

**COVID-19Health Sign-in System**

2020/2021(2)



Group Member：秦政瀚 戈梓闻 叶凯玮

张玉凡 郑锴

Teacher： 颜世航

**Catalogue**



**[Chapter I: Project description 1](#_Toc21574)**

[1.1 Background of the project 1](#_Toc28624)

[1.2 Project objectives 1](#_Toc15781)

**[Chapter II: Project analysis 2](#_Toc19771)**

[.1 2. Technical feasibility 2](#_Toc25578)

[2.2 Economic feasibility 3](#_Toc32119)

**[Chapter 3: Project Design 3](#_Toc20023)**

[3.1 Overall project architecture 3](#_Toc8494)

[3.2 Project needs analysis 4](#_Toc16470)

[3.3 System outline design 5](#_Toc9126)

[3.4 Detailed System Design 7](#_Toc23693)

**[Chapter 4: User Manual 17](#_Toc22973)**

[4.1 Login Mini Programs 17](#_Toc14643)

[4.2 Check-in cards 18](#_Toc27058)

[4.3 Check punch records 20](#_Toc12220)

[4.4 Viewing and modifying user information 21](#_Toc19067)

[4.5 Administrator checks user punch 23](#_Toc6486)

**[Chapter 5: Data Structure Analysis 25](#_Toc24175)**

[5.1 Redis Data structure 25](#_Toc3412)

[5.2 MySQL Data structure 28](#_Toc3209)

[5.3 ThreadLocal Thread Isolation Data Structure 28](#_Toc6128)

**[Chapter 6: System Maintenance 29](#_Toc5272)**

[6.1 Teamwork Platform 29](#_Toc21556)

[6.2 Project back-end environment 30](#_Toc32276)

[6.3 Back-end deployment environment configuration 30](#_Toc27141)

[6.4 Project front-end environment 30](#_Toc15985)

**[Chapter 7: Team Division 30](#_Toc14755)**

# Chapter I: Project description

## 1.1 Background of the project

Since the outbreak of the new crown epidemic in 2020, our government has actively responded to the epidemic situation, and the measures to prevent and control the epidemic situation have achieved remarkable results. The people work together and cooperate actively to help the whole society resume work and return to production, and college students can return to campus to learn knowledge. But globally, the epidemic remains grim. In order to prevent the epidemic from spreading to the campus and win the battle of prevention and control of the epidemic situation in the education system, colleges and universities actively respond to the requirements of the Ministry of Education and local governments and carry out online "health cards" one after another, and strive to accurately grasp the students' dynamics. However, at present, the operating process of the epidemic health card system which has been released and run online is complicated and cumbersome, which is not easy for users to use, the interface data response time is longer, the high concurrency state is easy to appear downtime, the user information is easy to leak, and there are shortcomings in many aspects. The epidemic health card system developed by our team has been optimized in data storage, high concurrency processing, information security and so on, which can provide users with a better experience.

## 1.2 Project objectives

The accumulated technical reserve, computer network principle, operating system, database technology, software system design foundation and other theories are combined with the knowledge of data structure learned in this semester for practical combat, and the project experience is accumulated from the actual combat. Summarize their own shortcomings. At the same time, it provides an optimal solution to the epidemic health card system for schools or related application scenarios.

# Chapter II: Project analysis

## 2.1 Technical feasibility

### 2.1.1 front-end technical feasibility

The front-end technical solutions discussed by the team include html5、 WeChat Mini Program and Android development. The html5 interface is easy to implement, but independent of platform hosting; Android development, users need to download independent client APP, and need users to follow up APP update status, maintenance and implementation is more cumbersome; WeChat Mini Program can be released online with WeChat Mini Program platform. Comprehensive consideration, the team selected WeChat Mini Program as a front-end development program.

### 2.1.2 back-end technical feasibility

The team considers the following options in the back-end application development framework: the common Java Web project development process is cumbersome and the configuration cost is too high; Although the Spring framework has the advantages of reducing the coupling within the system with the IoC inversion of control and DI dependency injection, the cost of configuration is still huge. As a result, the team agreed to use the Spring Boot framework for development.

The team has some ambiguity in the selection of persistence layer framework. The main two options are MyBatis framework and JPA framework. JPA framework can directly persist complex Java objects and use simple, but SQL statements are automatically generated, so code readability is poor, and for some advanced business or complex queries need to be implemented manually SQL;MyBatis Although there are some shortcomings such as large code writing workload and poor database portability, there is no need to transplant other types of databases for this project, and project developers have a certain sql language skills, and MyBatis own dynamic SQL programming characteristics, Can better support advanced business query. As a result, the team finally decided to adopt MyBatis as the persistence layer framework.

In the aspect of database selection, the popular MySQL is selected as the main database. Taking into account the project's demand for fast data storage, the memory-based Redis is selected as the cache database, and the Redis has the characteristics of single thread and multiplexing non-blocking IO, which avoids unnecessary context switching and competitive conditions. Therefore, the speed of reading data is faster.

Back-end deployment selection Tencent cloud CentOS lightweight server, can quickly complete the construction and project deployment.

## 2.2 Economic feasibility

The cost of the project is mainly used for server rental, domain name purchase, related software product subscription. By using student discount, it costs 108 yuan to buy the right to use the server for one year ,29 yuan to buy domain names ,99 yuan to subscribe to RDM Redis visual management tools, and less labor costs and project maintenance costs at this stage. The total cost of the project is within the economic reach of the team.

# Chapter 3: Project Design

## 3.1 Overall project architecture

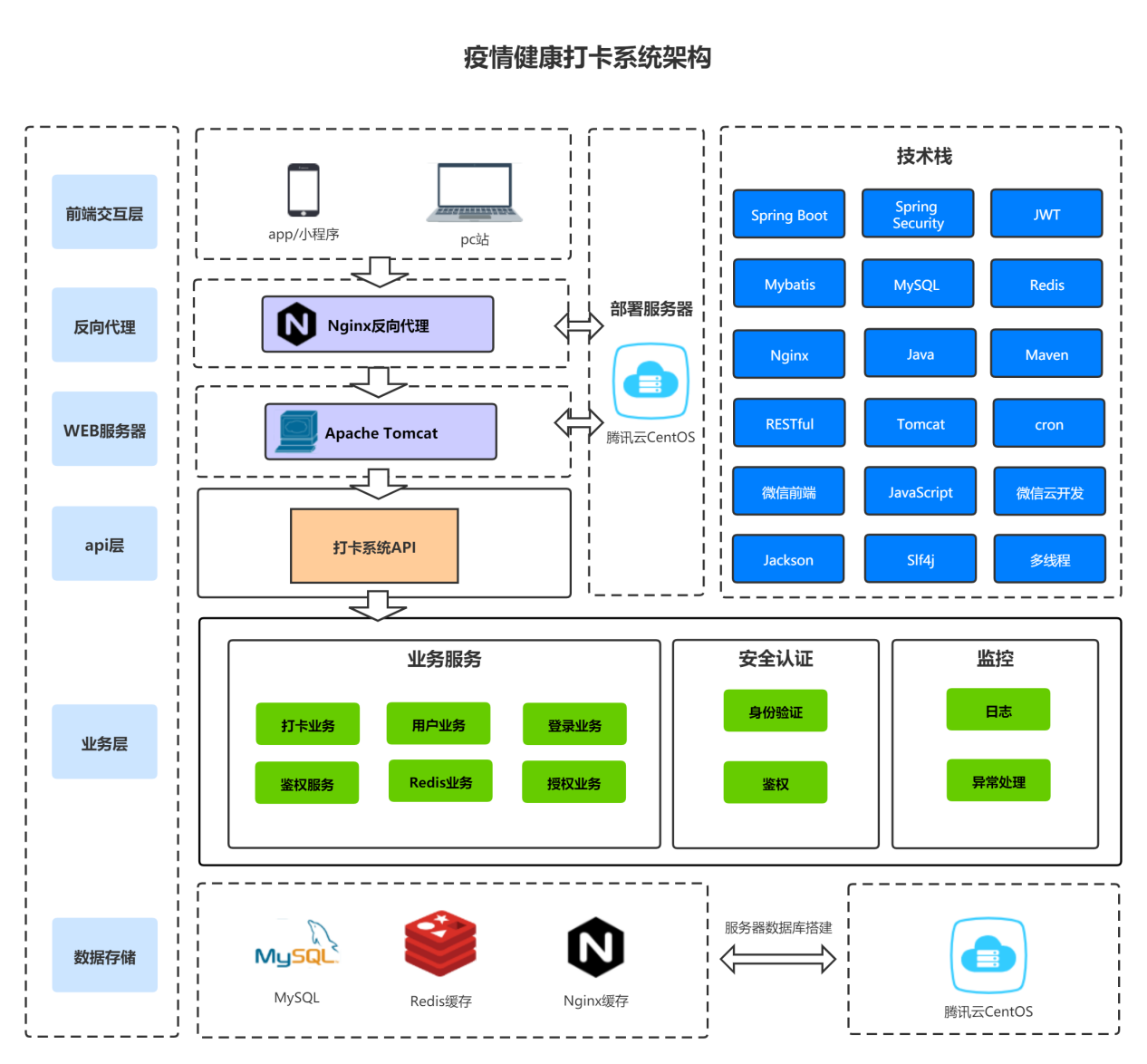


Figure 1 Chart of Health Card System

Description:

1. Thorough anterior and posterior separation
2. Client uses Mini Programs
3. The back-end service uses Java language to build engineering based on Spring Boot framework
4. Back-end services deployed to Tencent cloud CentOS servers
5. Tomcat server via Nginx reverse proxy
6. All services follow RESTful software architecture style
7. Mobile side based on Oauth、Java Web Token security authentication
8. Client (Mini Programs) acquires server resources by requesting API interface
9. MySQL system will be the main database, save project information and resources
10. By Redis, the system solves the problem of high concurrency and fast data access

## 3.2 Project needs analysis

### 3.2.1 functional requirements analysis

**Summary of System Functions:**

* WeChat authorized login
* Save user login status for a period of time
* User requests for authentication
* User punch (time, location, body temperature, health code, etc.)
* Users view personal punch information (day, month)
* User changes personal information (name, gender, class, specialty, etc)
* The administrator checks the punch information
* Administrator background management users (add, delete, change, check)

### 3.2.2 Interface Requirements Analysis

* Simple operation, beautiful interface.
* Provides the user each kind of operation prompt information.

### 3.2.3 Performance Requirement Analysis

* Instant visible: the user card processing results will be displayed in the interface immediately, users and administrators can quickly obtain card information.
* A complete user authentication system is established to encrypt and encapsulate the important data contained in the network request.
* Under a certain number of concurrency, the system should run quickly, efficiently, stably and reliably.
* The system structure has good expansibility and is convenient for future function maintenance and expansion.

## 3.3 System outline design

### 3.3.1 System Case Diagram



Figure 2 Examples of health punch system in epidemic situation

### 3.3.2 System Architecture Mode



Figure 3 Schematic Diagram of the System Architecture

### 3.3.3 functional characteristics

#### 3.3.3.1 WeChat authorized login

|  |  |
| --- | --- |
| User scenario | User to open the epidemic health card system WeChat Mini Program to perform login authorization |
| Functional description | WeChat authorized login |
| Priority | High |
| Input/preconditions | Click the WeChat Mini Program |
| Requirements description | 1. WeChat Mini Program one-click authorization login |
| Output/post condition | 1. Click login to enter the card system (if the first login will pop up user information perfect interface) |
| Supplementary note | After the user mobile phone number + password registration login |

#### 3.3.3.2 User punch case

|  |  |
| --- | --- |
| User scenario | The user opens the punch page to punch the card operation |
| Functional description | User punch |
| Priority | High |
| Input/preconditions | Open Punch Page |
| Requirements description | 1. the user fill in the information successfully |
| Output/post condition | 1. If the user punches for the first time on the day, the system displays the information about the success of the punch 2. If the user repeatedly punches the card that day, the system shows that the user has successfully punched the card |
| Supplementary note | No |

## 3.4 Detailed System Design

### 3.4.1 WeChat authorized login module



Figure 4 WeChat login logic



Figure 5 WeChat login sequence diagram

#### 3.4.1.1 WeChat authorization login process description

1. User click WeChat authorized login.
2. WeChat Mini Program gets temporary login credentials code, and returns to the developer server authentication authorization control class.
3. WeChat authorization control class obtains request body information and verifies request body information (code field can not be empty). If it is legal, it calls the WeChat login interface of authentication authorization service class.
4. The user login method of authorized authentication service class is called WeChat tool class, and the appid and secret key of temporary login voucher code、 WeChat Mini Program are passed to the corresponding method of WeChat tool class as parameters.
5. In the WeChat tool class, the code、appid、secret passed in is passed to the WeChat interface service through the RestTemplate encapsulation request.
6. The WeChat interface service returns openid、session\_key and other information to the WeChat tool class, and encapsulates the JSON format data into the corresponding data transmission object by calling the Jackson tool class.
7. A WeChat tool class returns the data transfer object to the WeChat login method of the authorized authentication service class, and obtains the openid. in the data transmission object through the corresponding method
8. By calling the method in the third party login service class, the incoming openid is passed in as a parameter.
9. The third party login service class reads the third party login information (such as login mode, openid, etc.) from the database and encapsulates it into the third party login entity class object through the incoming openid and corresponding methods, and returns it to the third party login service class.
10. If the lookup information is empty (that is, the first time the user logs in), the third party login entity class is created to record the third party login information.
11. Create the user entity class and record the initialized user information.
12. If the lookup information is not empty (that is, the user is already a system user), the processing login process method of this class is called. By calling the user service class method and passing in the user id as a parameter.
13. In the user service class, the method of finding user information according to user id is called to read the current user information from the database and encapsulate it as a user information object. The user service class returns the encapsulated user information object to the processing user login information method in the authorized authentication service class.
14. The token signature method of this class is called in the method of processing login process, the user information object is encapsulated into JWT load, and the encapsulated JWT load and the custom JWT key are passed in as parameters by calling the tool class.
15. JWT signature method in the tool class generates the load and key through the HMAC256 encryption algorithm token and returns it to the authorized authentication service class.
16. Authorization authentication service class returns token and user information objects to authentication authorization control class.
17. Authentication authorization control class encapsulates token and user information into a custom format response body and returns it to the WeChat Mini Program login page.

#### 3.4.1.2 Save Login Status within Fixed Time & Login Status Expired

1. Since the user triggers a request for each operation of the WeChat Mini Program, the back end verifies that the token in the request header is legal when processing each request, and if it is illegal (token error or token expiration), it prompts for a new login.
2. The signing token method of the JWT tool class, by setting the expiration time for the token, if the user is online, it will constantly refresh the expiration time of the user. When the user login Mini Programs again need to re-authorize login, if the token is not expired, you can directly enter the card interface.

### 3.4.2 High concurrency, high latency

If the front end requests direct access to the background MySQL database, it may instantly crush the underlying database and cause the business to stop. Or as the system business expands, the query conditions tend to be complicated, and the response time of the query results can not be guaranteed, which leads to the decline of the user experience. The problem of high concurrency and high delay has become a problem that must be solved.

#### 3.4.2.1 Access restricted section classes

1. back-end definition @RequestLimit annotation sets the default number and time of access for the same user to request the same request.

2. implement RequestLimitAspect section-oriented programming based on @RequestLimit. In this section class, the user's IP request address is obtained by custom IP tool class, and the method class and method of user request are obtained by reflection mechanism. The expiration time is set to the expiration time of the record. If the same request number of the same user exceeds the number of access limited by @RequestLimit, the corresponding custom global exception is thrown and fed back to the client to prompt the user to "exceed the limit of access times ".

#### 3.4.2.2 Redis high concurrency, high latency

1.Redis its own characteristics :1 Redis is based on memory, And the data is stored in a certain data structure, Therefore, the speed of reading and writing data is fast; ②Redis is single - threaded, Save the time for context to switch threads; ③Redis use IO multiplexing, non-blocking IO mechanisms, Can handle concurrent connections, Redis non-blocking IO implement event-driven IO reuse model internally, Efficiently handle active concurrent connections; ④Redis underlying storage uses a fast linked list quicklist data structure (compressed list ziplist combined with ordinary linked list), Make it fast insert, delete, It will not cause too much space redundancy.

2. back end business logic implementation: after the user logs in, the user hot spot information will be stored in the Redis cache, without the need to read the user data from the MySQL, which shortens the query time of the data.

The high concurrency resolution performance can be improved by Redis master-slave replication principle, breakpoint continuation, diskette replication, expiration key processing, and the high availability of redis server can be improved by sentinel mode and automatic partition.

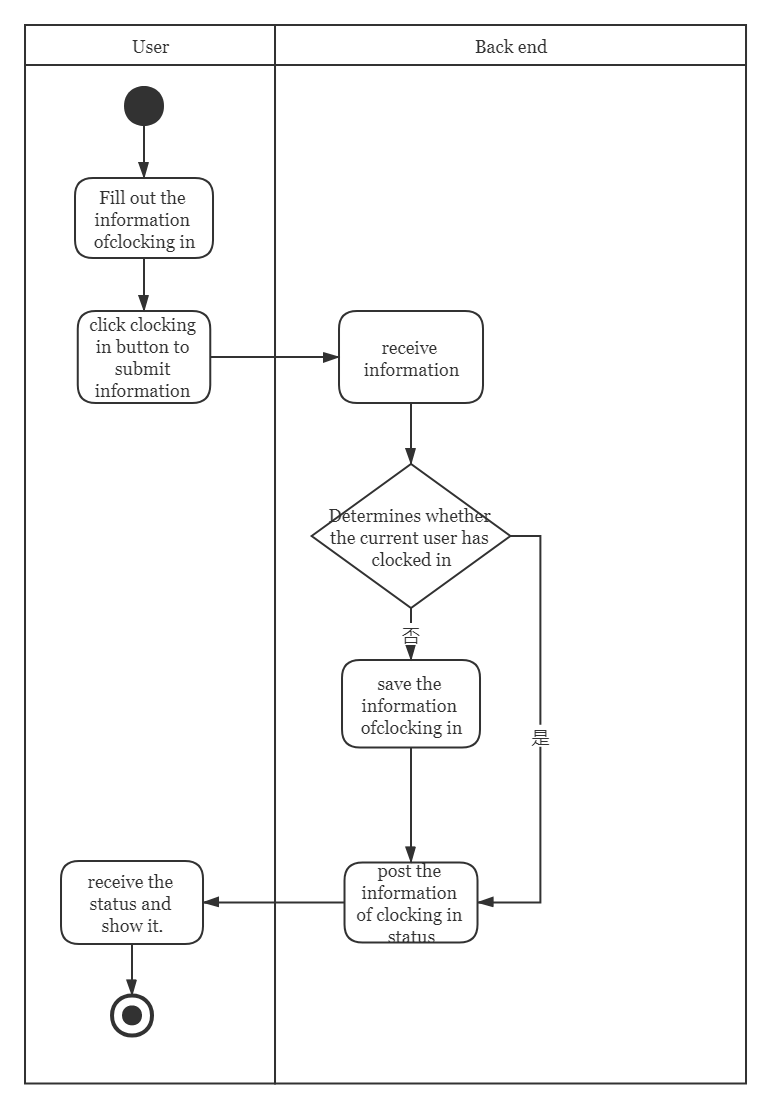
#### 3.4.2.3 Nginx High concurrent processing

1.Nginx also adopts epoll multiplexing IO model, which can handle a large number of concurrent connections.

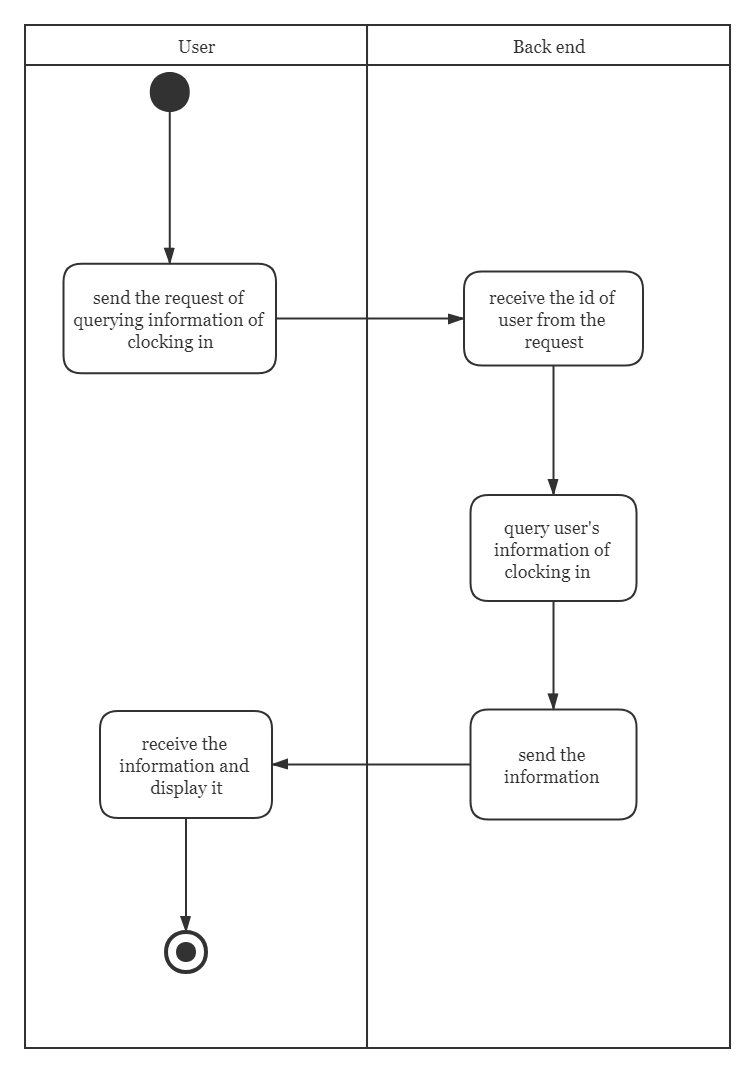
Later 2., a set of high concurrency optimization solutions can be specified by Nginx load balancing characteristics.

### 3.4.3 user punch module

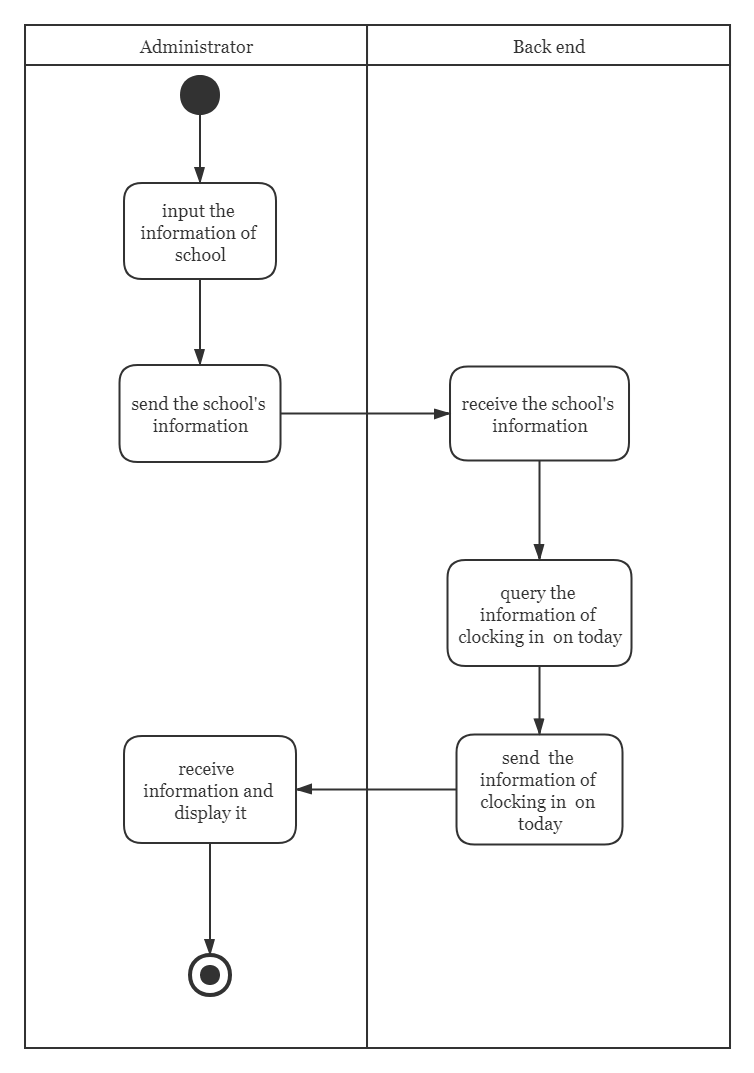
#### Activity Diagram 3.4.3.1



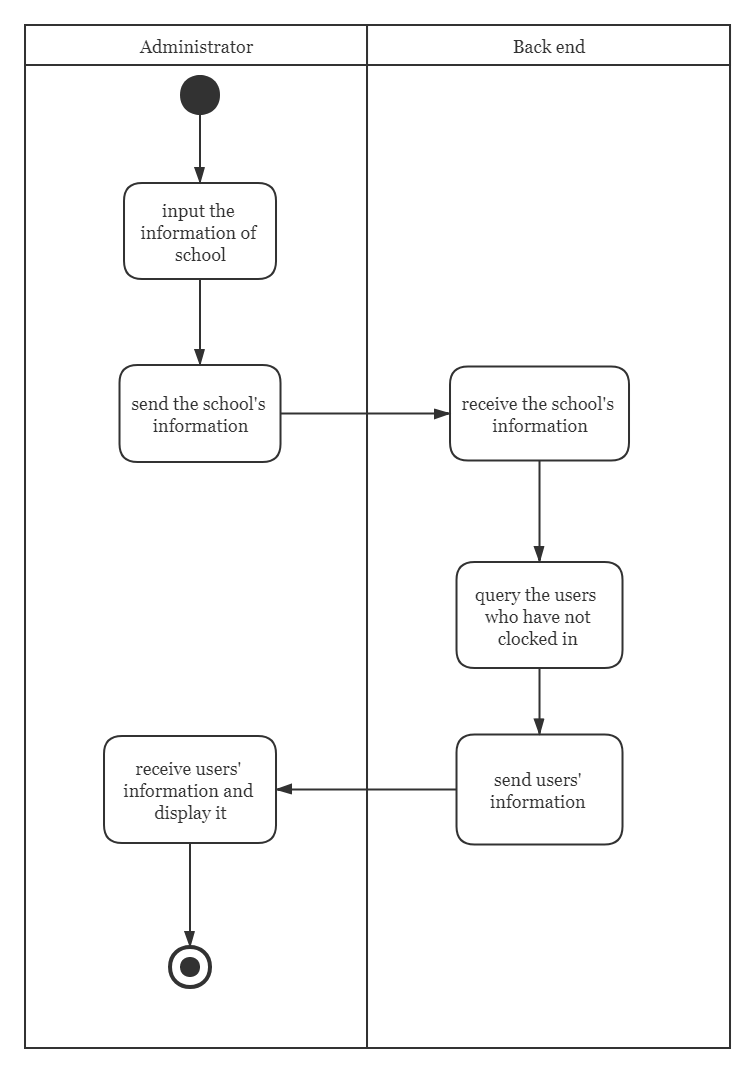
User card check-in activity chart



User Enquiry Past Card Record Activity Chart



The administrator inquires today card information activity chart



Administrators query today's untyped users

#### Class and Interface Design 3.4.3.2

1. **Timecard**

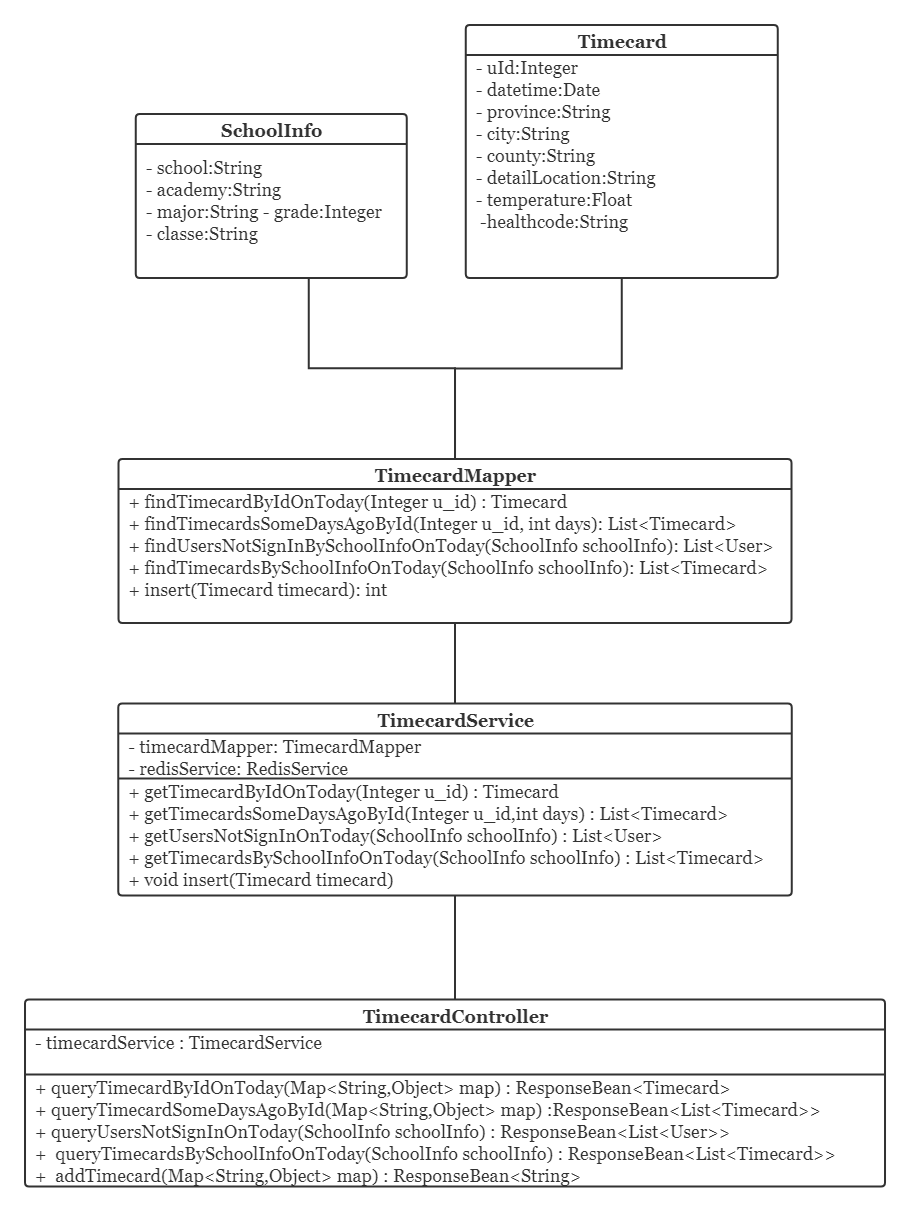
A simple class encapsulates Java punch information.

1. **SchoolInfo**

A simple class encapsulates Java user campus information.

1. **TimeCardMapper**TimecardMapper class implements the basic operation of MySQL data (add card information, query today card information, query user information, etc)
2. **TimeCardService**TimecardService class uses the TimecardMapper object to add and delete the data, and uses the RedisUtil to save the data punched today as a cache in the Redis.
3. **TimeCardContorller**

Timecar dCardContorller class receives data from the front end and encapsulates or judges it to the TImecardService object Process.



UML class diagram

### 3.4.4 Key Data Encryption

#### HMAC256 3.4.4.1

HMAC256 algorithm is used to encrypt the token signature method JWT the project. HMAC is a way to construct message authentication using a one-way hash function. HMAC operation uses Hash algorithm to generate a fixed length message summary as output with a message M and a key K as input. The key length of the HMAC can be of any size. If it is less than n (the size of the Hash output value), it will weaken the security of the algorithm. The method to solve the above defects is to use a strong pseudo random generator to generate the key randomly, and the key needs to be updated periodically, which can reduce the risk of weak key of hash function and the damage caused by key exposure.

# Chapter 4: User Manual

## 4.1 Login Mini Programs

### 4.1.1 Function Profile

User one-click authorization to click login.

### 4.1.2 operations

Enter the Mini Programs initial interface and click "Login ".



As shown above, click "login" and jump to the check-in card interface.



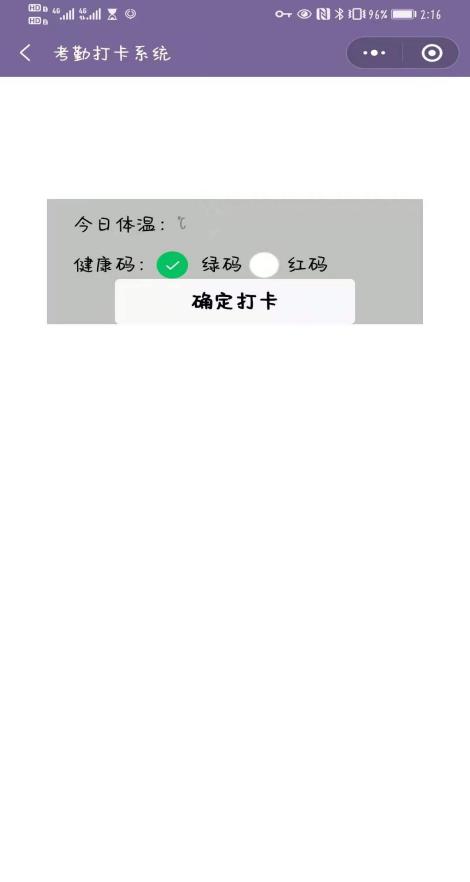
## 4.2 Check-in cards

### 4.2.1 Function Profile

This function is used for user card to submit today's body temperature, whether green code and other information.

### 4.2.2 operations

Click the middle blue start punch button.



As shown above, enter body temperature, select green code or red code and click "OK punch ".



As shown above, the interface automatically jumps, the "start punch" button changes to "already punched" and turns gray, and pops up the "punch success" prompt, if clicked again, it will prompt "not to repeat the punch ".

## 4.3 Check punch records

### 4.3.1 Function Profile

This function is used for users to view history card records.

### 4.3.2 operations

Click on the upper right corner of the punch interface "punch record ".





As shown above, the page jumps to display information such as punch time, punch location, body temperature and green code.

## 4.4 Viewing and modifying user information

### 4.4.1 Function Profile

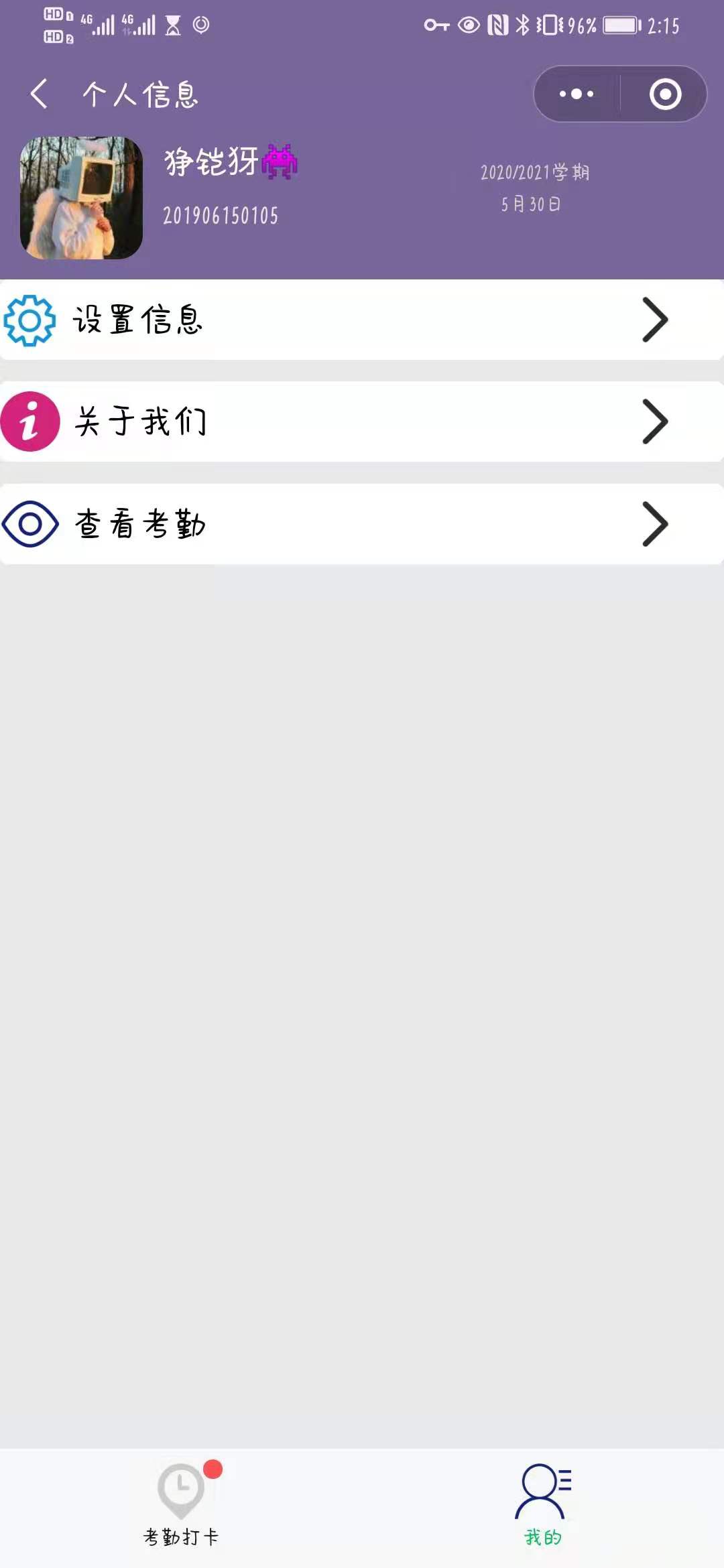
This feature is used to bind personal information, view personal information and modify it for the first time.

### 4.4.2 operations

Click on the lower right corner of the punch interface "mine ".



As shown below, the interface jumps to the personal information interface.



Click on "Settings Information ", the interface jump as shown below.



You can modify school, gender and other information, click OK to modify to save the modified information.

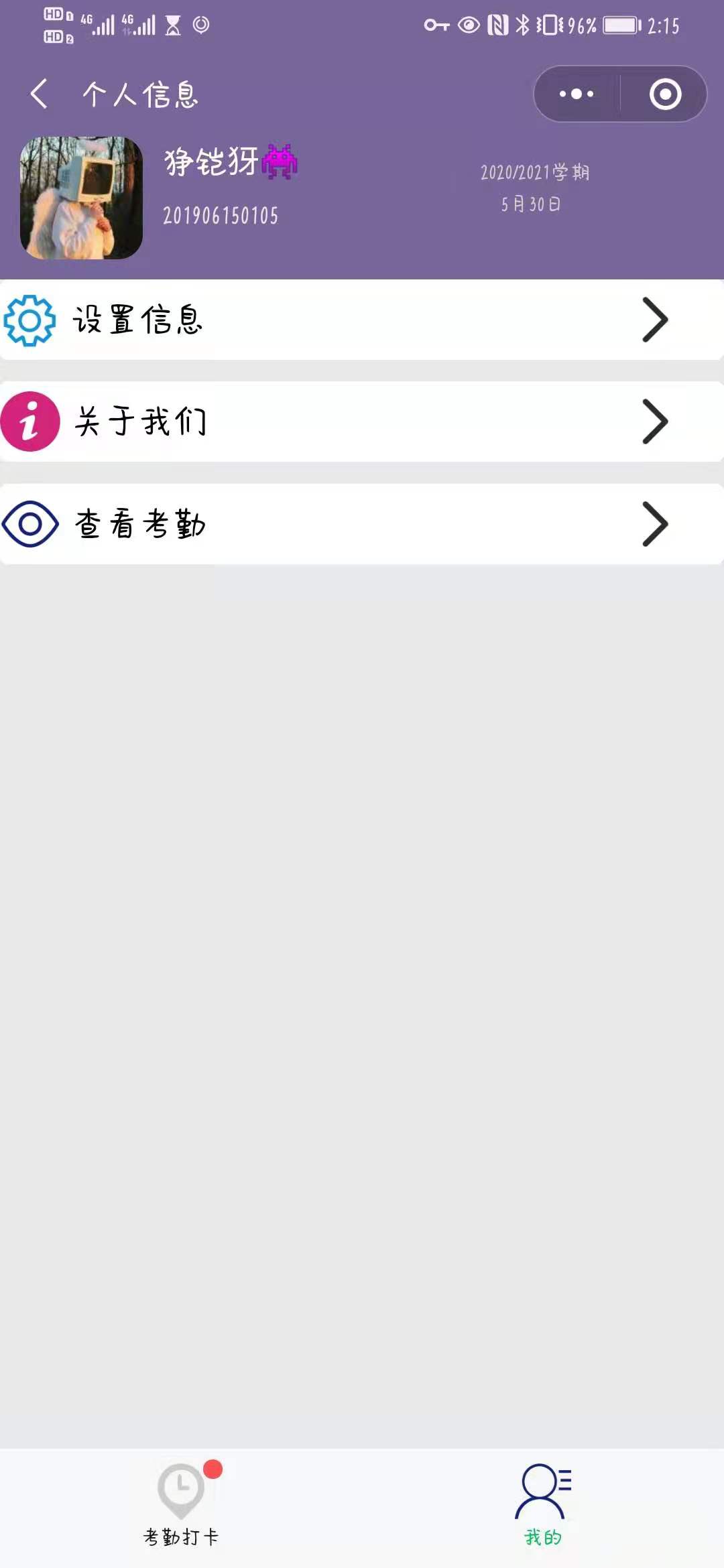
## 4.5 Administrator checks user punch

### 4.5.1 Function Profile

This function is used for administrator users to view all user card attendance.

### 4.5.2 operations

Click "View attendance ".



Page jump as shown below, showing the user and its card information.



Click on the right arrow, the information interface jumps to display the punch user and its punch information.



# Chapter 5: Data Structure Analysis

## 5.1 Redis Data structure

### 5.1.1String type

1. Redis designed a simple dynamic string SDS (Simple Dynamic String) as the underlying implementation, SDS the object contains three properties:

* len saved the actual length of the string.
* free represents the number of bytes not used in the buf.
* buf[] array is used to save each character element of a string.

1. c strings in the language do not record their own length, each time the length of the acquisition string will be traversed, the time complexity is O (n), and the Redis acquisition string as long as read the value of the len, so the time complexity is O(1).. The
2. SDS provides two strategies: spatial preallocation and inert space release. When allocating space to strings, more space is allocated than actually, which can reduce the number of memory reallocations caused by continuous string growth.
3. SDS is binary secure, can store strings as well as binary files (such as pictures, audio, video and other files of binary data).

### 5.1.2Hash type

1. There are two ways to implement Hash objects ziplist、hashtable, in which the hashtable storage mode is key of String type and is also stored in the form of key value.There are four attributes in the hash table structure definition:

* DictEntry \*\*table： hash table array.
* Unsigned long size：hash table size.
* Unsigned long sizemask： used to calculate index values.
* Unsigned long used：hash the number of nodes in the table.

1. Compressed list (ziplist) is a sequential data structure composed of a set of continuous memory blocks. Compressed list can save space and use multiple nodes to store data in compressed list. Compressed lists are one of the principles underlying the implementation of list keys and hash keys. The memory structure of compressed lists contains the following nodes:

* zlbytes：4 bytes, record the number of bytes in memory occupied by the compressed list.
* zltail：4 byte, records the offset of the end node of the table from the starting address, which is used to quickly locate the address of the end node.
* zllen：2 bytes, record the number of nodes in the compressed list.
* entry： represents each node in the list.
* zlend： special end symbol for a compressed list xFF'.0

1. Each entry node in the compressed list consists of three parts:

* previous\_entry\_length： represents the length of the previous node entry and can be used to calculate the address of the previous node because the address of the node in the ziplist is continuous.
* encoding： save content content type and length.
* content： save the content of each node.

### 5.1.3List type

1.linkedlist： is a two-way linked list, like a common linked list, with pointers to front and rear nodes. The time complexity of insertion, modification, update is O(1), but the time complexity of query is O (n).

2.quicklist： a bidirectional linked list of ziplist. Since macroscopically quicklist is a bidirectional linked list, it has the advantage of bidirectional linked list; microscopically, it is a piece entry nodes, each piece of entry node memory is continuously and sequentially stored. The time complexity of (log2n) can be located by binary search.

The list in the 3.Redis can block queues, combined with lpush and brpop commands. Producers use lpush to insert elements from the left side of the list, and consumers use brpop commands to get elements from the right side of the queue for consumption.

### 5.1.4Set type

The underlying implementation of the 1.Set is hash tables and intset. intset integer set, there are three attribute values encoding、length、contents[], representing the encoding mode, the length of the integer set, and the element content.

2. an element is added to an integer collection, the collection is upgraded beyond the length of the original collection, as follows:

* First extend the size of the underlying array, and the type of the array is the type of the new element.
* The elements in the original array are then converted to the type of the new element and placed in the corresponding position of the extended array.
* Integer set upgrade will not be downgraded, encoding will remain in the upgraded state.

### 5.1.5ZSet type

1.ZSet is an ordered set, and the underlying implementation is ziplist and skiplist.

2.skiplist is also called jump table, jump table is an orderly data structure, jump table through each node to maintain multiple pointers to other nodes, so that some unnecessary nodes can be skipped, thus speeding up the search, delete and other operations. The jump table contains the following attributes:

* Header： points to the head node of the jump table.
* Tail： points to the tail node of the jump table.
* Level： record the number of nodes with the largest number of layers in the current jump table (except header nodes).
* Length： record the length of the jump table, that is, the number of nodes the jump table currently contains (except the header node)

The Redis jump table node is represented by a zskiplistNode structure, which is used to save the relevant information of the jump table node. The structure contains the following attributes:

* Layer (level): each layer has two attributes: forward pointer and span. The forward pointer is used to access other nodes located at the end of the table, while the span records the distance between the forward pointer to the node and the current node. When the program traverses from the header to the end of the table, access is carried out along the forward pointer of the layer.
* Back (backward) pointer: it points to the previous node located at the current node. The back pointer is used in the program from the end of the table to the header.
* Score (score): in the jump table, nodes are arranged from small to large according to their saved scores.
* Member object (obj): the member object saved by each node.

## 5.2 MySQL Data structure

#### 5.2.1B+Tree



Figure 6 B+Tree Structure diagram

1. B+Tree is an optimization on the basis of B-Tree, which makes it more suitable for the implementation of external storage index structure. InnoDB storage engine adopts B+Tree to realize its index structure.
2. B-Tree, each node contains not only the key value of the data, but also the data value, and the storage space of each page is limited. If the data data is large, the number of key that each node (i.e. one page) can store will be small. While in the B+Tree, all data recording nodes are stored on the same layer of leaf nodes in order of key value, but only key value information is stored on the leaf node, which can greatly increase the number of key values that each node can store. Reduce the height of the B+Tree. B+Tree differ somewhat from B-Tree:

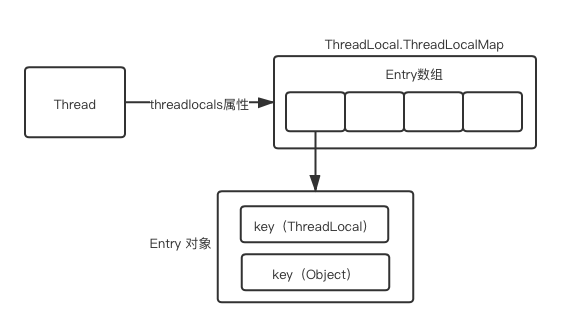
* Non-leaf nodes only store key value information.
* There is a chain pointer between all leaf nodes.
* Data records are stored in leaf nodes.
* Generally there are two heads of pointers on the B+Tree, one to the root node and the other to the leaf node with the smallest key value, and there is a chain ring structure between all leaf nodes, that is, data nodes. Therefore, two kinds of search operations can be carried out on the B+Tree :1 the range search based on the primary key and the paging search can be carried out randomly from the root node.

## 5.3 ThreadLocal Thread Isolation Data Structure

#### 5.3.1ThreadLocalMap

Projects use ThreadLocal to isolate data between threads when dealing with multithreading, ThreadLocal the essence of data isolation is that each thread Thread maintains its own ThreadLocals variables, and when each thread creates a ThreadLocal, Data is actually in the threadLocals (source code declares threadLocals as a ThreadLocalMap type) variable, other threads have no way to obtain, thus achieving isolation.

1. ThreadLocalMap the underlying data structure is very similar HashMap, but the Map interface is not implemented. In fact, the source code packages the ThreadLocal object as the specific content of the key, storage Object as a Entry object. Then the array of type Entry is placed in the table attribute of the thread to store the local variables. ThreadLocalMap structure is as follows:



1. The reason why ThreadLocalMap use Entry arrays: in fact, a thread can have multiple ThreadLocal to store different types of objects, but these different types of objects will be placed in ThreadLocalMap example of the current thread, so arrays are needed to store them.
2. ThreadLocalMap Hash conflict resolution and Hash conflict resolution :1 ThreadLocalMap gives each ThreadLocal object a threadLocalHashCode, when stored During the insertion, Location to the corresponding position in the table based on the hash value of the ThreadLocal object, Then make conditional judgment: if the current position is empty, initializes a Entry object to be placed in this position, If the current position is not empty, Find the next empty position, Until it is empty. The probability of set and get conflict is greatly reduced, An open-access Hash conflict resolution; Entry actually inherits from WeakReference weak references, Even if a ThreadLocalMap Hash conflict, After the original Entry was replaced, i.e. there is no strong reference pointing to the original Entry, So the Entry will be recycled the next GC, which greatly reduces the debris generated after frequent Hash conflicts.

# Chapter 6: System Maintenance

## 6.1 Teamwork Platform

Collaborative development platform: Github

Github address (currently private warehouse): https://github.com/Qin10/Covid19SignInSystem.git

## 6.2 Project back-end environment

Integrated development environment: IntelliJ IDEA、Eclipse、Visual Studio Code

Development languages: Java、SQL

Database: MySQL、Redis

Database visualization tools: Navicat Premium 12、RDM

Git tools: Git Bash

Security Terminal Simulation Software: Xshell 6

SFTP /FTP Client: Xftp 7

Web Debugging Tools: Postman

Log output: MDC、Slf4j

## 6.3 Back-end deployment environment configuration

### 6.3.1 Server Configuration

Server: Tencent Cloud Server

Server operating system: CentOS 7.6 64 bit

Server specifications :1 core 2 G

Server Public Network IP：121.5.115.174

### 6.3.2 server environment

JDK version :1.8

Database version: MySQL 8.0.25、Redis 5.0.5

Server configuration: Tomcat、Nginx (Nginx reverse proxy Tomcat)

Server domain name: zjutleo.cn

## 6.4 Project front-end environment

Integrated development environment: WeChat developer tools

Development languages: JavaScript

Deployment environment: Tencent cloud server

Web Debugging Tools: Postman

# Chapter 7: Team Division

|  |  |  |
| --- | --- | --- |
| Name | Role | Responsibilities |
| Qin Zhenghan | Leader, backend development | Project requirements analysis, back-end project architecture, SpringBoot+MyBatis+JWT+Redis+Wechat integration, database table design, user-related services, server build and deployment, interface testing and project integration, documentation |
| Gozi | Team members, front-end development | Project requirements analysis, front-end project architecture, front-end core business writing, front-end project docking, interface beautification, document writing |
| Yi Kaiwei | Team members, backend development | Project requirements analysis, back-end core business writing, database table design, project interface testing, project document writing, document translation |
| Zhang Yufan | Team members, front-end development | Analysis of project requirements, development of front-end project testing, PPT and documentation |
| Zheng Cheng | Team members, front-end development | Analysis of project requirements, development of front-end project testing, PPT and documentation |