WiseInvest Fund Trading System Design Documentation

Table of Contents

---

1. Introduction

1.1 Purpose

This design documentation provides comprehensive high-level and detailed design guidance for the WiseInvest Fund Trading System, clarifying its architecture, interfaces, and data processing methods.

1.2 Background and References

The WiseInvest Fund Trading System offers convenient fund trading, management, and intelligent services. It adopts a microservices architecture, using Vue3 + TypeScript + Element-Plus for the frontend, and Spring Cloud + Nacos + MySQL + MyBatis-Plus for the backend.

1.3 Assumptions and Constraints

- Project duration: March 2025 to June 2025
- Development devices: 3 Windows PCs and 1 Alibaba Cloud server
- Collaboration tools: GitHub and Feishu

---

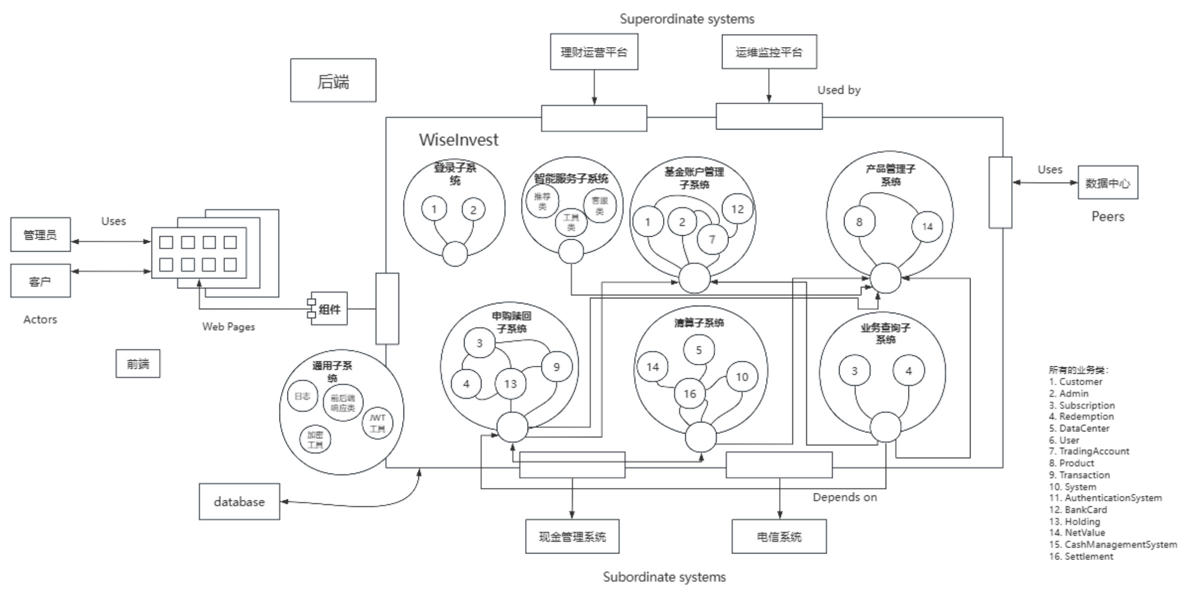
2. System Design

2.1 System Architecture

WiseInvest consists of several subsystems:

- Login Subsystem
- Fund Account Management Subsystem
- Subscription and Redemption Subsystem
- Settlement Subsystem
- Product Management Subsystem
- Business Query Subsystem
- Intelligent Services Subsystem

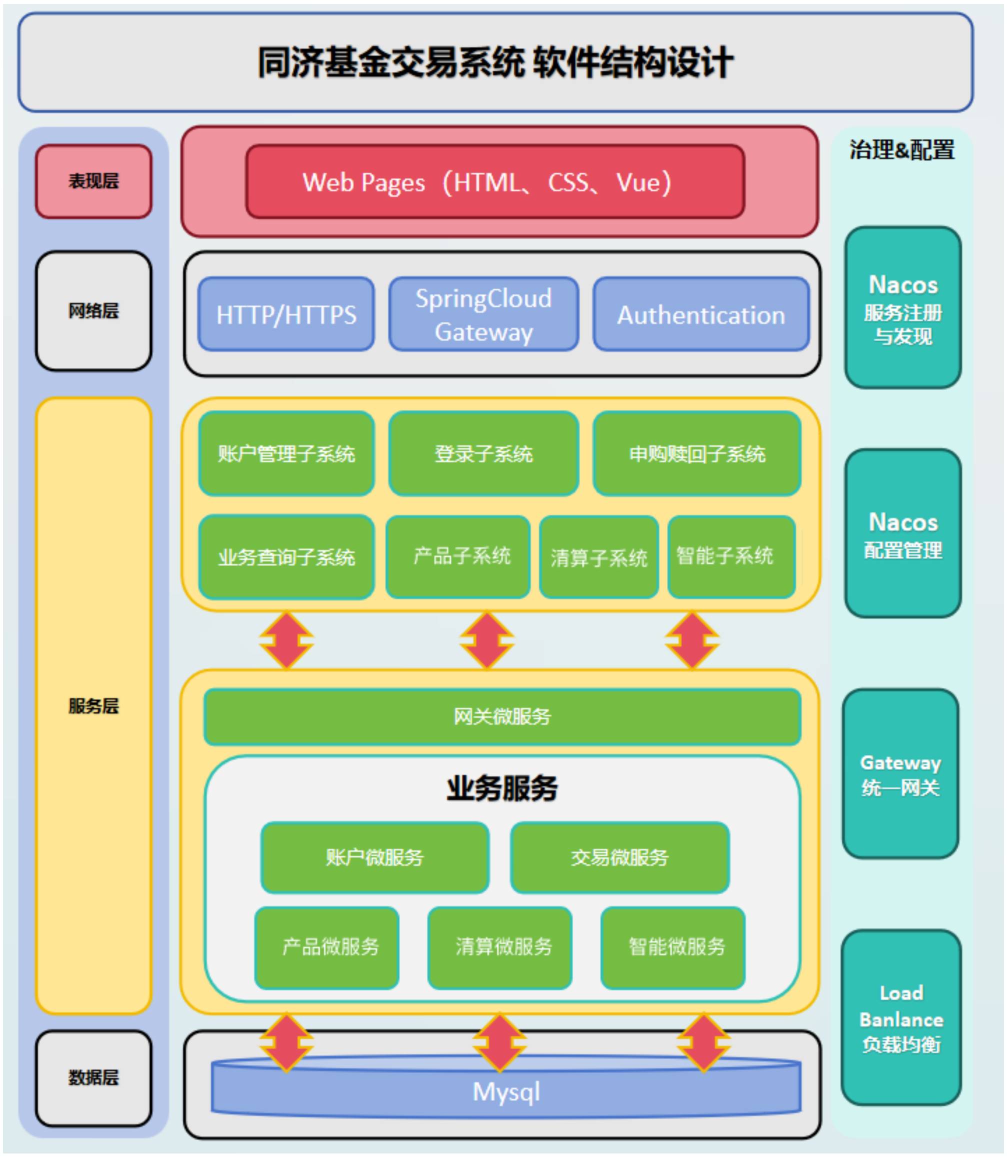
系统体系架构图



2.2 Software Architecture

The system adopts a microservices and layered architecture, including:

- Presentation Layer (HTML, CSS, Vue)
- Network Layer (HTTP/HTTPS, Spring Cloud Gateway)
- Service Layer (Account management, login, trading, product management, settlement, business query, intelligent services)
- Data Layer (MySQL, Nacos service governance)



2.3 Subsystems and Class Design

Fund Account Management Subsystem

- Classes: Admin, Customer, TradingAccount, User, BankcardDTO, CustomerDTO, UpdateInfoDTO, BankcardVO, CustomerVO, Bankcard, BankcardBO
- Mapper: AdminMapper, BankcardMapper, CustomerMapper, TradingAccountMapper
- Service: AdminService, CustomerService, TradingAccountService and their Impl
- Controller: AccountController

Login Subsystem

- Classes: LoginDTO, AdminMapper, CustomerMapper, LoginService, LoginServiceImpl
- Controller: LoginController

Subscription and Redemption Subsystem

- Classes: Holding, Redemption, Subscription, Transaction, HoldingDTO, RedemptionDTO, SubscriptionDTO, RedemptionBO, SubscriptionBO
- Mapper: RedemptionMapper, SubscriptionMapper, HoldingMapper
- Service: TransactionService, TransactionServiceImpl
- Controller: TransactionController

Product Management Subsystem

- Classes: Product, NetValue
- Mapper: ProductMapper, NetValueMapper
- Service: ProductService, ProductServiceImpl
- Controller: ProductController

Settlement Subsystem

- Classes: OurSystem, SystemMapper, SettleService, SettleServiceImpl, NetValue, Bankcard, RedemptionBO, SubscriptionBO
- Controller: SettleController

Business Query Subsystem

- Classes: Transaction, TransactionVO, QueryService, QueryServiceImpl
- Controller: QueryController

Intelligent Services Subsystem

- Recommendation: RecommendEngine, RecommendationDTO, RecommendationResult
- Customer Service: ChatbotService, KnowledgeBase, QueryMessage, ResponseMessage
- Management Tools: KnowledgeEditor, AlgorithmOptimizer
- Controller: SmartServiceController

2.4 Interface Design

- Fund Account Management Interfaces (Create account, update risk assessment, bankcard management)
- Login Interfaces (Verification, password update)
- Product Management Interfaces (Add, edit, list)
- Trading Interfaces (Subscribe/redeem, cancel orders, record query)
- Settlement Interfaces (Daily initialization, market reception, transaction confirmation, data export)
- Intelligent Service Interfaces (Recommendation query, user inquiry)

---

3. External Interface Design

3.1 Data Center Interface

- Interface Type: RESTful API
- Data Format: JSON
- Functions:
  - Receive market data: daily net values of fund products from the data center.
  - Upload data: trading records and settlement results for real-time updates.

3.2 Telecom System Interface

- Interface Type: SOAP Web Service
- Data Format: XML
- Functions:
  - Verification code validation via telecom system
  - Security: SSL/TLS encryption to ensure communication safety

3.3 Cash Management System Interface

- Interface Type: RESTful API
- Data Format: JSON
- Functions:
  - Bankcard binding, subscription deduction, redemption deposit

---

4. Database Design

4.1 Database Overview

Database	Main Tables	Description
Account DB	customer, admin, trading_account, bankcard	User, admin, and account info
Product DB	product, net_value	Fund product info and net values
Settlement DB	our_system	Simulated trade date and system status
Trading DB	subscription, redemption, holding, transaction	Various trading records and holdings
Intelligent DB	recommendation_log, chatbot_query, knowledge_base	Recommendation logs and knowledge base

4.2 Table Structures

customer

Field	Type	Description
fund_account	bigint	Fund account number
name	varchar(45)	Name
phone_number	char(11)	Phone number
risk_level	int	Risk level

product

Field	Type	Description
product_id	int	Product ID
product_name	varchar(45)	Name
product_type	varchar(45)	Type
product_status	int	Status
risk_level	int	Risk level

net_value

Field	Type	Description
product_id	int	Product ID
date	date	Net value date
net_value	double	Net asset value

subscription

Field	Type	Description
transaction_id	bigint	Transaction ID
fund_account	bigint	Fund account
product_id	int	Product ID
subscription_amount	double	Subscription amount
application_time	datetime	Application time

recommendation_log

Field	Type	Description
id	bigint	Log ID
fund_account	bigint	Customer account ID
recommended_product	varchar(45)	Recommended product
algorithm_version	varchar(20)	Algorithm version
create_time	datetime	Recommendation time

chatbot_query

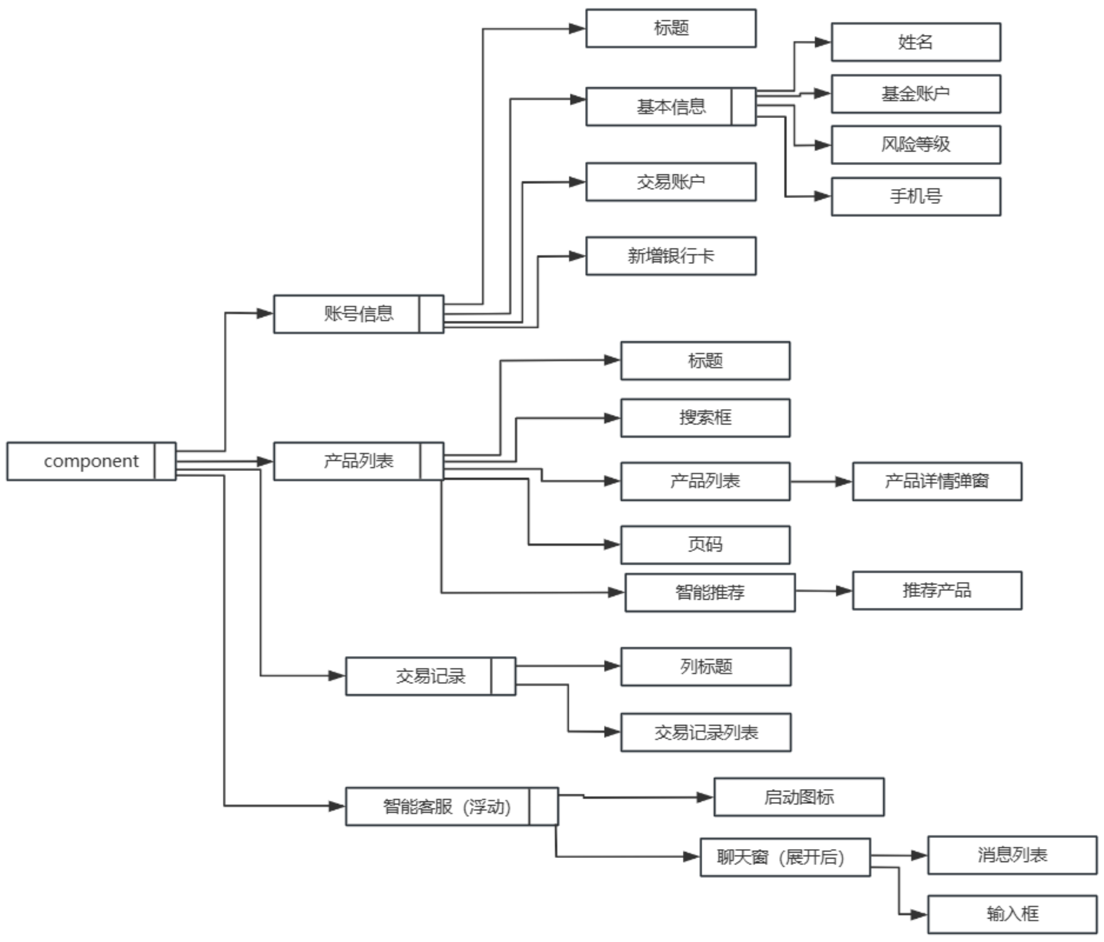
Field	Type	Description
id	bigint	Query ID
fund_account	bigint	Customer account ID
question	text	Customer question
response	text	System response
create_time	datetime	Query time

5. Internal Interface Class Design

Interface Class	Microservice	Method	Description
AccountClient	Trading	getBankcard(id), updateBalance(bankcard)	Access account-related data
ProductClient	Settlement	getNetValue(productId, date), insertNetValue(netValue)	Product info interface
SettleClient	Trading	getSystem()	Get system status
TransactionClient	Settlement	getValidSubscriptions(date), confirmSubscriptionBatch(map)	Transaction request handling
SmartClient	Intelligent Service	recommendProducts(accountId), askQuestion(query)	Personalized recommendations and Q&A

6. User Interface Design

- The system interacts with users via web pages, and content modeling is done using a data tree as shown below.



7. System Error Handling Design

- Error types: login failure, data access failure, network error, transaction failure
- Remedies: automatic retry, local caching, user prompts
- System maintenance: logging, real-time monitoring

8. Maintenance and Extension

- Logging, monitoring checkpoints, dedicated utility modules
- Continuous optimization and updates for intelligent algorithms and the knowledge base