Design and Implementation of Government Affairs Management System Oriented to Grassroots Community

Min-Yu Zhou[1†], Peng Nie[2*], Quan-Sheng Zeng[1†], Hao-Bin Chen[2†], Tong-He Zhang[1†], Zi-Han Wang[2†]

[*]Corresponding author: niepeng@dbis.nankai.edu.cn

[†]{1911529, 1911396, 1911397, 1911518, 1911486}@mail.nankai.edu.cn

[1]School of Computing, Nankai University, Tianjin, China

[2]School of Cyberspace Security, Nankai University, Tianjin, China

Abstract—With the deepening of informatization and intelligence across society, the construction of smart communities has boomed. However, the construction progress differs from place to place, and the outcome is also not quite satisfying. To promote the development of the smart community, we designed and implemented a grassroots community intelligent service platform. The system is designed to meet the needs of community governance, providing information management, population monitor, and grid management, etc. We also addressed some technical issues such as security, cache, and interface. The grassroots community intelligent service platform not only provides a convenient way to manage community affairs, but also helps reduce the workload of community workers and improve the governance effectiveness. We hope this work could make a positive effect on the development of smart communities.

Keywords - e-government platform; smart community; grassroots governance

1 INTRODUCTION

With the continuous development of information technology, the level of informatization and intelligence has been improved greatly across society. As an important part of smart cities and e-governance, the construction of smart communities has emerged in different regions [1].

The community is the basic unit of governance in the city and the bridge to connect with the people. It is necessary to strengthen the construction of the community governance system, pushing the focus of social governance down to the grassroots level, realizing the benign interaction between government governance, social regulation, and residents' autonomy. [3][4]

In the meantime, there exist some problems like separated data, inconvenient user interface, and complicated old software in the construction of the smart community [2]. We intend to explore

MSIEID 2022, December 09-11, Chongqing, People's Republic of China
Copyright © 2023 EAI
DOI 10.4108/eai.9-12-2022.2327703

the new model of grassroots governance of the "Handheld Cloud Community" by building a grassroots community intelligent service platform, which mainly provides data management capabilities and service access capabilities. It provides functionalities such as management of various information in the community, monitoring of key populations in the community, and access to various services. The information stored in the platform can be analyzed and summarized for abnormal situation warnings and decision-making reference. The user interface is designed to be mobile-first and suitable for most mobile devices, which enables access almost anywhere. The goal of the platform is to open up a two-way channel between the community and residents, build a collaborative co-governance platform, reduce the workload of grid staff, improve community management efficiency, and enhance residents' sense of belonging and happiness.

2 MATERIALS AND METHODS

2.1 System Design

The core of the system is a database application. As shown in Figure 1, the system can be divided into two parts: the business subsystem and the access control subsystem. The business subsystem contains the most functionalities of the platform, but before one can access them, his/her identity must be assured by the access control system.

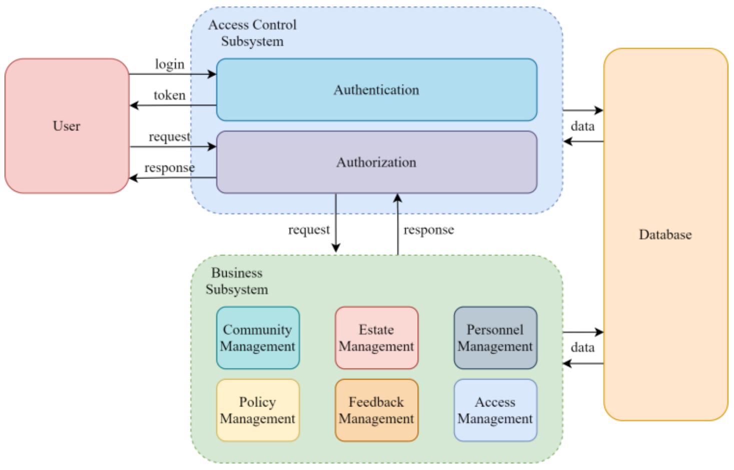

Figure 1. Overview of the system

2.1.1 Business Subsystem

The business subsystem provides the main functionalities of the platform. It can be divided into several modules, including the community management module, estate management module, personnel management module, policy management module, feedback management module, and access management module. Each module has one or several categories.

The community management module is designed to fit the needs of multiple communities. It provides the capability of adding and deleting a community, as well as searching and modifying.

The estate management module is designed to cooperate with grid management, which is a management method that divides the entire community into several grids and assigns each grid with a grid-charger to take care of things. The module provides multi-level management from grids and buildings to units and rooms.

The personnel management module provides management interfaces of staff, grid chargers, and residents. Managers can filter the name list by customizable conditions, which is helpful for community management.

The policy management module provides policy uploading and classification, which can be used to answer residents' inquiries.

The feedback management module provides problem and reply management, which can help deal with some community affairs.

The access management module is designed to work with the access control subsystem. The module provides role-based user permission management, which authorizes a user by associating the user with a specific group of roles. This is more concise and effective when dealing with a number of permissions.

2.1.2 Access Control Subsystem

The access control subsystem provides authentication and authorization. Authentications are done by the traditional combination of username and password. The authorization model is based on RBAC (Role-Based Access Control) model. Considering the demands of our system, we add the department part, which mainly serves as categories.

2.2 System Implementation

The system is decomposed into back-end and front-end. Back-end implements most of the business logic, processing users' requests, checking permissions, and giving responses. Front-end presents the user interface, providing concise and efficient operation to the user. The communications between the back-end and front-end are made possible through RESTful APIs [7][8]. The technical architecture can be described in Figure 2.

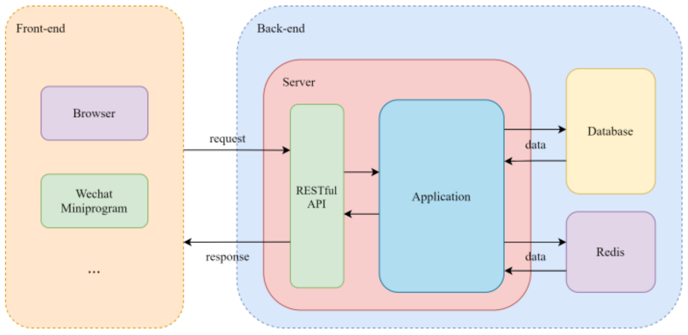

Figure 2. System technical architecture

2.2.1 Back-end framework

The back-end of the system is based on Spring Framework and referred to the back-end implementation of the open-source project RuoYi-Vue [5]. Based on the classic three-layer architecture, the back-end is divided into several modules. [11][12]

2.2.1.1 Admin module

This module mainly contains the Controller Layer. The Controllers receive requests from the front-end and deal with authentication control, authorization control, and data verification. After processing in the lower layer, they return responses in JSON format.

2.2.1.2 Framework module

This module is mainly the integration and configuration with some external frameworks. In the back-end implementation, we combine several open-source frameworks to implement general business such as access control, using the interfaces provided by Apache Shiro to implement authorization control, using the interfaces provided by JWT (JSON Web Token) to implement authentication control, and using relevant schemes in RuoYi-Vue and interfaces provided by Shiro-Redis implementation of the login user cache.

2.2.1.3 System module

This module is mainly related to the Service Layer, DAO (Data Access Object) Layer and Entities of the specific business of our system. The DAOs provide persistent storage of data in a database. The Services handles the data from the Controller layer in more detail and simultaneously uses multiple interfaces of DAOs as a transaction to ensure ACID properties of database access.

2.2.1.4 Common module

This module contains common utilities used in modules at all levels, most of which referred to implementations in RuoYi-Vue, as well as basic entities in the system.

2.2.1.5 Quartz module

This module provides timing tasks independently. The major components of the Quartz module can be summarized as follows:

- A database that stores the configuration detail (i.e., the execution plan) of each timing task.
- A task pool where valid tasks reside.
- An executor that fetches tasks from the task pool and executes the tasks according to the execution plan.
- An array of implementation-agnostic interfaces for the frontend.

To express a task concisely, the Quartz module introduced cron expression. In 1968, K. Thompson introduced the concept of the regular expression that allows for fast pattern matching [10]. Cron expression is a regular-expression-like notion used to describe timing tasks. Users will have access to the system-provided APIs to create cron expressions to schedule quartz jobs that run periodically at a fixed interval.

2.2.2 Stateless Authentication with Token

Traditional authentication methods use sessions and cookies. Considering that almost all users will use mobile devices to access our system, we use token authentication instead. Specifically, we use algorithms and interfaces provided by JWT (JSON Web Token) and add JWT authentication facets to each request-mapped handler in the Controllers using Spring Boot's AOP mechanism [9].

2.2.3 Logged-in User Cache

To improve the concurrency of some high-frequency operations, such as querying basic information and permissions of a logged-in user, we use a memory database rather than a disk database to store frequently-used data.

In the payload field of the JWT, it is common to save the user's ID and use the ID to query the database for subsequent authentication and authorization. This process will join 3-6 tables and require transactional processing. For each request from a logged-in user, the same tables are searched before any business is processed, which is disadvantageous in a high-concurrency application environment. Before going into the details of the cache implementation, it is necessary to explain the relationship between the logged-in user and the real user. A real user can log into the system with different identities on different devices, that is, a real user record in the MySQL database will correspond to multiple logged-in user records in Redis. Based on the interfaces provided by Shiro-Redis, we cache the basic user information (indexed by UUID in the payload of the token, UUID is randomly generated every time a user logs in) and the back-end permissions of each user (indexed by the hash value of user ID and user identity) in Redis. In order to provide multiple device login restrictions for the same user, a set of multiple logged-in users' IDs indexed by the corresponding real user's ID is also stored in the cache. The set can also be used to update the cache of the logged-in user in time when the basic information or permissions of a real user is updated.

2.2.4 Identity Switch

Based on the RBAC model and combined with business demands, we propose a set of solutions from database design to the way of interaction with the front-end.

Given that a user may use more than one identity, there is a need to share the same basic information for their multiple identities while providing permission isolation between these identities [6]. For example, user Zhang is a resident and staff in community A, and also serves as a grid-charger in community B. Zhang does not want to copy the same basic information to all the three systems for staff, grid-chargers, and residents, nor does he want to query any relevant information of community A when serving as a grid-charger of community B. This is the original purpose of using multiple identity switches. Fortunately, guided by the RBAC model, we designed the following data table.

In the Permit table, the add, edit, delete, query, export permissions (hereafter referred to as "back-end permissions") of each table in the database are saved. In the Role table, there are several roles with different permissions that are mapped to Dept's three records. These three records represent three user groups for which our system is geared: staff, grid-chargers, and residents. The Staff, GridCharger, and Resident tables are child tables of the User table and record more information about different identities of the same user.

In practical applications, considering user experience, it is improper to expose all URLs to users regardless of their identity, and only return whether they have permission to access the URL after their clicks. Therefore, the front-end should know which components need to be displayed to the currently logged-in user, and we refer to the relevant permissions as display permissions. The granularity of display permissions is at the front-end component level, while in the back-end it is at the table-operation level. We save corresponding display permissions and community ID in the logged-in user cache and back-end permissions in the permission cache. The front-end and the back-end use separate sets of permissions, with the front-end using display permissions to ensure user identity isolation, and the back-end using community IDs and back-end permissions to ensure user community and identity isolation.

2.2.5 User Interface

The user interface is implemented using uni-app framework, which is a universal mobile interface development framework based on Vue.js. We also used uView UI toolkit to deliver an elegant and uniform visual experience. The user interface is presented in the form of web pages, and due to the mobile nature of uni-app, the page is suitable for most mobile devices. In addition, we reserved an intelligent chat system entry for future extensive use.

3 RESULTS AND DISCUSSION

3.1 Deployment

The system has been experimentally deployed on Alibaba Cloud ECS (Elastic Compute Service). A domain name with an SSL certificate has been resolved to the server, which enables access via browser. The screenshots are shown in Figure 3.

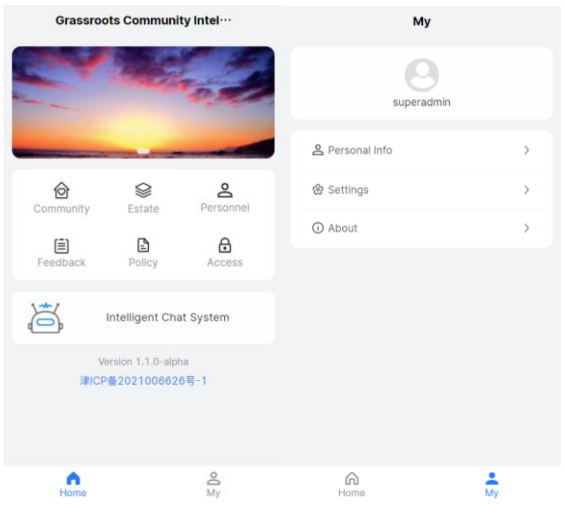

Figure 3. Screenshots of the platform

3.2 Cooperation with community

This work is under the cooperation of Donghuijiayuan Community in Hedong District, Tianjin. The community has many residents with a complex structure and features a large elderly population and a large floating population. At present, the community adopts grid management, and a grid-charger needs to manage 300 households or more. For the elderly, disabled persons, and other key groups, to grasp their health and psychological conditions, grid-chargers are required to communicate with them regularly. The grid-chargers are also responsible for helping to solve various problems of the residents. The community also needs to grasp the movement trajectory and living conditions of the floating population and other types of population. At present, these tasks are carried out by community workers and the workload is huge. It is difficult to guarantee that there will be no mistakes in these complicated tasks manually. The grassroots community intelligent service platform built by us will be helpful to them.

In order to ensure the clarity of the purpose and the accuracy of the results, the project team has held many in-depth discussions with the program. The community provides data and other relevant information, while the project team analyzes demands and develops the platform. Relevant personnel involved in privacy and security issues in the above process have signed relevant agreements.

4 CONCLUSION

In response to the existing problems of the smart community and the needs of the cooperating community, we designed and implemented the grassroots community service platform. In the platform, we established a grid-managed system model. Then we looked into some problems like authentication, cache, and identity isolation. Finally, we encapsulated the system with a mobile-first web interface. Our work could have a positive effect on the development of the smart community.

Acknowledgments. This work is supported by "Hundred Projects" for Undergraduate Innovative Research of Nankai University No.202110055737. Sincere gratitude goes to everyone who has helped us.

REFERENCES

[1] Qi, L., Guo, J. (2019) Development of smart city community service integrated management platform. International Journal of Distributed Sensor Networks,15(6):

[2] Li, S.Q. (2021) Research on the Development Trend of Smart Community. International Journal of Social Science and Education Research,4(7):

[3] Gu, Q.H. (2020) Frame-based Conceptual Model of Smart City's Applications in China. IOP Conference Series: Earth and Environmental Science,615(1):

[4] Liu, C., Wu, H.Y., Wang, J.Y., Wang, M.K. (2020) A Unified Fourth-Order Tensor-Based Smart Community System. Sensors (Basel, Switzerland),20(21):

[5] RuoYi. (2021) RuoYi-Vue. https://gitee.com/y_project/RuoYi-Vue.

[6] Ma, L., Yan, Y.J., Jiang, H.W., Zhou, Y.J. (2020) Analysis of Privilege Escalation Based on Hierarchical RBAC Model. Journal of Physics: Conference Series,1575(1):

[7] Fielding, Thomas, R. (2000) Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine.

[8] Kaur, D.H. (2021) A Performance Comparison of RESTful Applications Implemented in Spring Boot Java and MS.NET Core. Journal of Physics: Conference Series,1933(1):

[9] Sabir, B.E., Youssfi, M., Bouattane, O., Allali, H. (2019) Authentication and load balancing scheme based on JSON Token for Multi-Agent Systems. Procedia Computer Science,148:

[10] Thompson, K. (1968) Programming Techniques: Regular expression search algorithm. Commun. ACM 11(6): 419–422.

[11] Chen, G.H., Xu, J.M. (2020) Design and implementation of efficient Learning platform based on SpringBoot Framework. J. Journal of Electronics and Information Science,6(1):

[12] Quan, Y. (2019) Design and Implementation of E-commerce Platform based on Vue.js and MySQL. In: The 3rd International Conference on Computer Engineering, Information Science & Application Technology (ICCIA 2019). Chongqing, China. pp.460-465.