring Boot framework design, it realizes the collection service of the operating state of the service node, the basic environment for the operation of the microservice infrastructure platform
3	Monitoring summary module	It is integrated in the service node collection module to collect, analyze, summarize, and evaluate the status information of service instances and service nodes
4	Monitoring data storage module	A module based on the JPA framework to design and implement the persistence of the collected data and realize the persistent storage of the monitoring data by cooperating with the monitoring summary module
5	Monitoring display module	It provides the cluster operation and maintenance status viewing function for the operation and maintenance personnel. The user calls the monitoring display module through the control module to view the status of each object in the cluster

Table 2: Basketball microservice management and control module.

1	Service warehouse management	Operation and maintenance personnel configure the warehouse address of service files required for cluster deployment through service warehouse management and manage the service version in the warehouse
2	Service deployment management	Provide a one click cluster service deployment function for operation and maintenance personnel and automatically deploy the platform after obtaining the deployment services and service scale required by users
3	User management	Manage users in the control module, i.e., operation and maintenance personnel, mainly including registration and login functions to assist subsequent operations
4	Cluster management	Provides users with cluster management capabilities. Users can create, delete, modify, and query clusters through the control module interface
5	Node management	It provides users with the management capability of server nodes. Users can join the cluster, remove, modify, query, and explore the server nodes through the control module interface
6	Service management	It provides the operation and maintenance personnel with the management ability of services in the cluster. The operation and maintenance personnel control the service table through the control module interface and add services through the cluster warehouse to provide deployable services for the cluster
7	Service instance management	It provides operation and maintenance personnel with the ability to manage service deployment instances in the cluster. Users can deploy, modify, close, delete, and view service instances through the control module interface
8	Configuration management	Provides users with the ability to configure and manage service instances. Users can view, modify, and delete specific service instances running in the cluster through the control module interface
9	Service instance scheduling	It provides users with service instance scheduling capability

Table 3: Microservice relationship specification.

1	User table	Mainly used to store user information, including user password, permission, logical deletion, and other fields
2	Cluster table	It mainly stores the cluster information, defines the cluster through the cluster name and description information, displays the operation status of the cluster through the status information, determines the location of the configuration center in the cluster through the configuration path, and provides the logical deletion field to provide the logical deletion function for the platform
3	User cluster table	An intermediate table generated by the many-to-many relationship between users and clusters. It is used to store the relationship between users and clusters
4	Node table	It is mainly used to store the basic information of the node, which is the hosting object of the service instance running in the platform, including the node status and the cluster to which the node belongs, and realize the logical deletion of the node through the node status
5	Node resource table	It mainly relies on the existence of the node table. Because of the one-to-one relationship with the node table, the primary key of the node table is used as its own primary key and the immutable basic information in the node is stored, including the CPU number of cores, memory size, hard disk usage, and name and version of the operating platform
6	Node dynamic resource table	Mainly used to store the platform's collection information on the running status of the server node, collect the running status of the node at a preset timing, and evaluate the running quality of the node after the resource
7	Service table	It mainly stores the service information that can be run in the platform, filters abnormal services, and serves subsequent instances
8	Cluster service table	Used to associate services with clusters. A service can also be used by multiple clusters; thus, it exists as an intermediate table
9	Service instance table	Because services run in the form of service instances in the cluster, the instance table mainly stores the basic information held by the running services
10	Instance resource table	It has the same meaning as the node resource table and has a one-to-one relationship with the service instance. It is used to store the environment information of service operation
11	Service dynamic resource table	Used to store the running information of the service instance, including the dynamic running status of the service instance. In addition, its specific quality evaluation value is also stored
12	Warehouse table	It is mainly used to store the information of the warehouse where the project files of the service are located, which is convenient for the automatic deployment and use of the platform. It contains information such as the deployment type, the username and password of the server node where the warehouse is located, the address, and the path

Table 4: Functional test results of different method design platforms.

Functional module	The proposed method	The method of [4]	The method of [6]
Basketball microservice monitoring module	Normal	Normal	Abnormal
Basketball microservice fault-tolerance module	Normal	Abnormal	Normal
Basketball microservice management and control module	Normal	Normal	Abnormal

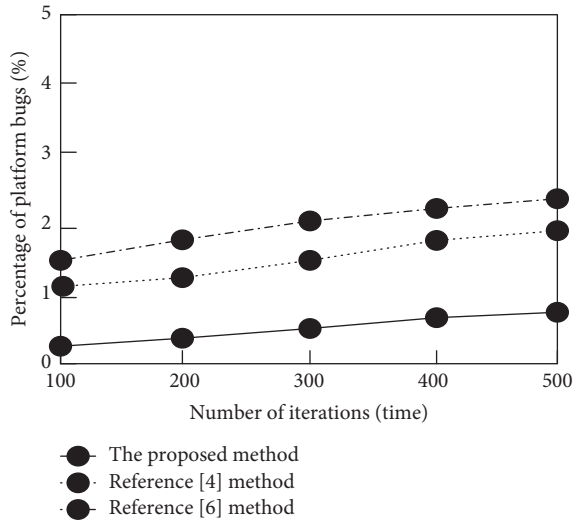

Figure 5: Security test results of different method design platforms.

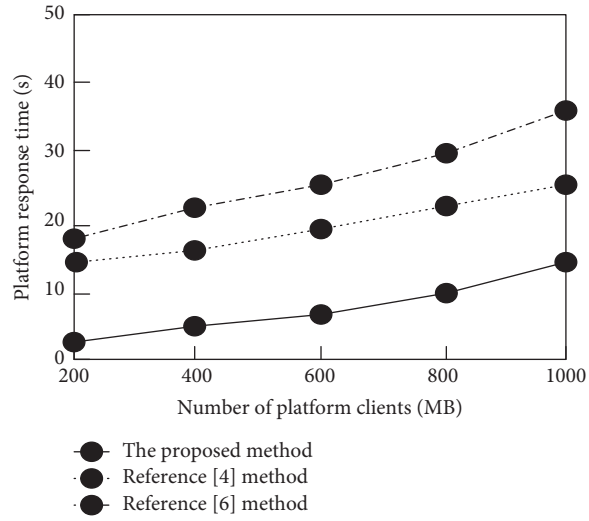

Figure 6: Response time test results of different method design platforms.

respectively. First, functional test cases are designed, and then the corresponding test cases are executed to verify whether the functions realized by each module can meet the requirements. The methods of [4, 6] and the proposed methods are compared to verify the functional test results of the platform designed by different methods, as shown in Table 4.

According to Table 4, the basketball microservice monitoring module and basketball microservice control module of the platform designed by the method [4] are in normal state, but the basketball microservice fault-tolerant module of the platform designed by the method in [4] is abnormal. The basketball microservice fault-tolerant module of the platform designed by the method in [6] is in normal state, but the basketball microservice monitoring module and basketball microservice control module of the platform designed by the method in [6] are abnormal. All functional modules of the proposed design platform are in normal state. It can be seen that the proposed method has good functionality.

5.3. Platform Security Test Results. On this basis, the security of the platform designed by the proposed method is further tested and the proportion of platform bugs is taken as the platform security evaluation index. The lower the proportion of the number of platform bugs, the less the vulnerabilities of the method design platform and the higher the security of the platform. We have compared the methods of [4, 6] and the proposed methods, respectively, to verify the security test results of different design platforms, as shown in Figure 5.

As can be seen from Figure 5, with the increase of iteration times, the number and proportion of bugs on different design platforms increase. Among them, when the number of iterations reaches 500, the number of bugs of the design platform based on the method in [4] accounts for 2% and the number of bugs of the design platform based on the method in [6] accounts for 2.38%, while the number of bugs of the design platform based on the proposed method accounts for only 0.8%. It can be seen that the number of bugs of the proposed method design platform is relatively low, indicating that the proposed method design platform has fewer vulnerabilities and higher platform security.

5.4. Platform Response Time Test Results. The response time of the platform designed by the proposed method is further verified. The method in [4, 6] and the proposed method, respectively, are compared to verify the response time of the platform designed by different methods. The test results are shown in Figure 6.

As can be seen from Figure 6, with the increase of the number of platform clients, the response time of platforms designed by different methods increases. When the number of platform clients reaches 1000 mb, the response time of the platform designed by the method in [4] is 25.2 s and the response time of the platform designed by the method in [6] is 36.8 s, while the response time of the platform designed by the proposed method is only 15 s. Therefore, the response time of the proposed design platform is short.

6. Conclusion

The basketball microservice platform designed in the proposed study gives full play to the advantages of microservice architecture. This basketball microservice platform has good functionality and can effectively improve the security and response time of the platform. The response time of the proposed design platform is reduced. However, in the basketball microservice platform, when the scale of service instances deployed by service nodes is too large, the monitoring summary module still occupies too many resources without considering.

In future, we aim to work with more platform clients/MB. Second, the monitoring and acquisition architecture needs to be further improved and the storage can be optimized or stored using a time series database and NoSQL database.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] C. K. Rudrabhatla, "Security Design Patterns in Distributed Microservice Architecture," Article ID 03395, 2020, https://arxiv.org/abs/2008.03395.

[2] A. Avritzer, V. Ferme, A. Janes et al., "Scalability assessment of microservice architecture deployment configurations: a domain-based approach leveraging operational profiles and load tests," *Journal of Systems and Software*, vol. 165, no. 9, Article ID 110564, 2020.

[3] H. Wu and L. Wang, "Analysis of lower limb high-risk injury factors of patellar tendon enthesis of basketball players based on deep learning and big data," *The Journal of Supercomputing*, vol. 78, no. 3, pp. 4467–4486, 2021.

[4] N. N. Zhang, "Research on task driven basketball teaching mode based on ITbegin cloud platform," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 327, pp. 92–104, 2020.

[5] N. N. Zhang, "Research on task driven basketball teaching mode based on ITbegin cloud platform," in *Proceedings of the International Conference on Multimedia Technology And Enhanced Learning*, pp. 92–104, Leicester, UK, April 2020.

[6] L. Huang, C. Zhang, and Z. Zeng, "Design of a public services platform for university management based on microservice architecture," *Microsystem Technologies*, vol. 27, no. 4, pp. 1693–1698, 2021.

[7] K. Yin and Q. Du, "On representing resilience requirements of microservice architecture systems," *International Journal of Software Engineering and Knowledge Engineering*, vol. 31, no. 6, pp. 863–888, 2021.

[8] E. Al-Masri, "Enhancing the microservices architecture for the internet of things," in *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*, pp. 5119–5125, IEEE, Seattle, WA, USA, 10-13 December, 2018.

[9] T. Cerny, M. J. Donahoo, and M. Trnka, "Contextual understanding of microservice architecture," *ACM SIGAPP - Applied Computing Review*, vol. 17, no. 4, pp. 29–45, 2018.

[10] D. Taibi and K. Systä, "A decomposition and metric-based evaluation framework for microservices," in *Communications in Computer and Information Science, International Conference on Cloud Computing And Services Science*, pp. 133–149, Springer, Cham, 2020.

[11] K. Kravari and N. Bassiliades, "StoRM: a social agent-based trust model for the internet of things adopting microservice architecture," *Simulation Modelling Practice and Theory*, vol. 94, pp. 286–302, 2019.

[12] X. Zhou, X. Peng, T. Xie et al., "Fault analysis and debugging of microservice systems: industrial survey, benchmark system, and empirical study," *IEEE Transactions on Software Engineering*, vol. 47, no. 2, pp. 243–260, 2021.

[13] F. Zhiyong, X. Yanwei, X. Xiao, and C. Shizhan, "Review on the development of microservice architecture," *Journal of Computer Research and Development*, vol. 57, no. 5, p. 1103, 2020.

[14] N. Okumura, K. Ogata, and Y. Shinoda, "Formal analysis of RFC 8120 authentication protocol for HTTP under different assumptions," *Journal of Information Security and Applications*, vol. 53, Article ID 102529, 2020.

[15] A. Ginanjar and M. Hendayun, "Spring framework reliability investigation against database bridging layer using Java platform," *Procedia Computer Science*, vol. 161, pp. 1036–1045, 2019.

[16] R. Bucea-Manea-Oni and R. Bucea-Manea-Oni, "How to design a web survey using spring boot with mysql: a Romanian network case study," vol. 17, no. 2, Article ID 26458, 2019.

[17] H. Guo, Z. Xu, and G. Chen, "Design and implementation of microservice in immigration system based on spring cloud," *China Computer & Communication*, vol. 14, pp. 63–65, 2019.

[18] M. Zhang, B. Marculescu, and A. Arcuri, "Resource and dependency based test case generation for RESTful Web services," *Empirical Software Engineering*, vol. 26, no. 4, p. 76, 2021.

[19] Y. Feng, "Gradient feature extraction of landscape spatial pattern based on vega," *Computer Simulation*, vol. 37, no. 11, pp. 366–370, 2020.

[20] L. Wang, J. Li, and B. Li, "Tracking runtime concurrent dependences in Java threads using thread control profiling," *Journal of Systems and Software*, vol. 148, pp. 116–131, 2019.

[21] A. B. Manduri, A. Ghani, S. Shamshirband, and A. T. Chronopoulos, "Optimizing infrastructure as a service provider revenue through customer satisfaction and efficient resource provisioning in cloud computing," *IET Communications*, vol. 13, no. 18, pp. 2913–2922, 2019.