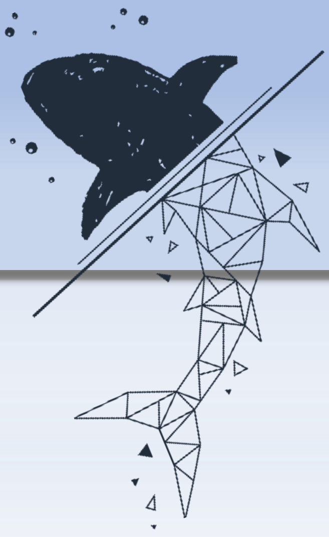
System Analysis and Design
Spring 2021

JIYU



Meet what you will meet

Team Member：

1851055	Mingjie Wang
1851049	Zhongyue Chen
1851231	Liyou Wang
1951724	Kaixin Chen
1954098	Ningyu Xiang

Instructor: Yan Liu

1. Introduction

1.1 Purpose

The purpose of this System Design Model Document (SDM document) is to present detail design model of the JIYU(Meet What You Will Meet). The document will cover the platform-independent architecture and related subsystems with specific interfaces. And some mechanisms and samples of system prototyping will be given.

1.2 Project Scope

The JIYU (Meet What You Will Meet) is a web-based club management system, whose goal is to simplify the club management process and provide an online club social platform. The users can access the system by PC or mobile device through a Website.

Specially, potential scenarios of this system mainly include searching clubs, browsing activities information and club information, joining clubs, dropping out of a club, and so on. Both club members and non-club members can browse the information and join in the activities, make comments through the system after they register and log in. In addition, club members can use the system to organize activities, making the organization easier. As for club union supervisor, they have the authority to manage the online forum and highlight the activities, which simplifies their work.

Besides, this online system also has the listed features:

- running in a networked environment.
- handling the user authentication for security.
- owning a centralized database and etc.

1.3 Glossary of Terms

TERMS	DEFINITION
JIYU	The name of our system, which means that you will meet the friends after using our system to join the world of different activities.
Club	A club is usually composed of a group of people in the university who have common hobbies and goals. It is also usually an important part of university life.
TOWS	A diagram to analyse the main internal strengths, weaknesses and external opportunities and threats closely related to the research object with the idea of systematic analysis, and draw a series of corresponding conclusions.
Online Forum	A website part, which refers to discussions among different people around the same topic
Authority	the different operations that each actor can perform when using this system.
Column	A collection of the same type of topic posts in the forum.
Publicize Activities	the club union supervisor can post the activities reported by the associations to the homepage, so that more people can pay attention to the activities.
Report	Some complaints made by users about the inappropriate columns or comments (e.g. insulting people or spreading terrifying information) on the forum. The supervisor will check whether it's a proper report.
Club Statics	Data on the development of the club, usually including changes in the number of clubs over time, participation in activities, etc.
Resume	In our system, it refers to an account's basic information when registered, including email, telephone number,student ID and some other minor information.
Activity	The online or offline activity held by the university club, whose main body is users or club members.
Modify Permission	The system modifies the user's authority and changes the scope of the user's function.

1.4 Progress on System Design

In the previous report, we have already designed an analysis model of JIYU club management system. We provide a concrete user interface, architecture, and class diagram of the project.

Based on the previous job, this time, we determined the microservice architecture of the project. We will provide specific technical architecture and logical architecture model, and design specific interface for our project, detailed description of part of the third party API, subsystem interface, analysis mechanism, use case realization and system prototype design. We focus on the specific design of the project, prepare to explain the main technology needed to implement the project, and determine how the project will be implemented.

The main improvement of our system are listed as follows:

1. Reconfiguration of the logical architecture through analysis mechanisms, etc.
2. In addition to the logical architecture, technical and physical architectures have been added. The required technology stack in each layer and the way to deploy it at the physical level are provided.
3. Add the specific interfaces. We add specific interface based on our new subsystem. More details such as parameter type and return value type are also taken into consideration. We also provide detailed instructions for the interfaces of the third party and some subsystems.
4. We use API gateway, data persistence and a series of design mechanisms to increase the project's security and reliability, high fault tolerance, and flexibility.
5. We analyse the mechanism in the model and select the appropriate design mechanism.
6. Apply to design patterns in the system design process for analysis
7. We personally practiced the construction of part of the project content and present the prototyping.
8. For the use case in conjunction with the structure of the system to analyze its implementation process

1.5 Description of Implementation Platforms and Frameworks

In the practical development, we will use microservice architecture. In microservice architecture, each microservice in the system can be independently deployed, and each microservice is loosely coupled. Each microservice focuses on just one task and does it well. This allows for low coupling, greater flexibility, targeted problem solving, easier independent development, and high availability and stability. So we need a series of proven techniques and frameworks to build the system. For each sub-system and component in the whole system, the overview of practical implementation is shown as follows:

Web application: Build the front-end frame through NPM, Vue-cli, node.js, element-UI, Vuetify and other tools. Using AJAX to update information on the page without refreshing the page.

API Gateway: Using API Gateway as a unified external interface to offer microservice by applying Spring Cloud Alibaba. Spring Cloud Alibaba contains the necessary components for the development of distributed application microservices, so that we can easily use these components to develop distributed application services through the Spring Cloud programming model.

Security: Authorization authentication is carried out through Sa-Token, and security testing is carried out based on this lightweight authorization authentication framework, which meets the development requirements of the JIYU community management system

Data Storage: We use Oracle database to store data and Redis to meet the security and consistency of data in high concurrency scenarios.

Server Framework: Using Spring boot as a powerful web-developing framework with high expansibility and excellent performance and use java for back-end development.

1.6 Architectural Styles

The system as a whole adopts the microservice architecture style, which realizes the decentralization of the design and improves the fault tolerance, scalability and maintainability of the system.

The idea of microservice architecture is not to develop a huge monolithic application, but to decompose the application into small, interconnected microservices. A microservice completes a particular function, and this system divides the entire system into five broad microservice subsystem parts.

Because the services provided by JIYU are relatively complete and independent, they can be packaged into different microservice clusters for other systems, so this system can be designed as a microservice architecture system with high cohesion, low coupling and independent deployment of different services, while microservice development is resilient, fast iterative and supports rapid updates, which can meet the needs of the current system.

In the design process for a single application in a microservice architecture, our analysis is shown below.

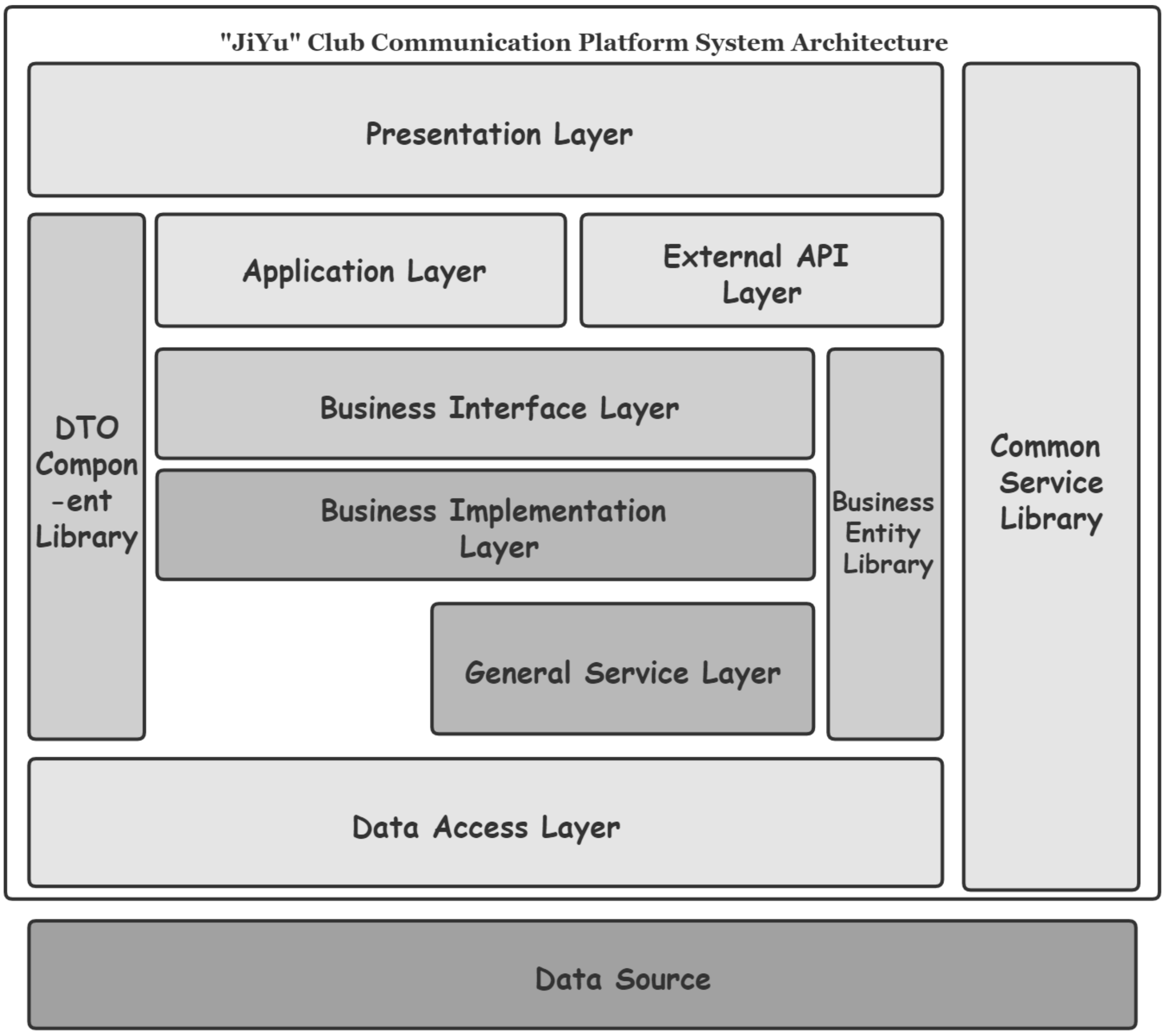
Garlan and Shaw divided software architecture styles into five categories, data flow style, call/return style, independent component style, virtual machine style and warehouse style. Among them:

- Data flow style includes batch processing sequence architecture style and pipeline/filter architecture style;
- The call/return style includes main program/subprogram architecture style, data abstraction and object-oriented architecture style, and hierarchical structure architecture style;
- Independent component style includes process communication architecture style and event-driven architecture style;
- Virtual machine style includes interpreter architecture style and rule-based system;
- Warehouse style includes database architecture style and blackboard architecture style.

Our system mainly adopts the call/return style to design the overall system. In terms of overall architecture, the architecture is designed from small to large based on data abstraction and object-oriented architecture, hierarchical architecture aspects

1. Data abstraction and object-oriented architecture.
   Data abstraction and object-oriented architecture. The components of data abstraction and object-oriented architecture style are objects, and objects are instances of abstract data types. In abstract data types, the representation of data and their corresponding operations are encapsulated. The behaviour of the object is reflected in its acceptance and request actions. Connectors are the way of interaction between objects. Objects interact through the calls of functions and procedures. Objects are encapsulated, and the change of one object will not affect other objects. Objects have state and operations, and are also responsible for maintaining state. This structural style includes features such as encapsulation, interaction, polymorphism, integration, and reuse.
   To achieve a low-coupling, highly reusable and maintainable system that is easy to upgrade and protects internal integrity, this system uses IOC (Inversion of Control) to build containers to achieve decoupling between objects and AOP (Tangent Oriented Programming) to extract public behaviors encapsulated as common services to reduce system redundancy and coupling between modules and improve system operability and maintainability.
2. Hierarchical structure architecture.
   The hierarchical system is organized into a hierarchical structure. The component implements a virtual machine in some layers. Connectors are defined by protocols that determine how the layers interact. Topological constraints include constraints on the interaction between adjacent layers. Each layer provides services for the upper layer, and when using the services of the next layer, you can only see the layer adjacent to it. The big problem is broken down into a number of gradual small problems, which are solved step by step, hiding a lot of complexity. When one layer is modified, it will affect at most two layers, and usually only the upper layer can be affected. The upper level must know the

identity of the lower level and cannot adjust the order between levels.
In the previous design for this system, we used a hierarchical architecture as shown in the figure, which divides the general three-tier architecture in more detail. However, in order to achieve low coupling of the system, etc. We have improved the architectural design, while retaining the hierarchical architecture, which will be described in detail later.



1.7 Design Patterns

In the process of our system development, the following design patterns are applied:

Singleton Pattern

In **Online Fourm System**, we apply singleton pattern to our online forum, for it has one and only one instance in the whole system.

In our project, we uses Java enumeration types to implement the singleton pattern. OnlineForum enumeration type in only one enumeration value INSTANCE, in the call to the static method. Therefore, OnlineForum().getInstance will always return this enumeration value to achieve the effect of a globally unique single instance.

API Instance Description

API	ONLINEFORUM().GETINSTANCE
Description	get the singleton instance of the Online Forum
Parameter	None
Return	OnlineForum.INSTANCE enumeration value

Dirty Flag Pattern

In **Activity System**, When activities are reported to the Club Union Supervisor, the Activity set that stores all the activities to be reviewed is altered. When the Club Union Supervisor need to calculate the current fee of all the activities, the examination needs to calculate from scratch ,which is very time-consuming.

So in order to save time, we can use the dirty-flag pattern. When the alteration of the activity set occurs, the dirty flag of the ActivitySet will be set to 1. In this case,when the Club Union Supervisor need to calculator the current fee of all the activities, he only need to check the dirty flag to determine whether the total fee need to re-calculate, which saves the time of calculating the activity fee.

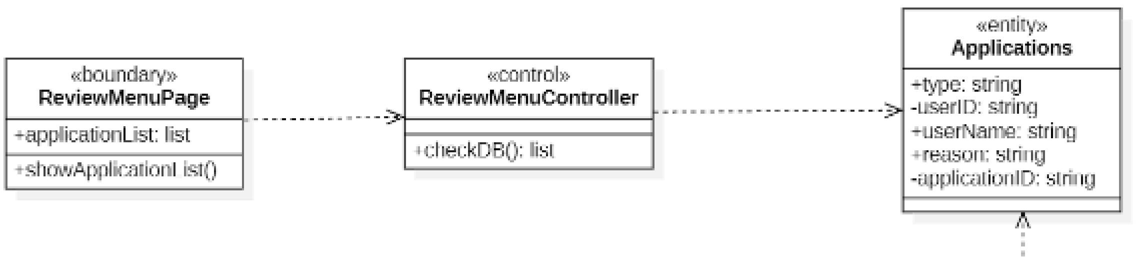
API Instance Description

API	ACTIVITYSET.BEVISITED(DOUBLE ALTER)
Description	Activityset is updated with insertion of new activities, deletion of old activities or alteration of the activity fee. In this case,set dirty flag to True
Parameter	the alteration of the sum of the activities
Return	None

API	ACTIVITYSET.GETVISITED()
Description	check the dirty flag value to see if ActivitySet has altered.
Parameter	None
Return	the boolean value of dirty flag

API	ACTIVITYSET.CALCULATEFEE()
Description	check and calculator the current cost of all the activities
Parameter	None
Return	the current cost of all the activities

MVC Pattern



In the **MembershipChange System**, after the club member or user submit a personnel change application, the corresponding actor needs to review the application.

In the MVC design pattern, we divide the application reviewing into Model-View-Controller three parts, making each module independent of the corresponding function.

ReviewMenuController is used to control the list of corresponding **Applications** from the database, displayed in the **ReviewMenuPage** boundary class, making the view and model separate. **ReveiwMenuPage** is used to present the list of Applications to the corresponding actors.

Interpreter Pattern

When users view recommendation activities, they often don't know how many activities are organized recently.
Therefore in this scenario, we set the sequence of the current activities according to the user's preference, as long as we provide the next activity for the user when they view it.
In iterator design mode, a method is provided to access elements in an aggregate object sequentially without exposing the internal representation of the object.
So we abstract: **SuggestedActivity** Interface, **Activities** Interface and **AvitivitiesName** Class. **ActivitiesName** class stores the recommended browsing order.

1.8 Critical Design Decisions

- In order to achieve a strong ease of use, our system has made a page designed in accordance with intuition and surveyed the user experience. Based on the results of the survey, the page is updated and specific functions are adjusted.
- Through the three architecture styles mentioned above (main program subprogram architecture style, data abstraction and object-oriented architecture style, hierarchical architecture style), we encapsulate the data we need, so that quick and convenient calls can be made, and achieve high interaction efficiency.
- Because this system is applied to schools, it is necessary to ensure the security of data. With the small range of application and data, it is more in line with the characteristics of the system to use a relatively lightweight framework for development, so our system adopts the Sa-Token authorization framework to protect users' information.
- At the same time, to ensure security, we also encrypt the privacy part of the database to prevent the leakage of database information.
- In the case that the forum deletion management mechanism requires manual review, we use Google’s Captcha framework and enter and identify the verification code for each publication to prevent malicious comments from robots.

2. Architecture Refinement

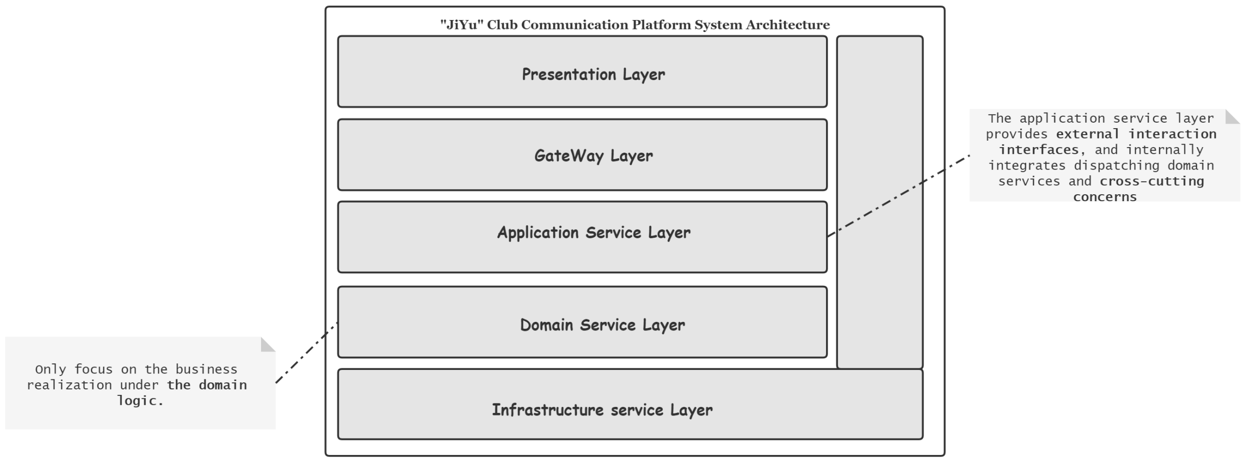
2.1 Platform-dependent Architecture

In the previous logical architecture design of the community exchange platform, we focused on the integrity of the system's internal functions and the security of data transmission, and combined the design of the actual technical and physical models, we chose to use **a microservice architecture system based on domain-driven design** to complete the improvement and upgrade of the system's logical architecture.

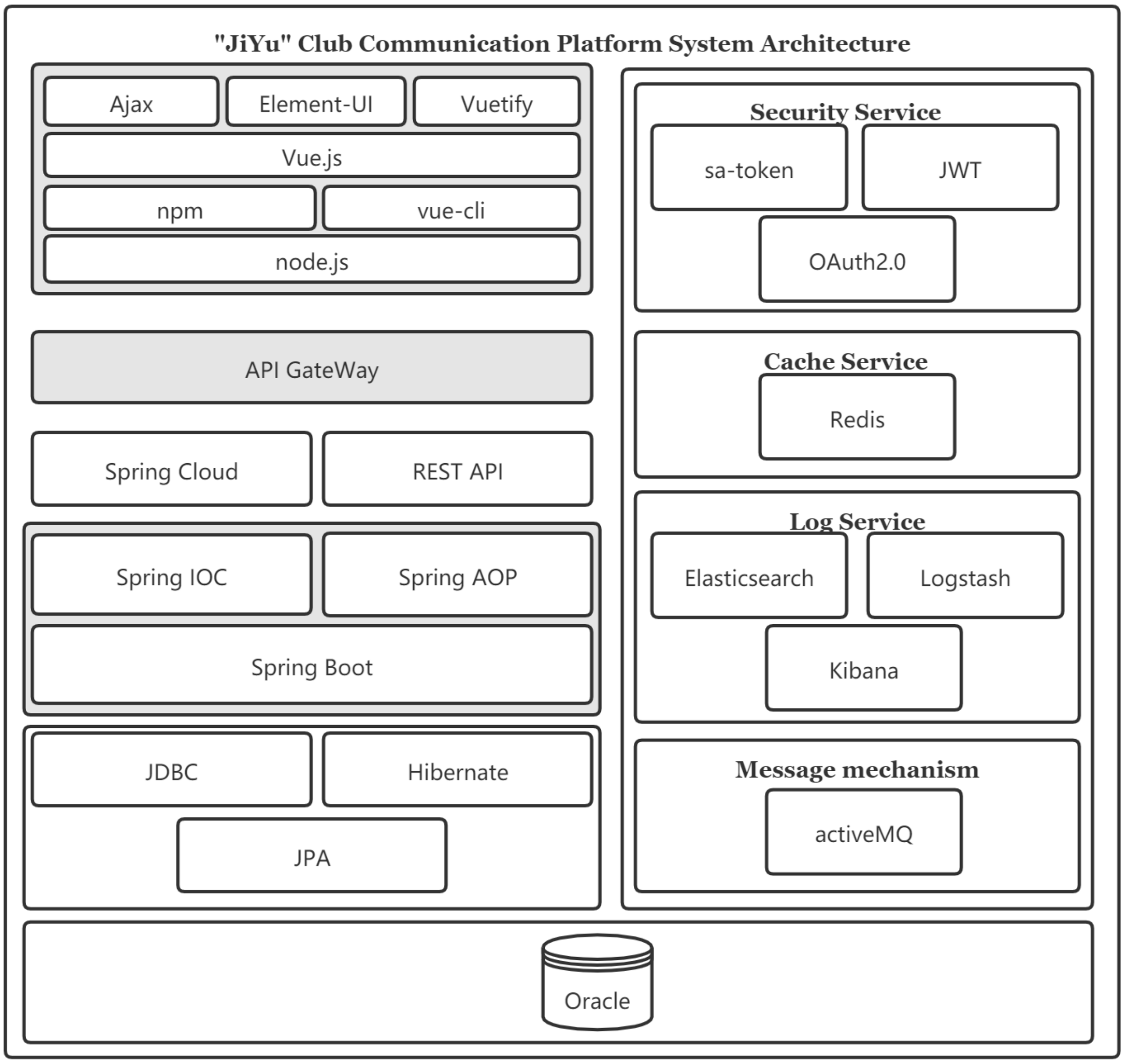
The reasons for choosing the transition from layered architecture to microservice architecture in this project are.

1. the services provided by the Keizai community exchange platform are relatively complete and independent, and can be packaged into different microservice clusters for other systems to use.
2. Microservice development is flexible, fast iterative and supports rapid updates.
3. Different services are deployed independently, with high cohesion in services and loose coupling between services.

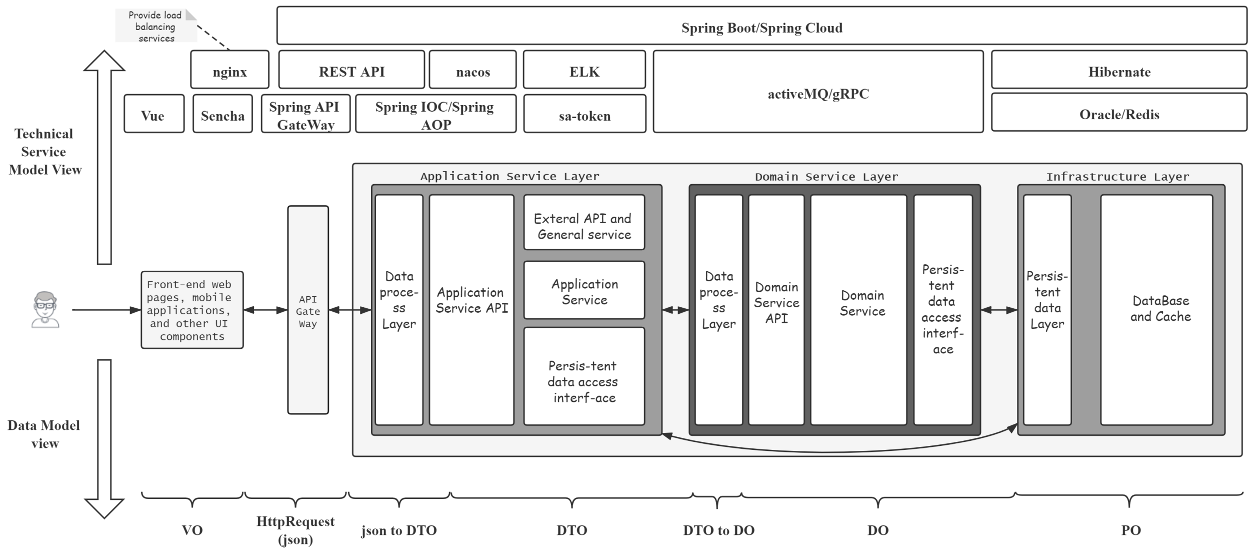
Based on the above characteristics of microservice architecture, and after weighing the pros and cons of microservice architecture and traditional layered architecture, in order to design a system architecture with **complete internal functionality**, **high security** and **light weight**, we choose to generalize and refine the previous architecture between layers appropriately to achieve the following logical architecture diagram.


"JiYu" Club Communication Platform System Architecture
Presentation Layer
GateWay Layer
Application Service Layer
Domain Service Layer
Infrastructure service Layer
The application service layer provides external interaction interfaces, and internally integrates dispatching domain services and cross-cutting concerns
Only focus on the business realization under the domain logic.

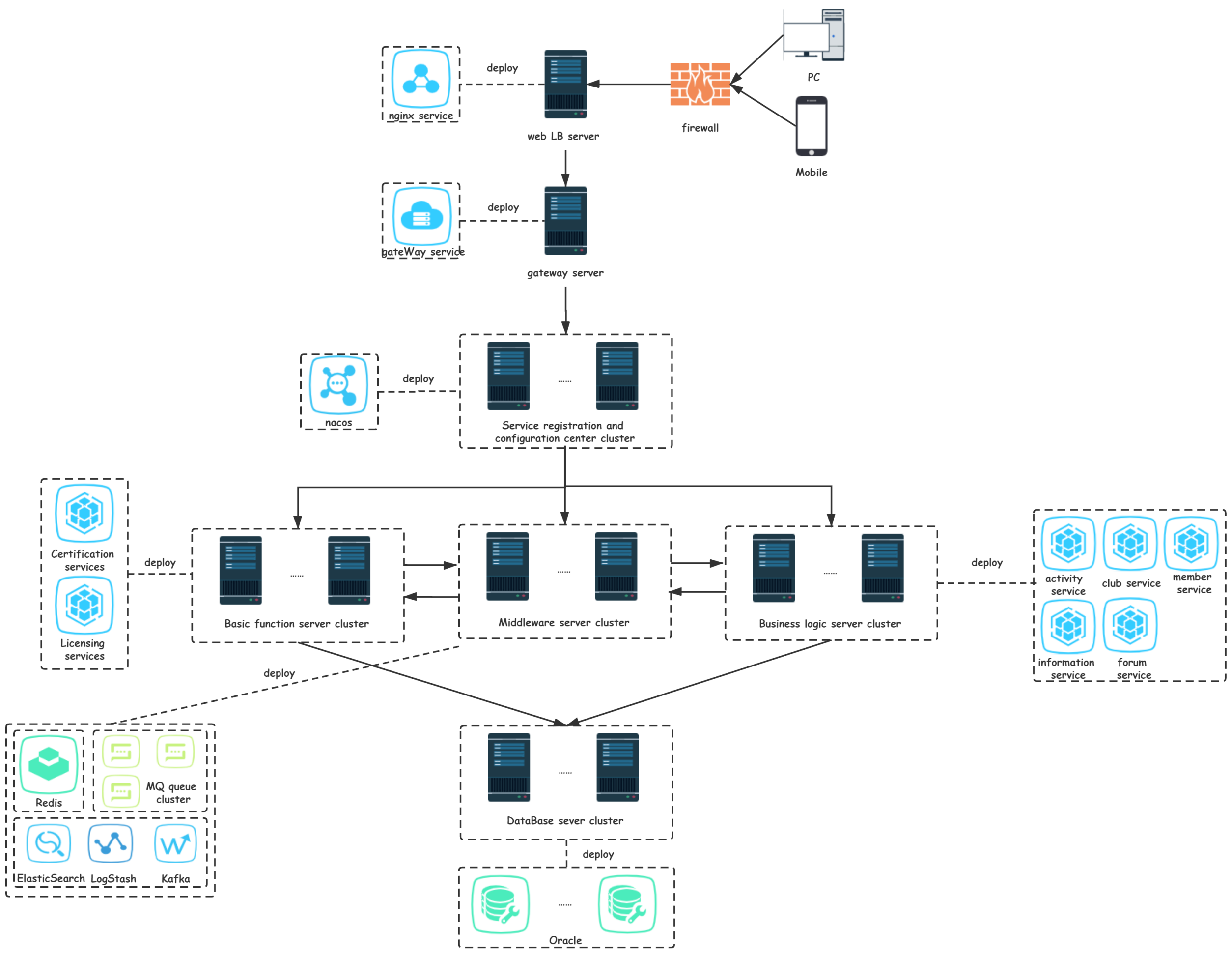
Based on the above improved logical architecture, in order to adapt the microservices architecture to the project size, we still use separate front-end and back-end development, with the front-end using Vue, Sencha and other libraries for development, and the back-end using SpringBoot and SpringCloud for development in general. The front-end and back-end communicate through REST-style API calls and json data files, and the layers communicate and convert through different data models (e.g., VO, DTO, DO, etc.). The system provides authorization and authentication through sa-token, complete log collection and data analysis visualization through ELK, and in order to effectively address asynchronous messages and traffic clipping, activeMQ message queues are used for message delivery and storage. The backend exchanges data with Oracle database or Redis cache through Hibernate's persistence mechanism. Based on the above process, the technology stack diagram of the design system is：



Combine the technical architecture with the microservice architecture to judge the integrity and correctness of the design process and design the following data and technical service model diagram：



In order to realize the mapping between the logical and physical architecture of the system, the physical implementation of the system is shown by means of a deployment diagram. Due to the characteristics of the microservice architecture, a cluster of service registries and configuration centers and three clusters of servers with different functions are used to realize the communication between microservices. The deployment diagram is as follows:



2.2 Subsystems and Interfaces

According to domain-driven design, in order to properly design the corresponding domain services, we chose to divide the business boundary involved in the system into four partial domains.

- Activity Domain
- Community domain
- User domain
- Online Forum domain

Based on the above domain design, in order to accommodate the moderate granularity of services in the system design process, we choose to further divide the microservices into the following systems based on functionality.

- Activity service system
- Club service system
- User service system
- Club member service system
- Online Forum service system

Club Service System

ORDER NUM	REST API	INTERFACE INTRODUCTION
001	POST api/club/clubID	The interface accepts the information of establishing a new club and returns whether the operation is successful .
002	PUT api/club/clubID	The interface accepts the information of modifying a club and returns whether the operation is successful.
003	DELETE api/club/clubID	The interface accepts the information of a club and returns whether the club is successfully deleted.
004	PUT api/club/clubID/clubleader	The interface accepts the information of a club leader and returns whether the leader is accepted.
005	PUT api/club/clubID	The interface accepts the information of a club and returns whether the club establishing application is accepted.

ORDER NUM	REST API	INTERFACE INTRODUCTION
006	GET api/club/clubID	The interface returns the information of a specific club.

Activity Service System

ORDER NUM	REST API	INTERFACE
007	GET api/activities/	The function of this interface is to accept the conditions of filtering activities and return the qualified activity sequence.
008	GET api/activities/activityID/	The interface accepts the activity ID returned by the front end, and returns the details of the ID activity by calling data.
009	POST api/applications/	The interface accepts the registration data of the user applying to participate in the activity at the front end, processes the logic and subscribes the message through the business layer, and the return value is whether the user operation is successful or not.
010	POST api/activities/	The interface accepts the activity application and approval status selected by the front end, and returns whether the operation is successful after logical processing and message passing.
011	POST api/activity-section/	The function of the interface is to receive the activity information data returned by the front-end, call the business logic to carry out new activities, and return whether the execution status is successful or not.
012	PUT api/applications/applicationID/	The interface accepts the activity application and approval status selected by the front end, and returns whether the operation is successful after logical processing and message passing.

Membership Service System

ORDER NUM	REST API	INTERFACE INTRODUCTION
013	POST api/memebership/leaderships	The function of the interface is to receive the application information data returned by the front-end, call the business logic to create new application, and return whether the execution status is successful or not.
014	POST api/memebership/memberships	The function of the interface is to receive the application information data returned by the front-end, call the business logic to create new club-join application, and return whether the execution status is successful or not.
015	PUT api/memebership/leaderships	The function of the interface is to access an application selected by the front-end and returns whether the operation is successful after logical processing and message passing.

Online Forum Service System

ORDER NUM	REST API	INTERFACE INTRODUCTION
016	GET /api/forum/columns	The interface returns all the columns in the forum.
017	GET /api/forum/columns/id	The interface returns the specific column.
018	GET /api/forums/columns/id/comments	The interface returns all the comments of the specific column.
019	GET /api/forums/columns/id/comments/id	The interface returns the specific comment of a certain column.
020	POST /api/forum/columns	The interface accepts the content of a new column, and returns whether the operation is successful.

ORDER NUM	REST API	INTERFACE INTRODUCTION
021	POST /api/forums/columns/id	The interface accepts the content of a new comment under a specific column, and returns whether the operation is successful.
022	PUT /api/forums/columns/id	The interface tries to update the content of a specific column, and returns whether the operation is successful.
023	DELETE /api/forums/id	The interface tries to delete a specific column, and returns whether the operation is successful.
024	DELETE /api/forums/id/columns/id	The interface tries to delete a specific comment of a certain column, and returns whether the operation is successful.
025	GET /api/report	The interface returns all the reports to be handled
026	GET /api/reports/id	The interface returns a specific report.
027	GET /api/reports/id/userid	The interface returns the ID of the user who raised the specific report.
028	POST /api/reports	The interface accepts the content of a new report, and returns whether the operation is successful.
029	DELETE /api/forums/reports/id	The interface tries to delete a specific report from the database, and returns whether the operation is successful.

User Service System

ORDER NUM	REST API	INTERFACE INTRODUCTION
030	POST /api/users/	The user realizes the account registration through this interface.
031	GET /api/users/?userID=id & userPassword = password/	The user realizes the function of logging in to the system through this interface.

ORDER NUM	REST API	INTERFACE INTRODUCTION
032	PUT /api/users/?userID=id/	The user realizes the request to change personal information through this interface.

2.3 Interface Specification

Since different roles participate in the community communication platform, in order to ensure data privacy security and transmission efficiency in the operation process, the project should include security services responsible for login authentication and permission authentication. Considering that the front and back ends of the project development process are separated, there are security requirements such as authentication, so **sa-token**, a lightweight and powerful security framework, is selected to provide system security services.

Login authentication

Since many operating interfaces in the community communication platform are restricted to access after login, login authentication is required before the interfaces that require login permission operations are executed. The sa-token provides interfaces for querying the login status and token information parameters. This interface can be used Cut into other interfaces to perform corresponding operations.

API	INTRODUCTION
StpUtil.**setLoginId**(Object loginId);	*Mark the account id of the current session login.*
StpUtil.**logout**();	*Log out of the current session.*
StpUtil.**isLogin**();	*Get whether the current session is logged in, return login state.*
StpUtil.**checkLogin**();	*Check whether the current session has been logged in, if not logged in, an exception will be thrown:* ***NotLoginException***
StpUtil.**getLoginId**();	*Get the login id of the current session, if not logged in, an exception will be thrown:* ***NotLoginException***
StpUtil.**getLoginIdByToken**(String tokenValue);	*Get the login id corresponding to the specified token, if not logged in, return* ***null***
StpUtil.**getTokenValue**();	*Get the token value of the current session.*

API	INTRODUCTION
StpUtil.**getTokenInfo**();	*Get the token information parameters of the current session.*

Take the user login interface as an example to show the process of calling the API:

During the login process, call the login interface to pass parameters as follows:

USERNAME	USERPASSWORD
testID	tongjitongjiSAD123.

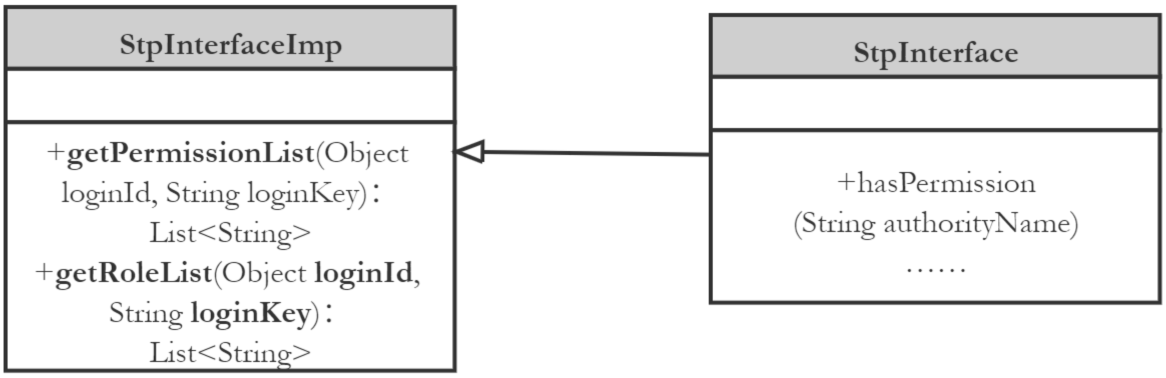
Access the login interface through **URL: /api/Login?username=testID&password=tongjitongjiSAD123/** (The actual URL may be slightly different, the purpose here is to give an example).

In the login business logic, the correctness of the account password will be judged. If the account is legal, the interface StpUtil.**setLoginId**(Object loginId) will be called to mark that the user is logged in.

Authority authentication

Because certain operations in the community communication platform need to be performed by specific roles, permission authentication needs to be introduced in some interfaces. sa-token uses to determine whether the account has a permission code to achieve permission authentication, but different projects have different permission requirements, so it is necessary to extend the sa-token framework to achieve the operation of obtaining a set of permission codes.

Therefore, the interface class StpInterfaceImp responsible for authorization authentication in the system is:


StpInterfaceImp
+getPermissionList(Object loginId, String loginKey): List<String>
+getRoleList(Object loginId, String loginKey): List<String>
StpInterface
+hasPermission (String authorityName)
……

The API is defined in the Permission Authentication Interface class as follows.

API	INTRODUCTION
StpUtil.**hasPermission**(String authorityName);	*Whether the current account has specified permissions.*
StpUtil.**checkPermissionAnd**(String athName1, String athName 2, ...);	*Whether the current account have specified permissions. [Specify multiple, all must be verified]*
StpUtil.**checkPermissionOr**(String athName1, String athName 2, ...);	*Whether the current account have specified permissions. [Specify multiple, as long as one of them is verified]*

Take the precise assignment of permissions to page elements as an example to control whether the elements on the page are displayed:

This function is difficult to achieve by only relying on the back-end authorization authentication, so it needs to be completed by the front-end. After logging in, the user returns all the permission codes owned by the account to the front-end at one time, and the front-end saves the collection of permission codes in *localStorage* or other global state management objects, and makes js judgments in the page elements that require permission control, such as:

```
<button v-if="authorityArr.indexOf('user:delete') > -1">delete
button</button>
```

You can hide the page elements of users who do not have relevant permissions through simple permission code transmission.

2.4 Example

The application of community communication platform is not limited to a single web project or application. For the convenience of users, the functions of different systems of community communication platform can be provided to external systems in the form of interface, such as student information system, student trading platform, etc., so as to provide convenience for users to receive and process information.

Taking the external interface in the activity organization system as an example, it shows how the external system calls the interface to complete the corresponding business logic in the corresponding scenario.The specific API design of the activity service system is shown below:

ORDER NUM
001
REST API
POST api/activities/
Interface
OrganizeActivities(ActivityDTO activityInfo): bool
Arguement
ActivityInfo : activityDTO
Return Value
bool
Interface Introduction
The function of the interface is to receive the activity information data returned by the front end, call the business logic to carry out new activities, and return whether the execution status is successful or not.

ORDER NUM
002
REST API
GET api/activities/
Interface
BrowseActivity(ConditionDTO activityCondition): list<BriefActivityDTO>
Arguement
ConditionDTO : activityCondition
Return Value
list<BriefActivityDTO>
Interface Introduction
The function of this interface is to accept the conditions of filtering activities and return the qualified activity sequence.

ORDER NUM
003
REST API
GET api/activities/activityID/
Interface

ORDER NUM
ViewActivity(str activityID): ActivityDTO
Arguement
activityID : str
Return Value
ActivityDTO
Interface Introduction
The interface accepts the activity ID returned by the front end, and returns the details of the ID activity by calling data.

ORDER NUM
004
REST API
POST api/applications/
Interface
ParticipateInActivity(str activityID , str userID , ApplicationDTO activityApplication): bool
Arguement
activityID : str , userID : str , activityApplication : ApplicationDTO
Return Value
bool
Interface Introduction
The interface accepts the registration data of the user applying to participate in the activity at the front end, processes the logic and subscribes the message through the business layer, and the return value is whether the user operation is successful or not.

ORDER NUM
005
REST API
PUT api/applications/applicationID/
Interface
ReviewActivity(str activityID , int state , str applicationID): bool
Arguement
activityID : str , state : int , applicationID : str

ORDER NUM
Return Value
bool
Interface Introduction
The interface accepts the activity application and approval status selected by the front end, and returns whether the operation is successful after logical processing and message passing.

ORDER NUM
006
REST API
POST api/activity-section/
Interface
PublicizeActivity(str activityID , int degree , str activityText): bool
Arguement
activityID : str , degree : int , activityText : str
Return Value
bool
Interface Introduction
The interface accepts the activity ID and the published copy, performs logical processing and message passing, and returns whether the operation is successful or not.

The external system calls to select activities API according to the conditions

After authorization and authentication, the external system can provide the API with HTTP requests containing the corresponding search parameters. The community communication platform will filter out the corresponding activity data according to the parameters and return it in JSON file format.

Examples of parameters are as follows:

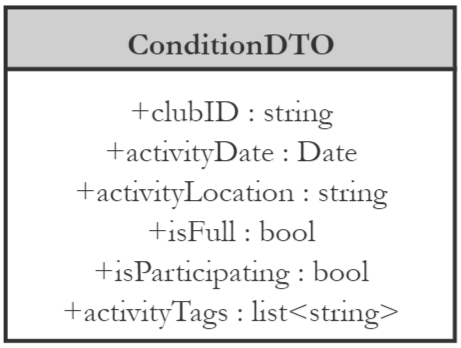
ARGUEMENT	VALUE	INTRODUCTION
clubID	c10001	The index of the club that carried out the activity.
activityDate	2021-06-15	The date of the activity.
activityLocation	JiShi building 434	The location of the activity.
isFull	false	Is the maximum number of people in the activity.
isParticipating	true	Whether the user sign up for the activity.
activityTags	[knowledge, software]	Descriptive label for the activity being carried out.

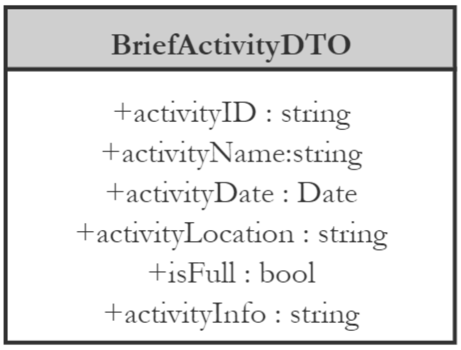
Therefore, the access URL is:

```
https://jiyu.com/activities/?clubID='c10001'&activityDate='2021-06-15'&isParticipating=true/
```

After passing the authorization authentication, you can return to all the activities held by the 'c10001' community on June 15, 2021.

In order to ensure the encapsulation and security of the data in the process of passing parameters or return values, DTO is used to wrap the data, the DTOs involved in the above activity interfaces include:





Therefore, the activity brief information is returned in JSON file format, such as:

```
{
    "status" : 0,
    "message": "SUCCESS",
    "data" : [{
        "activityID" : "a10001",
        "activityName" : "2021 Lawn guitar !",
        "activityDate" : "2021-06-15",
        "activityLocation" : "Front lawn of Southwest first
building",
        "isFull" : false,
        "activityInfo": "Welcome music lovers to join us!"
    },{
        "activityID" : "a10005",
        "activityName" : "2021 Lecture on mathematical modeling",
        "activityDate" : "2021-06-15",
        "activityLocation" : "A201",
        "isFull" : false,
        "activityInfo": "Full of dry goods, hair if you like math,
you must not miss it!"
    },{
        "activityID" : "a10010",
        "activityName" : "2021 SAP Employment sharing",
        "activityDate" : "2021-06-15",
        "activityLocation" : "JiShi building 434",
        "isFull" : false,
        "activityInfo": "Do you want to know about employment
benefits? Let's listen to the elders to share!"
    }]
}
```

Return the above JSON format data to the front end for processing, and then it can be rendered and displayed by the external system.

Sign up for activities through external system

After filtering the activity information through API in the previous step and returning to the external system, users can select the activity they are interested in enroll. The external system can call the API to submit the application information and wait for the approval of the activity.

Access URL through post:

```
https://jiyu.com/activities/
```

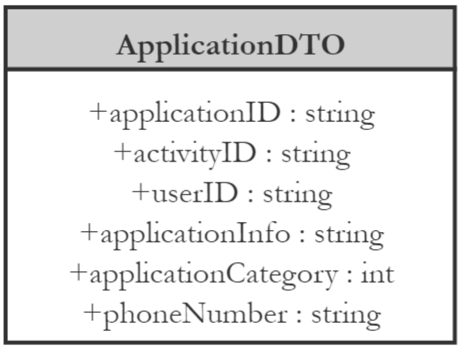
The data parameters submitted are as follows:

ARGUEMENT	VALUE	INTRODUCTION
activityID	a10001	The index of the activity attended by users.
userID	1956888	The index of the user attending the activity.
applicationInfo	"I'm glad to participate in the activity"	Personal information of participating activities.
applicationCategory	0	Types of applications.
phoneNumber	12323459856	Contact information of participating users.

The parameters are passed in JSON format in the body of the post request. The data example is as follows:

```
{
    "activityID" : "a10001",
    "userID" : "1956888",
    "applicationInfo" : "I'm glad to participate in the activity",
    "applicationCategory" : 0,
    phoneNumber : "12323459856"
}
```

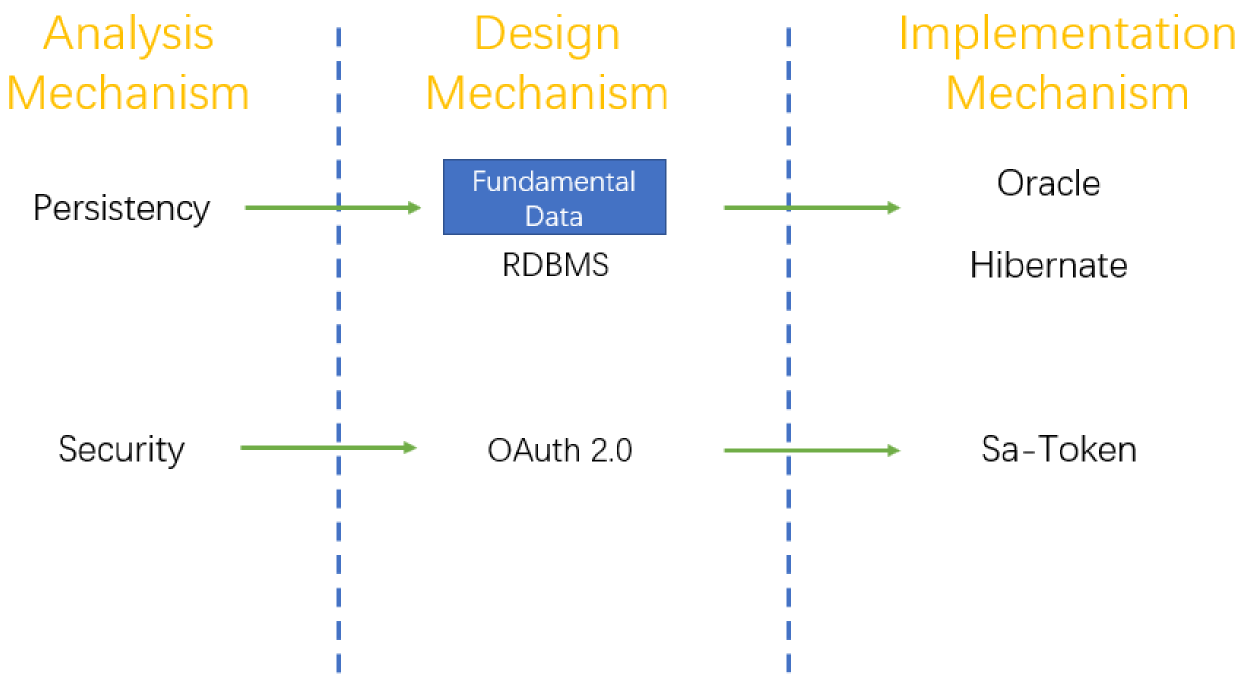
After the control layer component accepts the request, it automatically creates an index for the activity application content, and then encapsulates it as an application dto, which is passed down to complete the business logic. The dto content is as follows:



Finally, the execution result of the application operation is returned to the front end and displayed to the user.

3. Design Mechanism

Two design mechanisms of JIYU system are as follows:

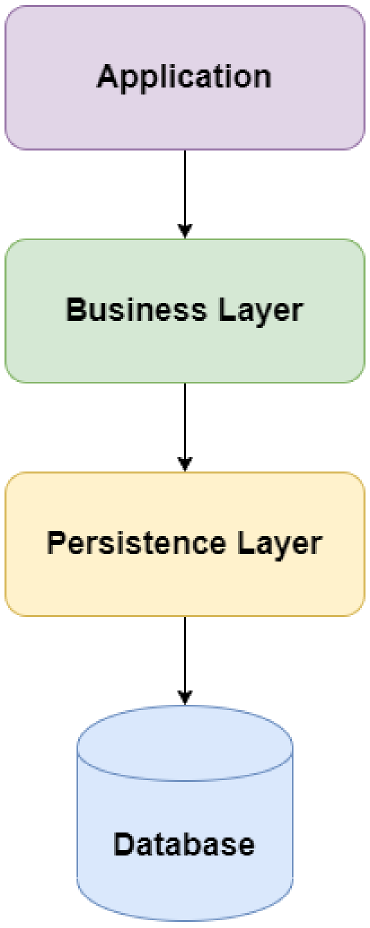


3.1 Data Persistence Mechanism

There are lots of data information in our JIYU(Meet what you will meet) system, which need to be preserved permanently for the realization of system functions. For example, the Club Union Supervisor needs to review the club leader's activity applications. As an object-oriented programming language, Java objects can only be saved in memory. Therefore, special databases need to be used for program data store, namely "Data Persistence". According to the requirements of our system, the relational database is selected for data storage.

Relational data and objects are two forms of business entities. Business entities are represented as objects in memory and relational data in databases. There are many association and inheritance relationships among objects in memory, which cannot be expressed by relational data. Therefore, Object Relational Mapping (ORM) is needed for data persistence.

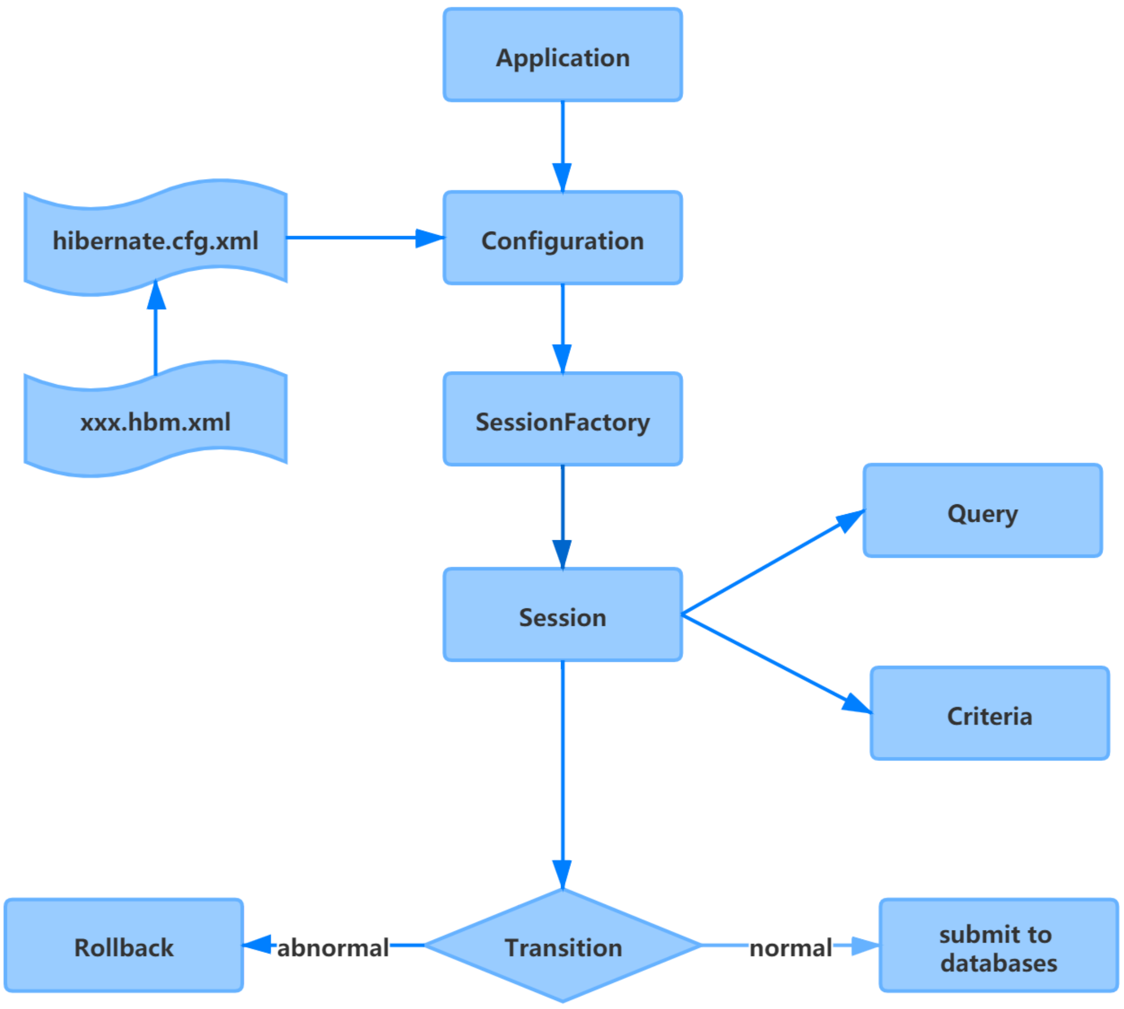
In order to achieve the goal, we implement a persistence layer between Business Layer and Database. Persistence layer is a kind of middleware, which can automatically store objects in databases, as well as load data from databases to memory. Under such an abstraction layer, the programs will indirectly access databases via session instead.



Hibernate

Hibernate is chosen as the persistence layer of our JIYU system. Hibernate is an open source and fully automatic ORM framework, which encapsulates JDBC with very lightweight objects, generating SQL statements and executing them automatically. Compared with MyBatis, Hibernate can automatically generate SQL statements, reduce the coupling between objects and DBMS, and have complete log systems and higher portability.

The mechanism of Hibernate is shown below:



Configuration Object

The Configuration object is the first Hibernate object created in any Hibernate application. It is usually created only once during application initialization, which represents a configuration or properties file required by the Hibernate.

SessionFactory Object

Created by Configuration object, SessionFactory object configures Hibernate for the application using the supplied configuration file and allows for a Session object to be instantiated. The SessionFactory is a thread safe object and used by all threads of an application.

Session Object

A Session is used to get a physical connection with a database, it provides an interface between the application and data stored in the database. The Session object is lightweight, short-lived and wraps the JDBC connection. Persistent objects are saved and retrieved through a Session object. The Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

Transaction Object

A Transaction represents a unit of work with the database. Transactions in Hibernate are handled by an underlying transaction manager and transaction (from JDBC or JTA).

Transaction is optional, Hibernate applications may manage transactions in their own application code instead.

Query Object

Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects. A Query instance is used to bind query parameters, limit the number of results returned by the query, and finally to execute the query.

Criteria objects

Criteria objects are used to create and execute object oriented criteria queries to retrieve objects. Criteria API is designed to build up criteria query objects programmatically where you can apply filtration rules and logical conditions.

3.2 User Access Control Mechanism

Our system involves many roles, including visitor, user, club member, club leader and so on. Different roles have different permissions. For example, the Club Union Supervisor has the authority to review club activity applications, while other roles cannot. We should ensure that every actor can get their own services or resources normally, so there is no doubt that this system should provide a comprehensive and friendly user access control mechanism. In addition, providing users with good experience by quick response is also one of our goals. Therefore, the following work is done.

Microservices Architecture

To make our platform easier to understand, develop and become more resilient to architecture erosion, we build JIYU with a Microservices-based Architecture, where the whole system is decomposed into different small loosely coupled services. In this microservices architecture, small autonomous teams are allowed to develop, deploy and scale their respective services independently. Appropriate language and tools can be chosen to build every specific service. And a lightweight centralized management can be used to coordinate these services.

In this architecture mode, the services that user requires are provided by different servers. Compared with the single application mode, system based on the microservices architecture mode can provide services faster. When one micro service fails, the other micro services will not be affected. Besides, each micro service can make full use of its hardware resources. And independent expansion will not affect other micro services.

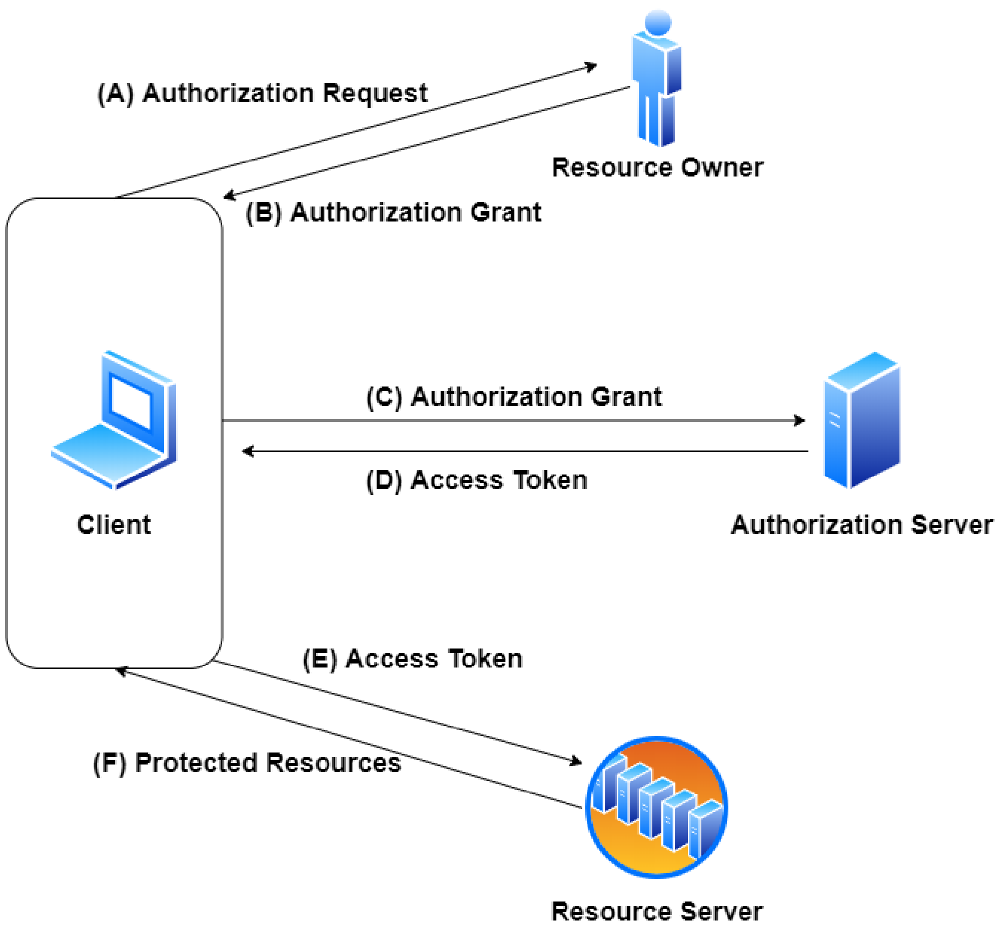
Authority Authentication Mechanism

In our JIYU system, we respond to users' requests in this microservices architecture mode based on OAuth 2.0 standard protocol and API Gateway.

OAuth2.0

Compared with cookie based authentication, token based authentication is stateless, can place Cross Site Request Forgery (CSRF), and supports multi site use. Therefore, our JIYU system uses OAuth2.0 protocol based on token for authority authentication.

The typical protocol flow of OAuth 2.0 is described well by this figure:



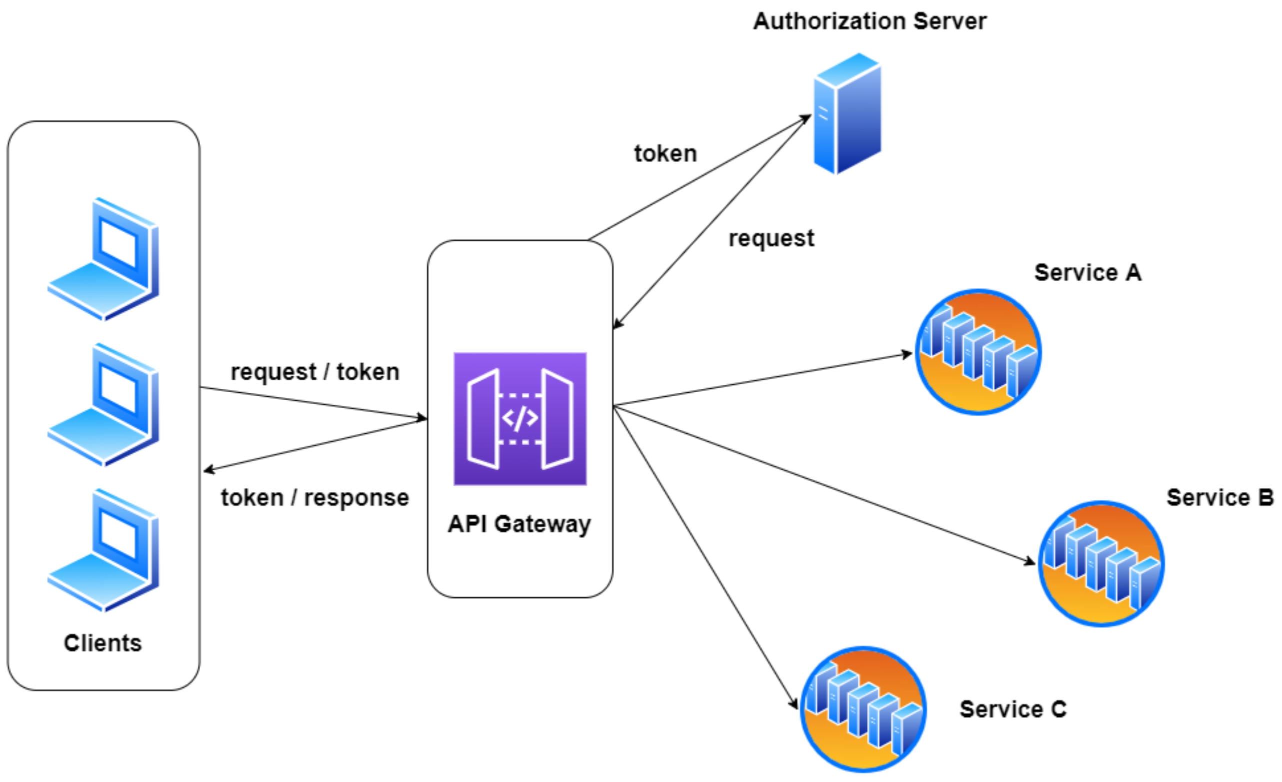
- The client requests authorization from the resource owner. The authorization request can be initiated directly to the resource owner. The client receives an authorization grant, whose type depends on the method the client requests authorization and types supported by the authorization server.
- According to the authorization grant, the client authenticates with the authorization server and requests an access token.

  The authorization server verifies the client's identity and issues an access token if the authentication is passed.
- The client requests the protected resource from the resource server and authenticates by presenting the access token. If the token is valid, the client's request will be served.

API Gateway

API Gateway is a standard component of microservices architecture, which is the conductor that organizes the requests being processed by the microservices architecture to create simplified experience for the user. The API Gateway takes all requests from the clients, then routes them to the corresponding microservice with request routing, composition and protocol translation. Typically, it processes a request by calling multiple microservices and aggregating the results. Therefore, the API Gateway reduces the number of round trips between the client and the application, aggregates background services, saves traffic, improves performance, and improves user experience.

With what we mentioned above, here is a simplified version of our platform's system architecture:



Sa-Token

Sa-Token is chosen as the authorization and authentication framework, which is a lightweight Java authority authentication framework.

Compared with other certification frameworks, Sa-Token has the following advantages:

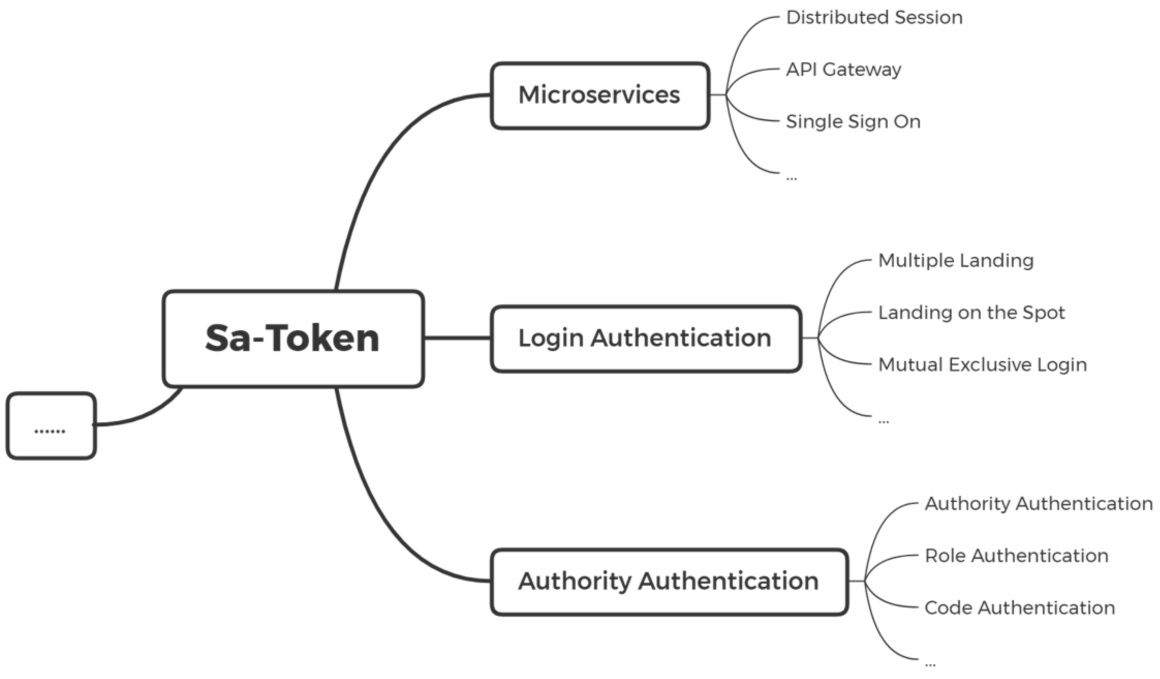
- Lightweight

  Sa-Token is a lightweight and zero configuration start-up framework, while Spring Security mainly provides security access control solutions for enterprise application systems therefore is difficult to learn.

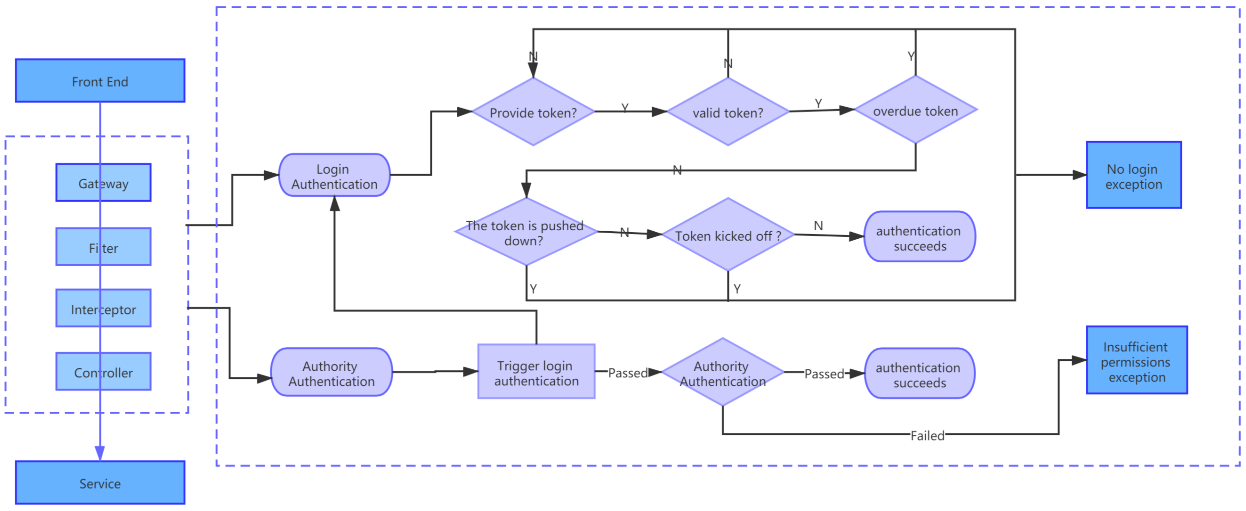
- Separation of front and rear platforms

  The design concept of separating front and back of traditional frameworks such as Apache Shiro, Spring Security lags behind, which makes them difficult to fit with the project; while Sa-Token is designed for the separation of front and back office architecture, which is more suitable for the system.

The functional structure of Sa-Token is as follows:



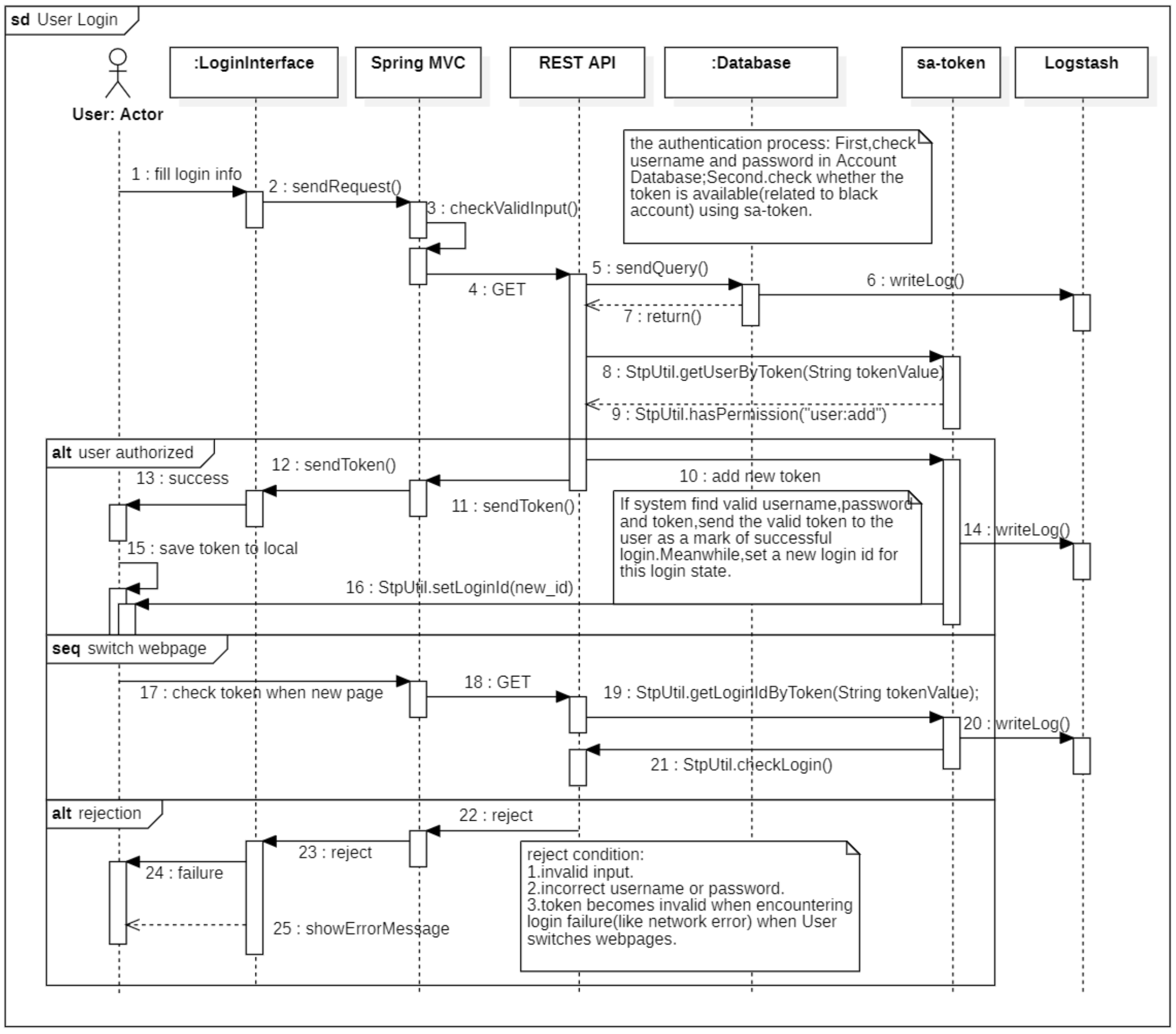
As the above figure shows, Sa-Token integrates the login authentication, authority authentication and dozens of permission related features under the microservices architecture mode. It mainly solves a series of authority related problems such as login authentication, authority authentication, session, single sign on, OAuth2.0, etc, and covers the solutions of most business scenarios. An amount of advanced features can be implemented with simple API calls.



As we can see in above figure, when the front end requests services from the resource owner, permissions need to be verified. Login authentication will be triggered before authority authentication. The authentication process is based on OAuth2.0 standard protocol, which is carried out according to whether the token is provided and the timeliness of the token. After the login authentication is passed, the authority authentication is further carried out. The back end will provide the requested service only if the authority authentication is passed.

4. Use Case Realization

4.1 User Login


sd User Login
User: Actor
:LoginInterface
Spring MVC
REST API
:Database
sa-token
Logstash
the authentication process: First,check username and password in Account Database;Second.check whether the token is available(related to black account) using sa-token.
1 : fill login info
2 : sendRequest()
3 : checkValidInput()
4 : GET
5 : sendQuery()
6 : writeLog()
7 : return()
8 : StpUtil.getUserByToken(String tokenValue)
9 : StpUtil.hasPermission("user:add")
alt user authorized
10 : add new token
11 : sendToken()
12 : sendToken()
13 : success
14 : writeLog()
15 : save token to local
16 : StpUtil.setLoginId(new_id)
If system find valid username,password and token,send the valid token to the user as a mark of successful login.Meanwhile,set a new login id for this login state.
seq switch webpage
17 : check token when new page
18 : GET
19 : StpUtil.getLoginIdByToken(String tokenValue);
20 : writeLog()
21 : StpUtil.checkLogin()
alt rejection
22 : reject
23 : reject
24 : failure
25 : showErrorMessage
reject condition:
1.invalid input.
2.incorrect username or password.
3.token becomes invalid when encountering login failure(like network error) when User switches webpages.

Login is a public use case for all users with a legal account. The process of login is demonstrated in the above sequence diagram.

According to our system design,we use Spring MVC to handle the login request, and REST API for sending the information command to the database that stores all of the users' accounts. For security authentication,since our system prefers light-weight authentication framework, we use Sa-Token (Github URL: https://github.com/dromara/sa-token) for account authority checking as well as login status examination.

The user inputs the username and the password in the login form. The information will be checked first by the Login interface() at the level of input correctness,that is,the system will reject login information with illegal format (for example, empty username and empty password). After that,the information will be parsed into JSON format. The login request will be sent to Spring MVC. Meanwhile, the JSON file with login information will be sent to the database through REST API, with corresponding query request sent to the database.

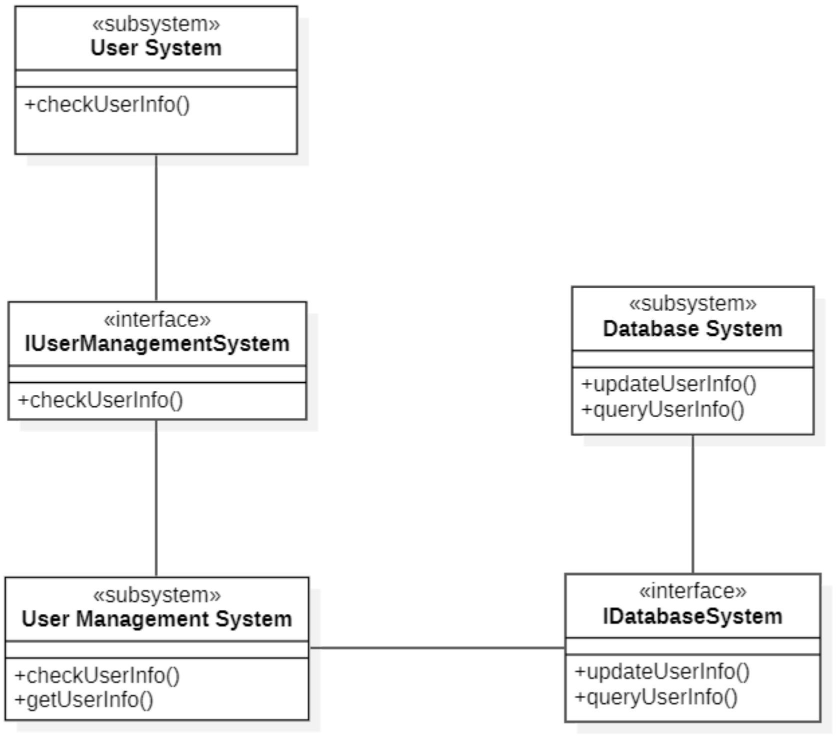
In the database, there is a user entity that stores all the information of the user. The database queries the user entity, finding whether the given information is in the user entity. If exists, then it returns the information and a success message to the user management subsystem,meaning that the username and the password can be found in the database.

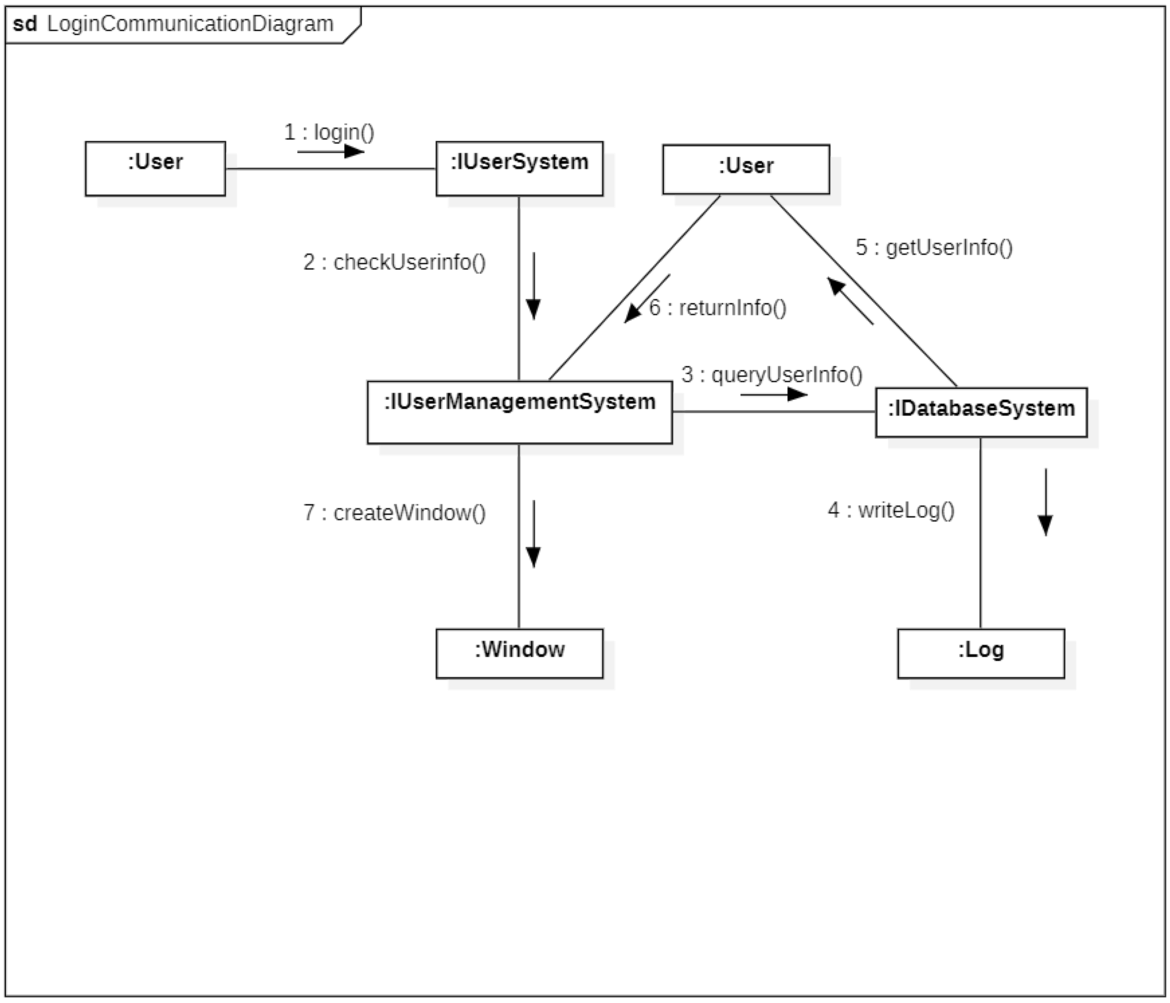
After getting the query result, our system will also launch a validation process to check whether the user account is allowed to login. The authentication process will be handled by Sa-Token. By invoking the StpUtil.getUserByToken method, it will returns whether the user's token is legal or not(the process will also involves interaction with black account database, which is out of our use case discussion).

If the token is proved to be legal (with illegal cases such as users has violating the laws,in which the user's token will be set illegal), a new token for the login will be generated, which will be sent to the login subsystem and saved to local.

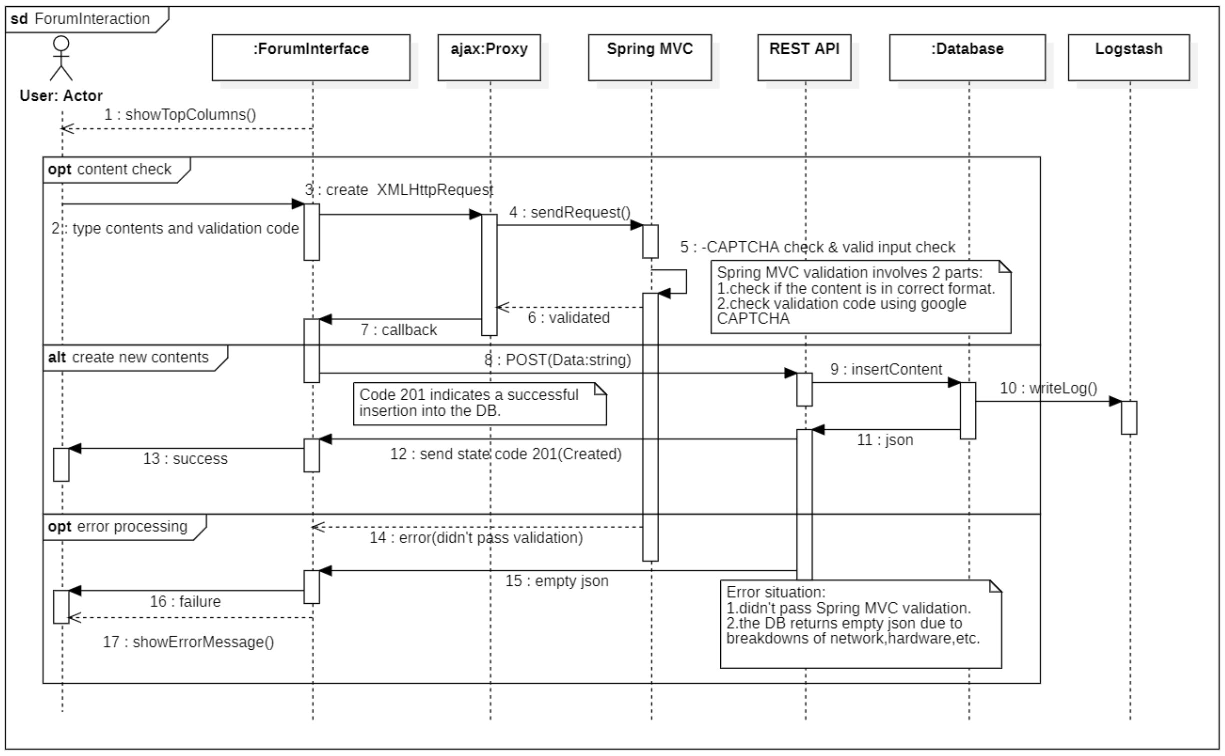
Whenever the user switch the webpage in our system,the token of the account will be checked by Sa-Token (through REST API) at the level of login status (the operation is powered by the StpUtil.getLoginIdByToken method) in case of system breakdown or other error cases. Specifically, when users' login status is affected by system breakdown, the token will be disabled and the system will reject the user for further using the service.

Corresponding subsystems and interfaces,together with their communication method, are represented at the below class diagram and the communication diagram:


«subsystem»
User System
+checkUserInfo()
«interface»
IUserManagementSystem
+checkUserInfo()
«subsystem»
Database System
+updateUserInfo()
+queryUserInfo()
«subsystem»
User Management System
+checkUserInfo()
+getUserInfo()
«interface»
IDatabaseSystem
+updateUserInfo()
+queryUserInfo()


sd LoginCommunicationDiagram
:User
1 : login()
:IUserSystem
:User
2 : checkUserinfo()
5 : getUserInfo()
6 : returnInfo()
3 : queryUserInfo()
:IUserManagementSystem
:IDatabaseSystem
7 : createWindow()
4 : writeLog()
:Window
:Log

4.2 Forum Interaction


sd ForumInteraction
User: Actor
:ForumInterface
ajax:Proxy
Spring MVC
REST API
:Database
Logstash
1 : showTopColumns()
opt content check
2: type contents and validation code
3 : create XMLHttpRequest
4 : sendRequest()
5 : -CAPTCHA check & valid input check
Spring MVC validation involves 2 parts:
1.check if the content is in correct format.
2.check validation code using google CAPTCHA
6 : validated
7 : callback
alt create new contents
8 : POST(Data:string)
9 : insertContent
10 : writeLog()
Code 201 indicates a successful insertion into the DB.
11 : json
12 : send state code 201(Created)
13 : success
opt error processing
14 : error(didn't pass validation)
15 : empty json
16 : failure
17 : showErrorMessage()
Error situation:
1.didn't pass Spring MVC validation.
2.the DB returns empty json due to breakdowns of network,hardware,etc.

Forum interaction is a use case including creating columns and leaving comments. The process of the use case is demonstrated in the above sequence diagram.

According to our system design,we use AJAX for data interaction. The Spring MVC framework,as is used in our previous use case realization, will be used here for handling request as well as validation process. The approved contents will be submitted to the database through REST API.

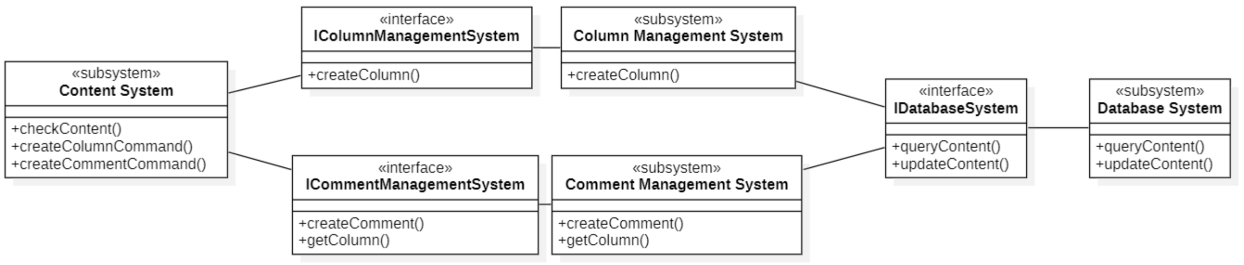
The user types contents he wants to submit,together with validation code for security check (for robot detection).After clicking the submit button, the corresponding HTML form data is generated, which is converted by AJAX techniques into the Java object. The corresponding request of this object is sent to Spring MVC.

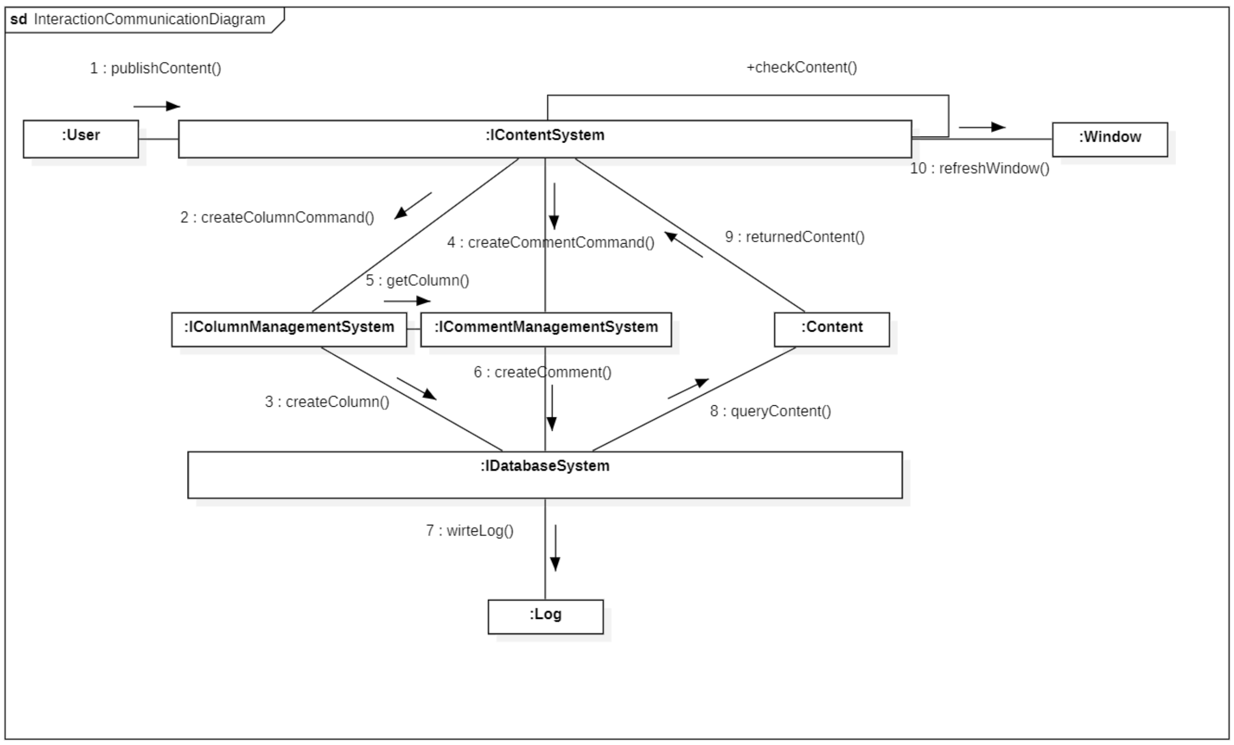
For validation process, Spring MVC will examine the correctness of the content format.(semantic level,not content level. the latter will be handled through the report mechanism) .Meanwhile, We use CAPTCHA , an external tool developed by Google for validation code (https://mvnrepository.com/artifact/com.github.penggle/kaptcha), to handle the pending validation code and prevent robot users. (Any content input with illegal format or wrong validation code will be rejected)

After passing the validation, the content to be submitted will be posted (POST command powered by REST API) to the content database,which will returns the result in JSON format,showing that the content is successfully loaded to the database. (If the operation goes wrong, an empty JSON will be returned and caught by error processing methods)

Finally, the system refreshes the webpage to display the forum with the new content.

Corresponding subsystems and interfaces, together with their communication method, are represented at the below class diagram and the communication diagram:



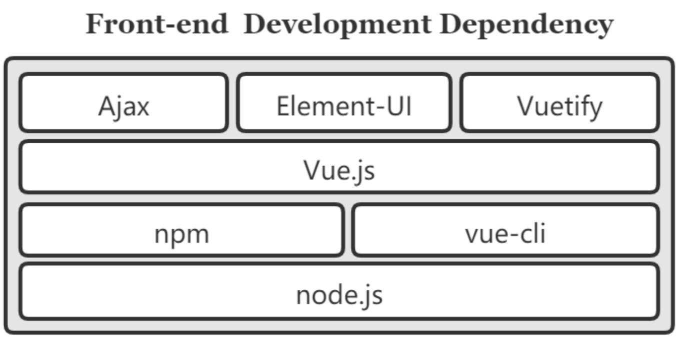


5. Progress on Prototyping

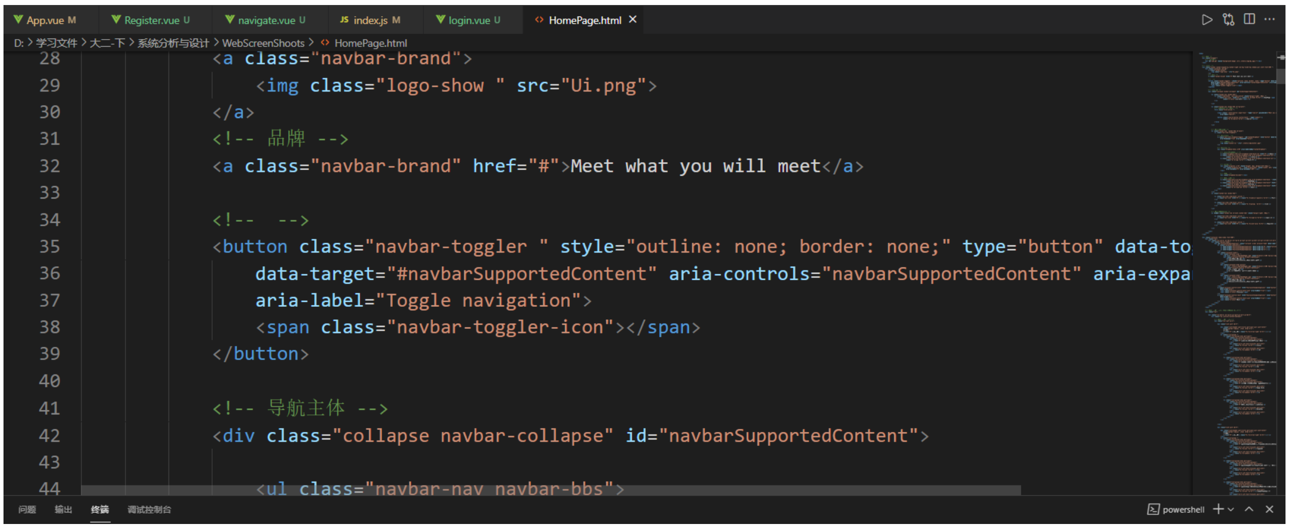
5.1 Front-end Prototyping

We use VUE.js framework to develop the front-end and Element-UI is used for the UI mode. At the same time, NPM package management tool is used as the front-end project package management tool, and Webpack is used. In addition, AJAX technology is used to communicate with the back-end.

After all, The dependency graph is as follows:



Taking the forum system for example, the front-end code is just as follows:



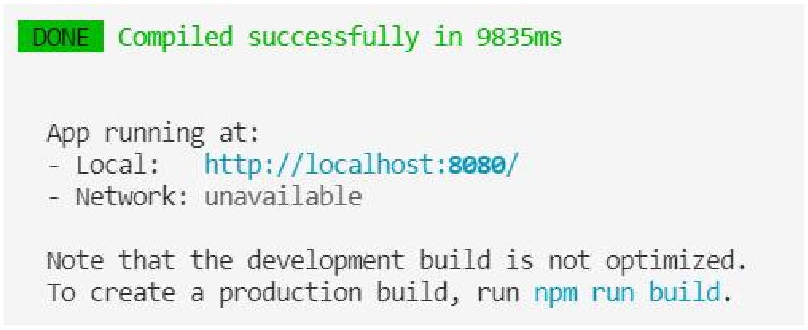
```
                <a class="navbar-brand">
                    <img class="logo-show " src="Ui.png">
                </a>
                <!-- 品牌 -->
                <a class="navbar-brand" href="#">Meet what you will meet</a>

                <!--  -->
                <button class="navbar-toggler " style="outline: none; border: none;" type="button" data-to
                    data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expa
                    aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>

                <!-- 导航主体 -->
                <div class="collapse navbar-collapse" id="navbarSupportedContent">

                    <ul class="navbar-nav navbar-bbs">
```

And we use **npm run serve** to start the front-end. After it is compiled, we can view the page at http://localhost:8080.

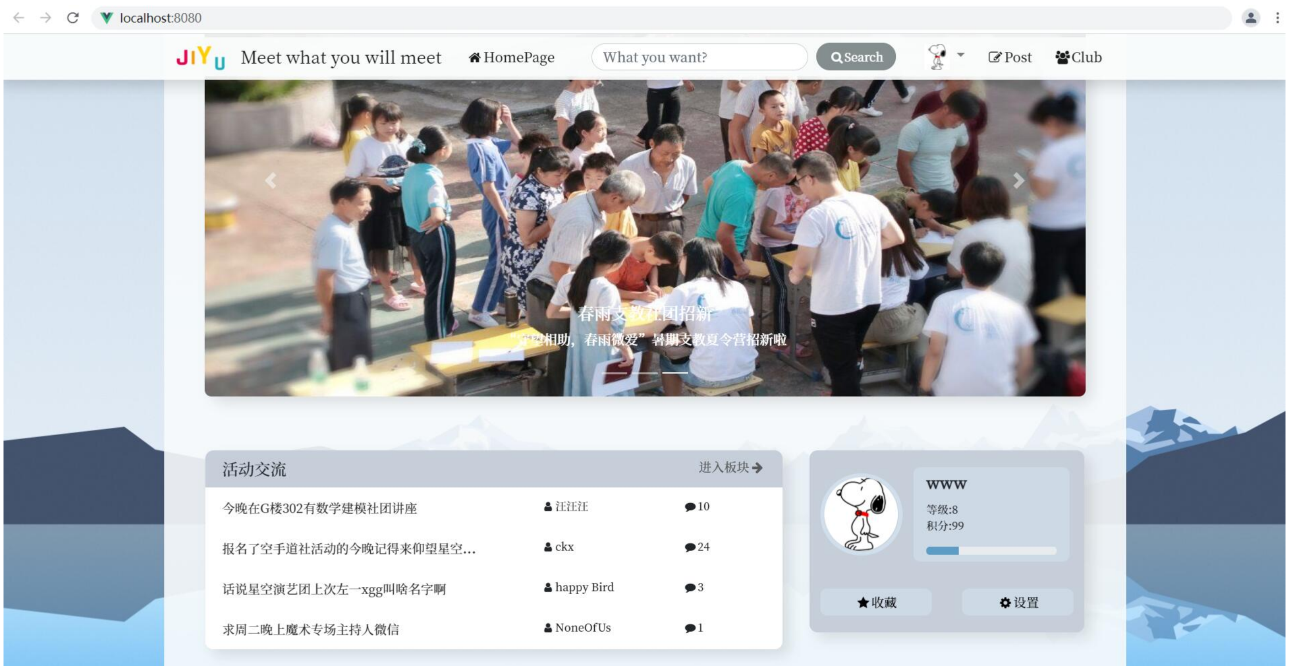


```
 DONE  Compiled successfully in 9835ms


  App running at:
  - Local:   http://localhost:8080/
  - Network: unavailable

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

By entering the above website, we can simply see our JIYU:



春雨支教社团招新

“守望相助，春雨微爱”暑期支教夏令营招新啦

活动交流 进入板块

今晚在G楼302有数学建模社团讲座	汪汪汪	10
报名了空手道社活动的今晚记得来仰望星空...	ckx	24
话说星空演艺团上次左一xgg叫啥名字啊	happy Bird	3
求周二晚上魔术专场主持人微信	NoneOfUs	1

www
等级:8
积分:99

收藏 设置

5.2 Back-end Prototyping

For the back-end of JIYU, we use Java for development. In order to separate the front and the back ends, we let the two systems communicate by using API documents. The front-end informs the back-end of its requirements(like get the most popular activities recently), and the back-end will write API document after completing such functions to the requirements.(For the specific architecture of the back-end, please refer to the second part of this document)

Firstly, we need to crate a class named Activity in our Java Project.

```
@Data
@Entity
public class Activity {
    @Id
    private String id;
    private String name;
    private Timestamp startTime;
}
```

And we use repository to store the set of the activity.

```
@Repository
public interface ActivityRepository extends JpaRepository<Activity,
String> {
    Activity findOneById(String id);
    Iterable<Activity> mostRecent(int t);
}
```

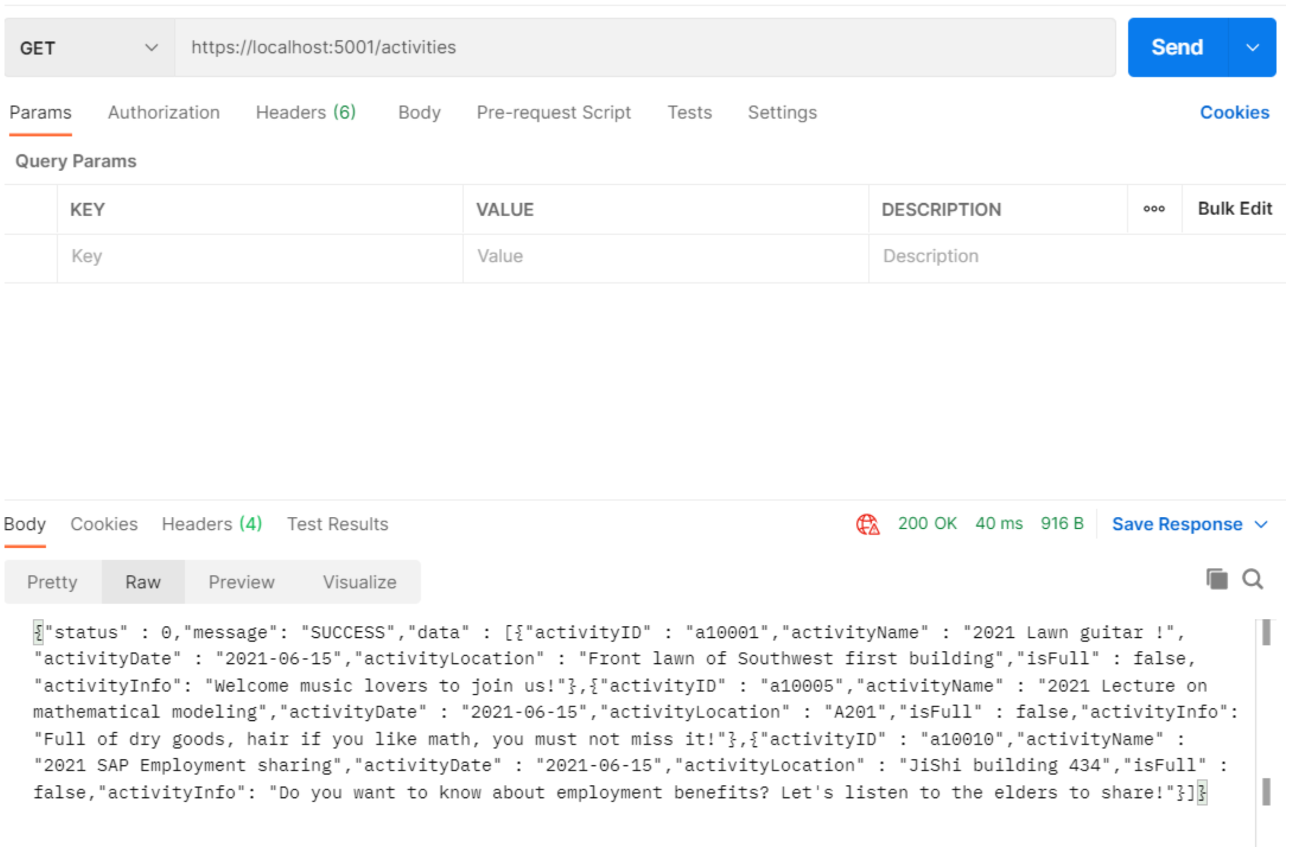
So the API function of getting the recent activities is as follows:

```
@RestController
public class ActivityController {

    @Autowired
    ActivityRepository activityRepository;

    @CrossOrigin
    @RequestMapping(value = "/recentActivities", method =
RequestMethod.GET)
    public String getRecentActivities() {
        ArrayList<Activity> at = (ArrayList<Activity>)
activityRepository.mostRecent(10);
        return at.toString();
    }
}
```

To test whether our API works properly, we can use some package grabbing tools like postman just as follows:



```
{"status" : 0,"message": "SUCCESS","data" : [{"activityID" : "a10001","activityName" : "2021 Lawn guitar !",
"activityDate" : "2021-06-15","activityLocation" : "Front lawn of Southwest first building","isFull" : false,
"activityInfo": "Welcome music lovers to join us!"},{"activityID" : "a10005","activityName" : "2021 Lecture on
mathematical modeling","activityDate" : "2021-06-15","activityLocation" : "A201","isFull" : false,"activityInfo":
"Full of dry goods, hair if you like math, you must not miss it!"},{"activityID" : "a10010","activityName" :
"2021 SAP Employment sharing","activityDate" : "2021-06-15","activityLocation" : "JiShi building 434","isFull" :
false,"activityInfo": "Do you want to know about employment benefits? Let's listen to the elders to share!"}]}
```

As for this API, we write it as follows so that the front- end programmer can easily communicate just by this API document:

ORDER NUM	REST API	INTERFACE
001	GET api/activities/	The function of this interface is to get the activities organized in the last week.

6. Project self-reflection

During the course of this semester, we have transformed a lot of unknown into known like:

UNKNOWN	KNOWN
The function of the use case diagram	It is used to describe our requirements.
The role of API	It can be used to simplify the application development process.
How to build system architecture	By reading the resources and asking the teacher, we learned to be a good system architecture.

Here are the self-reflection for each member in the team:

Mingjie Wang

In fact, before studying this course, I only had a very vague concept of software development. It can even be said that I only know that to make a software needs to write code, and that's all.
In the process of learning this course, I learn the function of use case diagram and how to translate my requirement into it. At the same time, through class learning and communication with teacher and classmates, I also understand the role of sequence diagram and activity diagram. With the deepening of learning, I also understand the important role of system architecture in a system.
Last but not least, one of the most important things is that I am better at working with team members. At the same time, I have a better understanding of what role I am more suitable to play in the team.
I believe that through this course, I can grow into an excellent software engineer in the future.

Ningyu Xiang

Through this semester I learned how to analyze a system from functional analysis to architectural analysis, and gradually gained a deeper understanding of the architecture of a software.
I learned how to draw use case diagrams and activity diagrams for system analysis, how to perform a simple layered architecture of a software system, how to improve the architecture of a software system to better fit its characteristics, and how to design the

physical mechanisms of the architecture. As I summarized the work done by my teammates, I also learned how they designed the system and made our system better in the exchange and discussion. But what we are doing now is in the design phase, and we still need to learn more about it when we develop it.

Kaixin Chen

Participating in a group to complete the UML project is a very rare experience. Through this period of experience, I learned how to get along with others, how to hand over and complete tasks, how to be a team member to make the team's progress more efficient. At the same time, I also learned a lot of basic techniques and methods of system analysis and design.

For example, at the beginning of processing use case realization, I was at a loss because of knowing too little about the system's architectural model, but after communicating with the member in charge of architecture in the group, I had a clearer understanding of the logical business of each architecture in the group and a basic idea of how to involve the users in the implementation of each part.

Even though the specific implementation of this part was still unknown, the logic of the cooperation of the techniques is clear. Really appreciate for this experience.

Zhongyue Chen

Through a semester of course work and course assignments, I experienced the entire process of building a system from scratch. In the process, I learned how to design the use cases involved in a subsystem and represent them clearly in the form of use case diagrams and use case specifications, as well as how to describe the functionality of these use cases in their entirety using activity diagrams and sequence diagrams.
In addition, I learned about the mechanisms that may be involved in the system design process, such as the data persistence mechanism and user access control mechanism used in this system. I have a rough understanding of the Hibernate and Sa-Token frameworks, but since the system is only in the design phase, I still need to learn the specific framework during the development process.

Liyou Wang

In this semester, the main problem I encountered in the course of system analysis and design was the lack of knowledge of the architecture level and the shallow understanding of the corresponding technology stack. The fixed programming mindset makes it difficult for me to consider the system from the designer's point of view, and most of the difficulties I face are caused by the programmer's mindset, but as I learn more, I can gradually consider the development architecture and patterns based on the characteristics of the system, and I think the development of this system design mindset will also motivate me to become a programmer who can design more clearly structured projects.

On the other hand, I also hope that this course will enable me to think and see beyond the programming perspective and to apply the analysis and development patterns I learned in the classroom to my future path, which will be a rewarding experience.

Known/unknown to Known/Known

In the process of implementing the logical architecture in the last assignment, it was clear to me that the services with high reusability in the business logic should be extracted as public services, based on the previous logical implementation only, in the process of implementing the technical architecture this time, combined with the microservice architecture and AOP design ideas, I think that the public services can be used as independent microservices to communicate with other microservices on the one hand, and on the other hand, the public services unrelated to the domain logic can be invoked by the application service dispatch and domain microservices together.

Contributions

This system analysis project has been discussed for three times. In the process of completing this task, all team members actively participate in the discussion and strive to complete their own tasks. Team members cooperate harmoniously, communicate and solve problems in time. The division of the work within the group was average and clear, as follows:

i. **Introduction**:
The introduction is written by Ningyu Xiang.

ii. **Architecture Refinement**:
The architecture refinement is mainly written by Liyou Wang while subsystem and interfaces are written by all members.

iii. **Design Mechanism:**
The design mechanism is written by Zhongyue Chen.

iv. **Use Case Realization:**
The use case realization is written by Kaixin Chen.

v. **Progress on Prototyping**:
The progress on prototyping is written by Mingjie Wang.

vi. **Project Self-reflection:**
All members in group participate in the project self-reflection.

vii. **Contributions of team members**:
The team member contributions are written by Mingjie Wang.

viii. **Document:**
The document is integrated and beautified by Mingjie Wang.

ix. **PPT Presentation**:
The presentation PPT is mainly made by Mingjie Wang.

STUDENT NUMBER	NAME	SCORE WEIGHT
1851055	Mingjie Wang	100%
1954098	Ningyu Xiang	100%
1951724	Kaixin Chen	100%

STUDENT NUMBER	NAME	SCORE WEIGHT
1851049	Zhongyue Chen	100%
1851231	Liyou Wang	100%