Information and Knowledge Management (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/infkm.2024.050109
ISSN 2523-5893 Vol. 5 Num. 1

Research on Information Management System of Small, Medium and Micro Enterprises (SMMEs) Based on Software Architecture

Ning Mao[1], Xinmao Wang[1]

[1]*College of Computer Science and Engineering, Jishou University, Zhangjiajie, 427000, China*

Keywords: Requirements Analysis; Security and Confidentiality Design; Enterprise Information Management; Software Architecture

Abstract: This paper takes the System for Small, Medium and Micro Enterprises (hereinafter referred to as SMMEs) as the research object for in-depth research and discussion. The system provides integrated business finance and product inventory management for SMMEs, offering theoretical guidance for the entire business process, including procurement and warehousing, sales billing, account checking, settlement and analysis. The role of system architecture analysis and design in software system and development is introduced with the SMME information management system as an example. This paper will primarily focus on the analysis, design, and implementation of system architectures, as well as the design of system security and confidentiality measures.

1. Introduction

The advent of the Internet has facilitated the continuous integration and reconstruction of technological innovation and commercial elements, which in turn has led to the emergence of new requirements for the development of micro, small and medium-sized enterprises. In order to address the six challenges faced by traditional operations and management, namely low efficiency of manual bookkeeping, difficult account checking and settlement, chaotic inventory management, high operating cost, difficult data coordination, and high cost of customer acquisition, it is necessary to implement a solution that will facilitate the transition from a manual to an automated system. The application of software architecture analysis and design methods in actual combat has demonstrated the value of such methods in guiding the actual development of enterprise management systems.

The utilisation of the scientific architecture requirements analysis method during the system development process can facilitate the establishment of a robust foundation for the overall project development. In the architectural requirements phase, the principal tasks include the acquisition of requirements, the identification of components and the division of system function modules. Firstly, the requirements must be identified and collected in order to ascertain the real needs and pain points of the enterprise. Secondly, a demand analysis must be conducted in order to complete the existing requirements sorting and component identification. Thirdly, the functional modules of the whole system must be further divided in order to provide systematic guidance for the subsequent development work.

The selection of an appropriate software system architecture style can not only reduce the cost of system construction, but also facilitate the enhancement of non-functional system indicators. The architectural style may be classified into five categories: data flow style, call/return style, independent component style, virtual machine style, and warehouse style. It is not uncommon for people to utilise data flow style, batch style, independent component object-oriented style, implicit calling, virtual machine interpreter style and warehouse-style database style[1]. Object-oriented programming abstracts real problems into objects and solves them by calling the properties or methods of each object. The implicit call is initiated by the occurrence of an event, and upon the triggering of said event, the system initiates the automatic calling of all registered processes. The interpreter style enables the interpretation of user instructions through an interpreter, thereby facilitating the implementation of flexible custom scenarios. In the database style, user information is stored in a database in order to ensure the persistence, security and sharing of data. In addition to the aforementioned five architectural styles, cloud native architecture is also a prevalent architectural style in the contemporary era. It is a collection of architectural principles and design modes based on cloud native technology, which aims to maximise the separation of non-business code parts in cloud applications.

In the context of the enterprise, the name, telephone number, bank card number, payment form, receipt and other information data are subject to the privacy of the individual or enterprise. Therefore, it is necessary to prevent individuals with ulterior motives from stealing data. In particular, instances of telecommunications fraud have become increasingly prevalent in recent years. It is therefore evident that the security of the system cannot be guaranteed by local design. The system's security and confidentiality are ensured at the application layer, transmission layer and data layer. At the application layer, double-layer encryption, graphic verification and mobile phone SMS verification are employed to guarantee the security of sensitive information and the system's anti-attack capability. At the transmission layer, HTTPS, timestamps, digital signatures and other technologies are utilised to guarantee data transmission. Finally, at the data layer, data backup and read and write separation are employed to guarantee the disaster recovery capability and performance of the system.

2. System Architecture Requirements Analysis

(1) Demand analysis and refining

Once the demand collection process has been completed, it is necessary to analyse and refine the practical problems and suggestions of multiple parties. This is followed by the construction of the key component of the project demand library, which is based on the "4 + 1" view. Finally, the existing requirements are prioritised. Subsequently, the project requirements library is further divided through the use of data flow diagrams, UML diagrams, and class diagrams, as well as other tools, in order to select reusable components. In the event that no components meet the requisite conditions, they should be temporarily stored in a list of components to be developed in file form. These components should then be developed one by one in the subsequent architecture implementation stage[2]. For instance, in the course of communication with enterprises, it became evident that the issue of return and exchange represents a significant challenge in the business process of enterprises. Firstly, the return and exchange process must involve all levels of enterprise, and there is often a lack of clarity or misunderstanding during the transmission of information. Secondly, the reconciliation and settlement of accounts is challenging. As the number of returns and exchanges reaches a certain threshold, the registration of enterprises becomes challenging, and coordinating with the upstream enterprise data becomes difficult. In order to address this issue, we conducted an investigation into the relevant components within the component library. Following this, we recorded the components that had not been successfully developed in a list of components to be developed, and awaited further

instructions.

(2) Requirements identification and collection

At the outset of a project, it is essential to ascertain the objective and parameters of the project's development. This is followed by the identification of relevant stakeholders in accordance with the project's scope. The enterprise information management system typically comprises end users, enterprise representatives, development teams, test teams, field experts and other personnel representatives. Once the research object has been determined, a preliminary understanding can be gained through the use of a questionnaire survey. Once the materials have been compiled, a targeted research meeting can be held to address the needs of all relevant stakeholders. The demand research meeting serves to ascertain the genuine business needs and enterprise pain points. This information is then further refined and summarized by the development team, thus enabling the collection of accurate and valuable data.

(3) System function and module division

Based on the preceding analysis and summary, the system function can be preliminarily divided into modules. In terms of the upstream enterprise, the system encompasses the functions of supplier, purchase order, processing order, receipt, and return form. With regard to the enterprise itself, the system encompasses product, merchant setting, inventory management, employee management, report, expense payment, business log, online cloud store, enterprise cloud warehouse, and other functions[2]. Finally, in terms of the downstream enterprise, the system encompasses customer, sales order, delivery note, receipt, and other functions. To illustrate, in order to meet the bespoke requirements of customers, upon receipt of the customised product information from customers, the enterprise transmits it to the corresponding suppliers, who then process it to complete the product customisation.

3. System Architecture Style Design and Implementation

In conjunction with the actual project, the enterprise management system employs three principal architectural styles: implicit call, database style and cloud native architecture. The implicit invocation style serves to reduce the coupling of the system, thereby ensuring its good reuse and scalability[3]. The database style is characterised by the retention of user data, screens and the integration of various business data, which supports visualization. The cloud native architecture is associated with a reduction in the development cost and the creation of a business that is more lightweight, agile and highly automated. The following section will provide a detailed overview of the three architectural styles employed in the system.

(1) Database style

The system employs the MySQL database, which offers low cost and good portability characteristics that align with the business's needs. Furthermore, the lake warehouse integration, which is based on an open architecture, enables the storage, extraction, cleansing, transformation, and integration of source data in a timely manner, ultimately leading to the creation of the final data collection. The data lake is capable of accommodating a wide range of data formats, greatly enhancing the flexibility of the system in terms of data storage. The subject-oriented nature of the data warehouse is particularly beneficial for the analysis of system reports, as it enables the integration of data at a higher level of abstraction. This enables users to submit different report requests, such as sales flow, purchase flow, delivery reminder table, and so forth. It eliminates the need for inefficient database searches, providing users with more comprehensive decision support through OLAP for corresponding topic queries and impromptu analysis. To illustrate, if a user requests a sales flow report, the initial display will show the sales flow for the entire year. The user can then view the monthly sales flow. At this juncture, OLAP will perform further drilling of data in the data warehouse,

in accordance with the classification requirements. This will result in the calculation and analysis of data storage in the data application layer, which will then be rendered through the front-end Vue3 to achieve enhanced visualization of the reports.

(2) Cloud-native architecture

The system is an enterprise management system based on Software as a Service (SaaS), where software applications are deployed on cloud computing servers and made available to users via the network. The system does not require users to purchase and maintain software applications or provide support for multi-terminal synchronisation. Instead, users can simply subscribe to the system on an as-needed basis. The implementation of the components is conducted in a sequential manner, commencing with the components to be developed and culminating in the assembly of the components based on the previous component mapping once all components are ready. In the component assembly phase, given that the entire project was developed using a native cloud-based architecture, we employed Docket for containerised component assembly and testing across different modules. The Mesh architecture mode separates the middleware framework from the business process, with the non-business part sunk. This has the effect of greatly improving the efficiency of development. In the testing phase of the system, the final testing phase proceeded at a considerably accelerated pace, exceeding expectations, due to the extensive testing of the previous micro-service architecture and Docket container technology during the development phase. The deployment and testing efforts have been significantly reduced due to the availability of cloud-native automated tools for automated delivery, which are not available in traditional development tools. Concurrently, a three-layer B/S architecture was adopted to divide the entire business application into a performance layer, a business logic layer, and a data access layer[4]. This architectural approach facilitates subsequent maintenance and cross-platform accessibility. Furthermore, during the software design phase, the enterprise, middle, and downstream micro service segmentation was implemented. This approach, although it increased the complexity of the test and operations, ultimately proved beneficial in enhancing the overall system.

(3) Implicit invocation

In the commodity management module of the system, it is often necessary to face frequent updates of a commodity state, which often involves multiple modules of the system. To illustrate, operations such as the addition of products, returns and exchanges, and inventory movements necessitate the invocation of multiple classes, including the inventory management class within the inventory management module and the product management class within the product management module, in order to respond. An implicit invocation is of particular importance in the context of product operations. Upon initiating the product addition operation, the user is afforded the opportunity to input the initial inventory and the upper and lower limits of the warehouse. In the event that a product is associated with upper and lower limits for the warehouse, the system will generate a listening object to monitor it. Should the product in question exceed the monitoring range, the monitoring object will return a message to the inventory management class, which will then label the product in red in order to prompt the user. In addition to the inventory management module, modules for return and exchange, processing lists, delivery notes, and other functions have been implemented using the implicit invocation style. The use of implicit invocation greatly reduces the coupling between different modules of the system, allowing for the completion of operations by simply calling the packaged management class. This avoids the problem of non-synchronisation of data.

4. System Security and Confidentiality Design

(1) Security design of the data layer

As previously stated, the system selects the MySQL database for storage, utilising the integrated

and open architecture to store, extract, clean, and transform the source data in a timely manner, and integrate it into the final data collection. The raw data is initially stored in different nodes of the data lake according to its various types, and then awaits the ETL process of the data warehouse. For example, the system's suppliers, payment orders, and other upstream data for the enterprise are stored on the same node, while inventory, product information, delivery orders, and other enterprise data are stored on another node. The data warehouse can ensure the timely recovery of system data in the event of malicious tampering with the MySQL database or damage to physical devices caused by uncontrollable factors[5]. Furthermore, the distributed architecture of the data lake enables the transmission of data from different nodes simultaneously, thereby enhancing the system's security and availability. Concurrently, we have opted to segregate the applied data for read and write operations in order to preclude external modifications to high-value information within the data warehouse. During the course of the system's daily operation, the user's reading request will be responded to first in the data warehouse. In the event that the requested data is not available in the data warehouse, a query will be issued to the database. Conversely, the user's writing operation can only be carried out in the database. Following this, the data warehouse will update the data independently. This approach not only enhances the system's performance but also guarantees the security and confidentiality of high-value data.

(2) Security design of the transmission layer

The system employs the use of HTTPS for the purpose of encrypted data transmission at the data transmission layer. Firstly, the Hypertext Transfer Protocol (HTTP) encrypts the plain text information. Secondly, the Transport Layer Security (TLS) / Secure Sockets Layer (SSL) protocol establishes the connection between the client and the server for the transmission of information. In order to prevent the third party from launching a data replay attack after obtaining the bag, a time stamp is added to the header of the package. Once the accepted time interval and the sending time interval exceeds 30 seconds, the request is considered invalid. This approach ensures that even a third party attempting to grab the package has only 30s of effective time, which not only protects against data replay attacks but also effectively limits the crawling of system data. Furthermore, given that the system's payment form and receipt list involve financial revenue and expenditure, we utilize digital signature technology to incorporate an MD5 information summary at the backend of the message. This serves to prevent any malicious alterations to user data that may occur after interception and the subsequent crawling of private information, such as phone numbers and bank card numbers. In the case of transmissions that do not involve sensitive information, the more efficient and faster UDP protocol is employed for communication and transmission, particularly in instances where graphic information is involved. This approach has the potential to significantly enhance system performance and reduce resource overhead.

(3) Safety design of the application layer

In the event that the system is confronted with the necessity of user registration and login, the initial encryption is performed using the AES symmetric encryption algorithm. This is then followed by the encryption of the encrypted password with Bcrypt, which is subsequently stored in the database. Upon authentication of the user, the same procedure is followed, with the hash value in the database being compared to verify the user's identity. In addition to identity authentication, the system employs a range of other security measures at the application layer, including graphic verification codes, mobile phone SMS verification, role permission control and other methods[6]. In the system, permissions are opened according to the groups of different roles in order to prevent the occurrence of erroneous operations on the data. For instance, the warehouse administrator is granted access to permissions pertaining to product, inventory management, purchase orders, return orders, and delivery notes, which are pertinent to their role. In contrast, the system administrator is granted access to all permissions, enabling them to make adjustments to the business operations. Upon successful

login, the server returns a token to the client, which is then stored in the Redis cache server as a key-value pair along with the user ID. The expiration time is also set to prevent security issues that may arise from prolonged login sessions.

5. Conclusion

In the context of the information management system of SMEs with numerous roles and intricate functional modules, the system can be divided through the application of analysis and design methods related to software architecture, which significantly enhances the efficiency of software development and the stability of the system. In light of the current development trend, cloud native edge computing is a promising field worthy of further investigation, particularly in light of the rapid advancement of the field of artificial intelligence, which renders the integration of edge computing platforms with artificial intelligence a more viable proposition. In addition, the security and privacy of cloud native edge applications, which are distributed widely and heterogeneous in their resource management, will also face new challenges.

References

[1] Zhang Linshan. Design and implementation of enterprise Engineering Archives Information Management System based on cloud computing [J]. Sme Management and Technology, 2023, (15): 141-143.
[2] Li Chunwei. Research and design of enterprise group archives management system based on microservice architecture [J]. Archives, 2023 (04): 95-97.
[3] Wang Yubin. Research and implementation of Bank OLAP System based on Data Lake [D]. East China Normal University, 2022.
[4] Li Fengwen, Zhou Lanying, Zhang Yifan. Overall analysis and analysis of software testing and software security [J]. Digital technology and applications, 2023, 41(09):225-227.
[5] Zeng Deze, Chen Yuhao, Gu Lin, Li Yuepeng. Cloud-native edge computing: Exploration and Outlook [J]. Journal of the Internet of Things, 2021, 5 (02): 7-17.
[6] Jun W, Nikolay A, Yuan J G, et al. Database system for managing 20,000 20-inch PMTs at JUNO [J]. Nuclear Science and Techniques, 2022, 33(3).