南京师范大学泰州学院

**毕 业 论 文**

|  |  |
| --- | --- |
| 题 目 | 基于微服务架构高性能售卡系统的设计与实现 |
|  |  |

|  |  |
| --- | --- |
| **学生姓名** | 沈海洋 |
| **学 号** | 09230630 |
| **专 业** | 计算机科学与技术 |
| **班 级** | 2306 |
| **指导教师** | 吉晓香 |

2025年 4 月

基于微服务架构高性能售卡系统的设计与实现

摘要

该售卡系统基于微服务技术开发，并且前后端均采用了较为新的开发技术以达到适应现代技术栈和工作流并为用户提供高效、方便的体验。

前端部分使用Vue3框架和Naive-UI组件库编写，其界面美观和响应式布局设计能够同时兼容桌面端、移动端等，而且在多语言和深浅色主题有较好的完成度。用户可通过该平台进行登录注册、包含hCaptcha 人机验证、两步验证、优惠券验证、订阅计划购买、查看订单、设备序列号绑定等操作。此外还内置有即时通信功能，用户还可以提交工单，向管理员或员工提出问题或需求，上手简单，不仅如此，内置的邀请返利功能，能够给予邀请者奖励，有多种支付方式支持，如支付宝、微信和 Apple Pay。管理员端提供丰富的功能模块，包括收入数据分析、权限组管理、公告与通知发布、订阅计划和优惠券管理、用户管理及知识库管理等。仅限管理员可以通过专用界面配置系统的基本功能（如网站名称、主题配色、支付接口、即时通知等），定制应用的功能和主题等。

关键词：微服务架构；Golang；Vue3；响应式设计；实时通信；全球化（i18n）。

**Design and Implementation of a High-Performance Card-Selling System Based on Microservice Architecture**

**Abstract**

The card sales system is developed based on microservice technology, and both the front and back ends use relatively new development technologies to adapt to modern technology stacks and workflows and provide users with an efficient and convenient experience.

**Keywords:** microservice architecture; Golang; Vue3; responsive design; real-time communication; globalization (i18n).

目　录

[摘 要 I](#_Toc692)

Abstract [I](#_Toc14547)I

[第一章 绪论 1](#_Toc194483193)

[1.1 研究的背景 1](#_Toc194483194)

[1.2 研究的目的和意义 1](#_Toc194483195)

[1.3国内外研究现状 1](#_Toc194483196)

[1.4 论文的结构安排 1](#_Toc194483197)

[第二章 关键技术 3](#_Toc194483198)

[2.1 Golang语言 3](#_Toc194483199)

[2.2 Vue3框架 3](#_Toc194483200)

[2.3 TypeScript语言 3](#_Toc194483201)

[2.4 Redis远程字典服务 4](#_Toc194483202)

[2.5 Etcd配置中心 4](#_Toc194483203)

[2.6 RabbitMQ消息队列 4](#_Toc194483204)

[2.7 MySQL数据库 5](#_Toc194483205)

[2.8 微服务与gRPC框架 5](#_Toc194483206)

[2.9 Docker容器与编排工具 5](#_Toc194483207)

[第三章 系统分析 6](#_Toc194483208)

[3.1 可行性分析 6](#_Toc194483209)

[3 6](#_Toc194483210)

[3.2 功能需求分析 6](#_Toc194483211)

[3.3系统操作流程分析 6](#_Toc194483212)

[3.3.1系统设计流程 6](#_Toc194483213)

[第四章 系统设计 7](#_Toc194483214)

[4.1 系统结构设计 7](#_Toc194483215)

[4.1.1用户模块 7](#_Toc194483216)

[4.1.2管理员模块 7](#_Toc194483217)

[4.1.3数据库初始化模块 8](#_Toc194483218)

[4.1.4运维人员运行环境配置模块 8](#_Toc194483219)

[4.1.5总体模块 9](#_Toc194483220)

[4.2系统时序图 10](#_Toc194483221)

[4.2.1用户注册模块 10](#_Toc194483222)

[4.2.2用户登录模块 11](#_Toc194483223)

[4.2.3用户下单模块 11](#_Toc194483224)

[4.2.4订单后处理模块 12](#_Toc194483225)

[4.2.5支付订单生成模块 14](#_Toc194483226)

[4.2.6查询支付结果模块 15](#_Toc194483227)

[4.3 数据库设计 17](#_Toc194483228)

[4.3.1数据库E-R图设计 17](#_Toc194483229)

[第五章 系统的实现 19](#_Toc194483230)

[5.1开发环境介绍 19](#_Toc194483231)

[5.1.1总体配置 20](#_Toc194483232)

[5.2开发软件介绍 20](#_Toc194483233)

[5.2.1 JetBrains Goland 20](#_Toc194483234)

[5.2.2 MySQL 21](#_Toc194483235)

[5.2.3 MySQL Workbench 22](#_Toc194483236)

[5.2.4 OrbStack 22](#_Toc194483237)

[5.3开发环境搭建 23](#_Toc194483238)

[5.3.1 Golang 23](#_Toc194483239)

[5.4创建项目 23](#_Toc194483240)

[5.4.1 创建前端项目 23](#_Toc194483241)

[5.4.2 创建后端项目 24](#_Toc194483242)

[5.5用户模块实现 24](#_Toc194483243)

[5.5.1站点介绍界面 24](#_Toc194483244)

[5.5.3用户注册页面 28](#_Toc194483245)

[5.5.5用户个人中心 32](#_Toc194483246)

[5.6管理员模块实现 37](#_Toc194483247)

[5.6.1管理员后台仪表板 37](#_Toc194483248)

[5.6.2服务端情况查看 38](#_Toc194483249)

[5.6.3系统设置 38](#_Toc194483250)

[5.6.4订阅计划管理 39](#_Toc194483251)

[5.6.6管理员工单处理 41](#_Toc194483252)

[5.7初始化模块实现 42](#_Toc194483253)

[5.8运维人员运行环境配置模块 43](#_Toc194483254)

[5.9杂项 44](#_Toc194483255)

[5.9.1语言全球化（i18n） 44](#_Toc194483256)

[5.9.2 pinia数据持久化 45](#_Toc194483257)

[第六章 系统测试 47](#_Toc194483258)

[6.1 测试方法 47](#_Toc194483259)

[6.2测试用例 48](#_Toc194483260)

[6.2.1 登录模块 48](#_Toc194483261)

[6.2.2 充值模块 48](#_Toc194483262)

[6.3测试总结 49](#_Toc194483263)

[总 结 50](#_Toc194483264)

[参考文献 51](#_Toc194483265)

[致 谢 51](#_Toc194483266)

第一章 绪论

## 1.1 研究的背景

身处2025年付系统、订单处理、商品分发等方面也需要对传统的应用进行升级或重构。

## 1.2 研究的目的和意义

目的：本课题的名称中可以看到主要在于完整的设计与实现这样的一个售卡平台，提供全面的用户和管理功能

## 1.3国内外研究现状

国外:通过

t、Gin等后端框架和可扩展性。当然国内支付方式已经十分成熟，但对接国际支付渠道的研究和实践比较少，也就导致部分应用在海外用户使用下较为困难。

## 1.4 论文的结构安排

第一章

绍来讲述每种技术选型的优势，也会讲到使用Golang语言与Vue

优化部分，我们怎样编写直观的页面。详细描述后端服务的实现、服务之间的通信、数据库操作、

第二章 关键技术

## 2.1 Golang语言

Golang（Go布在2009年，由Google创建，其主要作者是C语言的创造者，所以带来了高效和强大的并发支持。结合了C语言的高性能和Python的易用性，内置的

的虚拟机，误。最后是与Rust相比，Go在并发模型和学习曲线上占有一定优势，更适合快速开发。

## 2.2 Vue3框架

Vue3是Vue2的大版本更新，一种渐进式前端框架，作者是尤雨溪及其团队，Vue3在s也因此基本被它替代了，它更加注重性能和开发体验。其设计理念简洁、高效和易于使用是适合需要快速搭建文档或静态网站开发。VitePress我们将在后续编写文档时会用到。

## 2.3 TypeScript语言

TypeScript其实在2012就已经发布了，但是近几年才变得流行起来，它是由微软开发并

引入了一些新特性，如接口、枚举等，构建复杂的应用程序更为方便。这也是为什么Vue3用Ts进行了重构的原因，TS 与 Vue 的结合能显著提升开发体验与代码质量。

JavaScript和TypeScript的包含关系如下图所示：

图2.1 JavaScript和TypeScript的包含关系

## 2.4 Redis远程字典服务

Redis意为远程字典服务，如果这么说可能比较难以理解，首先它基于的是C/S架构可以使用TCP的代码片段，因此它是构建实时系统的重要组件。

下图展示了Redis在系统中如何降低数据库压力和加快读取缓存：

图2.2 Redis在系统中如何作为缓存使用

## 2.5 Etcd配置中心

首先我们得知道Etc，在早期的Unix文档中就有写到/etc目录是用于存放配置文件的，到对应的服务实例。

下图展示了Etcd配合微服务统一管理运行环境的架构图：

图2.3 Etcd配合微服务统一管理运行环境

## 2.6 RabbitMQ消息队列

RabbitMQ，关键的是其中的MQ，翻译为消息代理和队列，它广泛应用于分布式系统中用于在

ocketMQ，不用于各种领域，像电商、金融、物流等不同行业。

图示

描述已自动生成下图展示了消息队列的工作原理和流程：

图2.4 Etcd配合微服务统一管理运行环境

## 2.7 MySQL数据库

MySQL是一个开源关系型数据库管理系统，开源也是我选它的重要原因之一，最早的时候

资源规划ERP系统、客户关系管理CRM系统等，也都需要处理大量的业务数据。MySQL 的高性能、可靠性和扩展性，能够满足企业级应用对数据管理的严格要求。

## 2.8 微服务与gRPC框架

首先微服务是一种架构风格，不是说精简服务端而已，一个复杂的应用程序被拆分为一组小

务接口，允许客户端直接调用服务器上的方法，就像本地调用了一样，简化了繁琐的远程调用逻辑。

## 2.9 Docker容器与编排工具

Docker同样是开源的，它允许开发者将构建的应用程序及其依赖打包到一个可移植的容器中，然后发

后来被官方收录了，因为它真的很方便在定义和运行多容器的管理工具。一个YAML文件就能配置所有，随后使用一条命令就可以创建并启动，实现对复杂多容器应用的便捷管理。

第三章 系统分析

## 3.1 可行性分析

可行性分析是很重要的，归根结底如果没有通过可行性分析就不可能有这个系统和这个论操作性还能进一步明确系统需求，也能给予较好的情绪价值来增强开发者对开发过程的信心。

3

格遵循法律法规，确保系统合法合规。

会记录下

保障用户支付环节合法合规。另外系统开发中所有代码、设计和内容均为自主原创杜绝盗用，保证无侵权风险。

## 3.2 功能需求分析

功能需求分析是售卡系统开发的核心环节，它直接关系到系统能否满足用户和业务的实际需求。通过深入的市场调研和用户反馈收集，全面梳理了售卡系统不同角色的功能需求。

## 3.3系统操作流程分析

3.3.1系统设计流程

系统设计流程是开发中的关键环节用于在确保设计的严谨性并展现业务处理流程。刚起步

。

通过以上描述的设计流程，我们的系统系统具备了良好的可扩展性和高效性能并能够灵活应对变化，最大地满足用户和企业需求。

第四章 系统设计

## 4.1 系统结构设计

4.1.1用户模块

将它放在第一个的原因是因为用户模块是售卡系统中的核心组成部分，提供了丰富的功能来满好印象和积极情绪价值。通过这些功能，用户模块为用户提供了一个全面、高效且便捷的服务平台。

图表

描述已自动生成本售卡系统的用户功能模块图如图4.1所示：

图4.1学生功能模块图

4.1.2管理员模块

关键的管理控制中心提供常管理员可以迅速采取措施进行修复保障系统的正常运营。总之，管理员模块为系统运营提供了强大的支持，确保平台的高效、安全和稳定。

本售卡系统的管理员功能模块图如图4.2所示：

图片包含 图表

描述已自动生成

图4.2管理员功能模块图

4.1.3数据库初始化模块

这是一便运维人员。

文本

描述已自动生成数据库初始化模块图如图4.3所示：

图4.3数据库初始化模块图

4.1.4运维人员运行环境配置模块

这是一个特殊的模块，独立于主项目之外，运维人员运行环境配置模块提供一个 Web 界面，运维人员可以件，大大减少了出错的可能性。需要注意的是该模块的部署应该部署于内网中，配合网络权限管理使用，禁止将端口映射到公网中。

文本

描述已自动生成运维人员运行环境配置功能模块图如图4.4所示：

图4.4管理员功能模块图

4.1.5总体模块

用例分析结束架构，明确各个模块的职责和关系，从而提高开发效率和项目的可维护性。

本图书馆座位管理系统的系统总体功能模块图如图4.4所示：

图4.5系统总体功能模块图

## 4.2系统时序图

时序图不同于流程图，它用于描述对象之间交互过程，特别是时间顺序上的消息传递。这是流程图更是便于识别潜在问题或优化设计。下面将展示该系统中几个主要的功能的时序图。

4.2.1用户注册模块

用户可进行注册新账户到应用中，注册模块的时序图如图4.6所示：

图4.6用户注册时序图

4.2.2用户登录模块

用户可以使用账户登录进应用，这里做了一些优化，去除了一些不必要的细节，主要有简

用户登录的时序图如图4.7所示：

图4.7 用户登录时序图

4.2.3用户下单模块

用户下单的模块实现过程，用户点击下单后，先生成订单号，订单号生成的算法来自于当前的日期时间戳与用户id等，订单号生成完毕后直接返回给用户订单号，随后将数据打包为json发布到消息队列进行排队处理并附上路由键”order”，由后续的订单处理微服务进行订单处理。

用户下单的时序图如图4.8所示：

图4.8 用户下单时序图

4.2.4订单后处理模块

该模块解析订单 JSON 数据并开始数据库事务，查询用户邮箱、优惠券及订阅计划信息，检查库存后构建订单数据。根据订单时长计算价格和折扣，记录优惠券使用并更新剩余量。提交事务后，将订单存入 Redis，设置 5 分钟有效期，并通过协程监控订单状态，若未完成则更新数据库并恢复优惠券库存。整个流程包含错误处理与事务回滚机制，确保数据一致性。

订单后处理的时序图如图4.9所示：

图4.9 订单后处理模块时序图

4.2.5支付订单生成模块

支付订单生成模块再次检查金额是否有效并请求对应支付平台API生成支付链接，交由用户选择是打开链接还是扫描二维码完成支付，生成的订单会在Redis中存储一份用户后续的支付结果请求。

支付订单生成模块的时序图如图4.10所示：

图 4.10 支付订单生成时序图

4.2.6查询支付结果模块

前端进行轮训请求来查询支付订单是否付款完毕，由于支付平台的返回状态码格式不一致，这里统一成使用不同的http状态码来标识支付结果：

1. 200 WaitUserScanQrCode 预生成订单成功后的等待用户扫码
2. 102 ScannedQrCodeWaitUserPay 用户扫描二维码后生成订单等待支付
3. 408 TradeClosed 未付款交易超时关闭或支付完成后全额退款
4. 200 TradeSuccess 订单成功

查询支付结果模块的时序图如图4.11所示：

图 4.11 查询支付结果模块时序图

## 4.3 数据库设计

4.3.1数据库E-R图设计

这几乎是必须的步骤，在数据库设计中使用实体-关系图对数据之间的关系进行建模，用图形化的

进展的最坏情况来减少沟通中的误差。它不仅在数据库设计阶段至关重要还能够在项目维护过程中帮助开发人员理解现有的数据结构。本系统的E-R图如下所示：

1. 用户属性如下图4.12所示：

图4.12 用户属性图

1. 订单属性如下图4.13所示：

图4.13 订单属性图

1. 有效订单属性如下图4.14所示：

A diagram of a network

Description automatically generated

图4.14 有效订单属性图

1. 激活记录属性如下图4.15所示：

A diagram of a network

Description automatically generated

图4.15 激活记录属性图

E-R模型图如下图4-16所示：

图4.16 E-R模型图

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| deleted\_at | DATETIME | UTC |  | 软删除 |

表4-9 x\_activate\_record表（密钥绑定和激活记录表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度/单位 | 约束 | 备注 |
| id | BIGINT | 64 | PK,NN,AI | 记录Id |
| user\_id | BIGINT | 64 | FK | 用户Id |
| email | VARCHAR | 256 | FK | 用户邮箱 |

图 4.18 分布式架构的逻辑结构图

第五章 系统的实现

## 5.1开发环境介绍

本节介绍项目在开发期间所使用的设备、操作系统、软件及运行环境。需要注意的是，开发环境与生产环境存在一定差异，生产环境需根据具体的设备配置、网络条件等进行适配和调整。虽然本项目的核心运行环境（如 Nginx、Node.js、Golang）能够在多种操作系统（Linux、FreeBSD、Windows）下运行，但为了确保性能和稳定性，各微服务的容器化运行推荐在 Linux 环境中使用 Docker 技术。在其他系统下，可以根据需求选择合适的方案，如在 FreeBSD 系统中使用 Jails，或在 Windows 系统中利用 Windows 容器来实现微服务运行。

如需要根据具体的需求在本项目上进行改动或优化，遵循下面的配置可以最大程度地贴近生产环境，以减少部署和运行过程中的潜在问题。

5.1.1总体配置

1) 硬件配置

CPU：Apple M2 Pro

内存：32GB LPDDR5

硬盘：1TB PCIE

操作系统：macOS Sequoia 15.1.1 (Darwin24.1.0 arm64)

基本的输入输出

数据操作：MySqlWorkbench、DBeaver、RedisInsight、EtcdWorkbench

文档撰写：typora

插件：CodeGlance、RainbowBracket、gRPC、ProtoclBuffers、OrmHelper

## 5.2开发软件介绍

5.2.1 JetBrains Goland

JetBrains GoLand是一款集成开发环境。但它不仅能够编写Go语言，还能编写Java、Nodejs、Vue、JavaScrip后轻松驾驭。

丰富的代码分析工具能让我们及时发现代码中的潜在错误、性能问题以及代码规范的违反情况并给出详细且实]用的改进建议。它对Golang语言的各种特性和标准库有着出色像Idea一样还支持多种插件扩展，可根据自身需求进一步定制开发环境来满足多样化的开发任务需求。

图5.1 JetBrains Goland软件界面

5.2.2 MySQL

还可生成对应的SQL脚本来便于在不同环境中部署数据库可简化了运维人员的工作。我就是使用它来手动编写测试的SQL语句。

图5.2 MySQL Workbench软件界面

5.2.3 MySQL Workbench

Redis Insight是Redis可视化管理工具。界面直观，可便捷连本地或远程 Redis 服务器，清晰展示多种数据结构并支持操作。能监控内存、命令频率等关键指标。

图5.3 Redis Insight软件界面

5.2.4 OrbStack

OrbStack和Docker Desktop一样是用于在不同的系统上进行容器开发，但是它专为OSX的Linux发行版或容器，并能调用Rosetta2来运行x86\_64架构的容器和虚拟机。

图5.4 OrbStack软件界面

## 5.3开发环境搭建

5.3.1 Golang

本项目的后端程序使用Golang作为主要语言，主要有两种安装方式，使用包管

kanna@kannadeMacBook-Pro ~> npm -v # 应该打印 `10.9.0`

## 5.4创建项目

5.4.1 创建前端项目

在

}

启动成功后，访问终端中显示的本地开发服务器地址，就可以看到初始化后的 Vue 3 项目的基本界面。可以开始在这个基础上开发应用，比如创建新的组件、定义路由、管理状态等。如下图5.6所示：

图5.6 成功初始化前端项目

5.4.2 创建后端项目

为每一个微服务创建一个单独的文件夹并初始化Go模块：

# 每一个微服务都被放置在backend中

kanna@kannadeM

通过上述步骤后，Dockerfile 负责容器化，确保环境一致性，提高可移植性；Makefile 负责自动化任务，如构建、测试、部署等。两者结合使用，可以简化开发流程，加速 CI/CD，并提升团队协作效率。

## 5.5用户模块实现

5.5.1站点介绍界面

该页面是访问网站的默认页面，旨在向用户展示企业的基本信息和核心优势，增强品牌形象，提升用户信任感。总体使用使用F型排版，页面顶部的左侧展示品牌 LOGO、企业名称，顶部的右侧给予三个按钮，分别为语言切换、登录、注册账号，目前支持四种语言切换，分别为中文简体、中文繁体、日文和英文。中间左侧部分显示站点名称和站点的描述，右侧使用简洁的图形作为背景，引导用户的视觉向左侧调整，包含简短的企业口号，以增强品牌识别度。

图5.7 网站首页第1屏

向下滚动来到第二屏幕展示为何选择我们，将列出企业的核心优势，优点使用2\*2布局的栅格来展示，并使用响应式布局，同时适配桌面端、平板端、移动端。

图5.8 网站首页第2屏

首面尾部主要分为两个部分，第一个是快速链接，在这里可以展示企业的官方账号链接，用户点击对应社交账号的按钮后将开启一个新的标签页来跳转到官方账号页面。第二个部分为版权声明和备案，关于网站备案号，如果在中国大陆提供服务，那么就需要受到中国监管，因此需要进行ICP备案和遵守《网络安全法》中的内容审查和合规要

nv/welcome来获取页面配置，在后端部分含有两个缓存部分，一个来自于内存部分，使用一个map来缓存配置项目，该缓存的过期时间TTL设置为10分钟，到期后将从Redis服务器中重新拉取最新配置存储进缓存，Redis中使用SET数据类型存储配置，确保数据一致性。

// 后端执行查询和刷新缓存

if s.welcomePageConfigCache != nil

&& time.Since(s.welcomePageConfigCache.LastSaved) < s.welcomePageConfigCacheTTL {   
 // 直接返回缓存中的数据 并跳过后面的查询  
}

// ……  
for \_, key := range configKeys { // configKeys 是一个数组存储所需要的字段名  
 value, err := readSettingFromRedisCache("welcome", key)  
 // 省略处理错误  
 var parsedValue interface{}  
 if err := json.Unmarshal(value, &parsedValue); err != nil {  
 // 省略处理错误  
 continue

}  
 config[key] = parsedValue // 全部执行正确后写入缓存  
}

直观，同样也支持响应式，方便手机端操作。下图5.10展示了登录界面的布局。

图5.10 用户登录界面布局

在讲解详细代码之前，还需要了解一个认证技术JWT，由于http是无状态协议，也就是每个请求都是独立的，服务器不会自动记住客户端的任何状态信息。

JWT是一种用于安全传输信息的令牌，通常用于身份验证和授权。它由头部、负载和 签名组成，采用 Base64 编码，并通过 HMAC、RSA 或 ECDSA 进行签名，确保数据完整性和防篡改性。Token的内容如下图5.11所示：

图5.11 JWT Token内容

每次请求受保护的资源时，客户端在HTTP头部（Authorization: Bearer <token>）携带JWT，服务器接收到请求后，解析并验证JWT（检查签名、解析负载、校验是否过期）。如果验证通过，则执行请求，否则返回401未授权。

下图5.12展示了携带JWT Token一次请求的过程：

图5.12 携带Token进行请求

下面的代码片段展示了生成JWT的过程，需要注意的是，jwt-go库出现过安全问题，并且该库出现了安全问题，可以通过漏洞绕过JWT检查，已经在2022年5月被归档并

化(thisUser)

}

5.5.3用户注册页面

用户可以在这里进行注册新账户，要求用户至少给出邮箱地址和密码即可完成注册，要求密码需要满足为大写字母、小写字母、数字、符号其中的三种及以上，该页面中的邮箱验证码和hCaptcha人机验证都是可以自定义是否启用的，同样的，在管理员页面的系统设置>安全选单中，下图5.13为注册页面的布局：

图5.13 用户注册界面布局

如果管理员管理了设置中的用户注册，将不会渲染表单，并给出提示来告知用户。管理注册后的注册页面如下图5.14所示：

图5.14 停止新用户注册界面布局

},

};

邮箱格式验证函数 validateEmail，用于确保用户输入的邮箱地址符合标准格式，并针对 Gmail 进行额外限制。首先，它使用正则表达式检查邮箱格式是否正确，然后，它解析邮箱

式是否符合正则

const [localPart, domain] = email.split("@");

const isGmail = domain.toLowerCase() === "gmail.com";

const isGooglemail = domain.toLowerCase() === "googlemail.com";

// 如果是 Gmail 或 Googlemail，且开启了 Gmail 限制

if ((isGmail || isGooglemail)出登录。

下图5.15为用户首页的布局：

图5.15 用户首页布局

其中设计有可以自动适配操作系统的深浅色功能，使得用户即使在夜晚也可以舒适的访问站点，目前该功能已经在各类操作系统如Windows10及以上、macOS 10.12及以上，装有Wayland和X11窗口管理器的Linux，并且在基于chromium、Gecko、WebKit内核的浏览器上运行良好，深色页面如下所示：

图5.16 自动适配系统级深色

其中考虑到会有不少的用户会使用移动端设备来访问网站，包括智能手机、平板、电子书等等，因此所有的页面均使用响应式设计模式，下图5.17演示该网页在移动端设备上的适配情况，演示设备为iPhone 14系列。

图5.17 响应式设计适配移动端

代码部分将展示识别深色模式和切换、适配不同屏幕尺寸的设备、以及侧边栏的

该API的适配情况可最低可用的浏览器版本如下表5-1所示：

表5-1 x\_API支持情况

|  |  |  |  |
| --- | --- | --- | --- |
| 平台 | 浏览器/内核 | 最低版本 | 是否可用 |
| 桌面端 | Chrome | 76 | 可用 |
| Edge | 79 | 可用 |
| Safari | 12.1 | 可用 |
| Opera | 62 | 可用 |
| 移动端 | Chrome Android | 76 | 可用 |
| Chrome IOS | 67 | 可用 |
| Opera Android | 54 | 可用 |
| Safari on IOS | 13 | 可用 |
| Samsung Internet | 14.2 | 可用 |
| WebView Android | 76 | 可用 |
| WebView IOS | 13 | 可用 |

侧边导航栏如果在移动端则会为整个屏幕的横向宽度的80%，桌面端时为固定320px，这一部分使用JS配合原生的事件监听器完成，如果resize属性变化则调用

5.5.5用户个人中心

在用户中心，用户可以执行以下的操作，设置自己的基础信息，如头像、用户名，监视账户余额并可以点击跳转到充值页面，修改自己的密码，配置2fa两步验证，以及注销账户。这一部分将讲述的有图像的压缩、上传和存储，两步验证的呈现逻辑，用户注销时的软删除逻辑

分已经讲过两步验证的原理、二维码链接的生成和密码验证，此处不再赘述。扫描该二维码需要手机上安装有对应的软件。

图5.19 设置2fa两步验证

生成的二维码链接格式如下所示：

# TOTP

统设置>支付和返利中自行配置，随后在选择充值金额区域，提供了快速选择的固定金额选项，包括 100 CNY、200 CNY、500 CNY、1000 CNY、2000 CNY ，还设有其他金额的自定义输入框，最大金额限制为 10,000,000 CNY。右侧板块给出三种支付方式，选择后，系统会自动根据优惠信息计算最后需要支付的金额，如有优惠信息将显示在栏目中，也含有充值后余额展示。

用户充值页面的布局如下图5.20所示：

图5.20 用户充值界面设计

下面的代码部分将讲述到页面执行逻辑，包括跳转、支付二维码的展示、支付成功后的彩条飘舞动效设计等，首先讲述支付部分，关于支付的响应状态码含义，在4.2.6查询支付结果模块部分讲述过，此处不再赘述，下面将以支付宝的从生成订单到查询支付结果为例讲述整个支付过程。

首先来到支付宝开放平台的控制台申请一个网页/移动应用，下面给出订单码支付的介绍和接入文档以及开放平台的链接：

全部完成后的提示和得到的所有文件应如下图5.21和图5.22所示：

图5.21 配置好接口加密

图形用户界面, 应用程序, 网站

描述已自动生成

图5.22 获取到的所有证书和密钥文件

随后来到控制台的可调用产品部分，进入alipay.trade.precreate部分的文档，收银员通过收银台或商户后台调用支付宝接口，生成二维码后，展示给用户，由用户扫描二维码完成订单支付。下面给出文档的地址。

// alipay.trade.precreate(统一收单线下交易预创建)

https://opendocs.alipay.com/open/8ad49e4a\_alipay.trade.precreate?pathHash=d18bff53&scene=2ae8516856f24a5592194d237f3f235d

文档中主要有两部分，一是请求参数，二是响应参数，其中请求参数重又包含公共请求参数和业务请求参数，公共请求参数中主要包含的是商铺签约的一些配置来识

格式化金额付二维码和订单号，供前端展示或使用。整个流程包括 参数校验 → 金额计算 → 调用支付接口 → 结果处理，确保订单创建的准确性和安全性。

}

以上步骤都没有问题后，将提示用户扫描二维码，用户使用支付宝App扫描后随即转到选择支付方式，界面设计如下图5.23所示：

图5.23 等待用户扫描二维码

接下来就是在前端启用轮训请求后端接口来查询该订单的支付结果，是成功、失败、超时、还是用户取消了付款：

func DoAlipay

、描述、优先级等信息后提交。工单会出现在历史工单列表中，等待客服处理。每个工单会有不同的状态为未关闭和已关闭，工单的优先级可分为 高（红色）、中（黄色）、低（绿色），用于标识问题的紧急程度。下图5.25和图5.26展示了工单界面和新建工单的界面。

A screenshot of a computer

AI-generated content may be incorrect.

图5.25 历史工单列表

A screenshot of a computer

AI-generated content may be incorrect.

图5.26 用户提交新的工单

在介绍核心代码之前还需要了解一个新的通讯技术，WebSocket是一种全双工的网络通信协议，它允许客户端和服务器之间建立持久连接，并能在连接建立后随时相互发送数据，而无需像HTTP那样每次都要重新建立连接。WebSocket 适用于实时性要求高的应用场景，如聊天系统、在线游戏、股票行情推送、实时通知等。

当用户或管理员点击查看工单后，将会开启一个新的浮动聊天窗口窗口，它的宽高为480\*640px，

);

pipe := dao.Rdb.Pipeline() // 使用pipeline批量执行写入

for \_, msg := range chatHistory {

chatJson, \_ := json.Marshal(msg)

pipe.RPush(ctx, ch

Id).Order("`sent\_at` ASC").Find(&chatHistory)

s形式

}, nil // 此处的nil为没有rpc的错误

}

该章节讲述了工单部分的核心代码，是同时运用了消息队列（RabbitMQ）、缓存服务（Redis）和数据库（MySQL）实现聊天系统的最佳实践。

## 5.6管理员模块实现

5.6.1管理员后台仪表板

管理员看板是整个系统的核心管理界面，专为不同级别的管理员提供系统运行状态、业务数据和操作入口。根据权限的不同，系统管理员和普通管理员在看板上看到的信息和可执行的操作有所区别，系统使用 Apache ECharts 进行数据可视化展示，API 访问次数采用折线图展现趋势，每日收入以柱状图直观对比。图表主题色与系统主题同步，并支持深浅色模式自适应切换。数值部分采用动画数字效果，使数据从 0 平滑增长至目标值，增强动态视觉体验。如下图5.28所示：

A screenshot of a computer

AI-generated content may be incorrect.

图5.28 管理员仪表板页面

5.6.2服务端情况查看

服务端信息页面展示客户端到服务器的实时延迟，并统计网关服务的 200、404、401、500 状态码响应次数。下方表格记录 API 访问详情，包括接口名称、来源 IP 和状态码，并支持导出为 CSV 文件 便于分析。如下图5.29所示：

A screenshot of a computer

AI-generated content may be incorrect.

图5.29 服务端信息查看

5.6.3系统设置

系统设置中系统管理员可以进行系统中的设置，每一个栏目中有不同的配置项，该页面只对于系统管理员有效，其他用户强制访问将会被导航到403页面。

A screenshot of a computer

AI-generated content may be incorrect.

图5.30 系统设置页面

此处使用一个统一的函数，传入键和需要设置的值来请求后端服务器来更新：

const saveFiled = async (k: string, v: any) => { // 异步函数保存配置

let updateResponse = await settingStore.saveOption('invite', k, v)

if (data && up

if err := dao.Rdb.HSet(ctx, redisKey, key, valueStr).Err(); err != nil { // 处理错误 }

return nil

}

5.6.4订阅计划管理

在订阅管理中，管理员或员工可以对订阅计划进行所有管理：

A screenshot of a computer

AI-generated content may be incorrect.

图5.31 订阅计划管理页面

这是一个典型的crud，即增删改查的最佳实践，配合RestAPI风格实现一个接口上使用GET、P

、博客、技术文档等场景。Markdown的优点包括易于学习、语法简洁、支持多种平台，且便于进行版本控制和协作。相比复杂的HTML，Markdown让文本更易读、编辑和维护，适合快速创建清晰的文档。

A screenshot of a computer

AI-generated content may be incorrect.

图5.32 说明文档撰写页面

这里使用的Markdown编辑器是md-editor-v3，它使用jsx和typescript语法开发，支持切换主题、prettier美化文本等。下面介绍官网地址和使用方式：

**官网地址：** <https://imzbf.github.io/md-editor-v3/>

**Markdown语法教程：**<https://imzbf.github.io/md-editor-v3/zh-CN/syntax>

kanna@kannadeMacBook-Pro ~/D/c/nxc-frontend> npm install md-editor-v3 # 安装依赖

下面给出一个简单的示例，含有基本的编辑功能：

<template>

<MdEditor **v-model**="text" />

</template>

<script **setup**>

import { ref } from 'vue';

import { MdEditor } from 'md-editor-v3';

import 'md-editor-v3/lib/style.css';

const text = ref('Hello Editor!');

</script>

5.6.6管理员工单处理

文档编写页面支持MD语法，该页面的布局和使用与用户界面相似，但这里讲显示所有的用户工单信息，该页面的权限为管理员和员工都可以访问，方便快速的处理用户工单，头部模式展开的是待处理和紧急的工单了，默认折叠了已经完成的工单，在勾选所有后可展示。

关于消息的处理部分与用户部分相同，包括消息队列、Redis缓存和数据库，唯一的区别是消息的sender为”admin”来区分消息发送者，两者共用一个聊天框组件，下面是工单处理的页面设计：

图5.33 管理员工单处理页面

## 5.7初始化模块实现

在该后端系统中，初始化模块的执行流程主要包括配置数据库连接、检查系统环境、构建可执

件，配置数据库连接相关信息，包括数据库的账户、密码、端口等信息。此步骤确保系统能够正确连接并访问目标数据库。该文件是一个yaml配置文件，可以使用vim、nano等编辑器修改

4 架构的可执行文件。构建完成后，可执行文件exec被生成到 ./build 目录。

# 为macOS arm64架构编译可执行文件

build-darwin-arm64: $(BUILD\_DIR) mkdir -p $(BUILD\_DIR)/darwin-arm64 CGO\_ENABLED=0 \

GOOS=darwin GOARCH=arm64 $(GOBUILD) -o $(BUILD\_DIR)/darwin-arm64/$(BINARY\_NAME) $(MAIN\_FILE)

生成的可执行文件被执行，系统提示用户在哪个数据库中创建表。随后，程序会读取一个预定

统中需要安装git\_shell来获取bash的shell脚本支持，第一个参数为操作系统类型，第二个为操作系统的架构：

kanna@kannadeMacBook-Pro ~/D/c/nxc-frontend> chmod +x install.sh # 赋予可执行权限

kanna@kannadeMacBook-Pro ~/D/c/nxc-frontend> ./install Darwin arm64 # 在arm64下的macOS运行

文本, 信件

描述已自动生成

图5.34 运行后端初始化

## 5.8运维人员运行环境配置模块

由于项目采用了微服务架构，并且每一个微服务都需要对Mysql数据库，Redis等基础服务进行访问，那么就需要要一个连接信息，如连接到数据库所使用的主机、端口、用户名、密

化为一种通用的数据格式，如XML、JSON、YAML等，这里选择使用XML，下面是转换部分的代码，这里以存储消息队列的部分为例，其他部分与该部分相似：

下面是调用的函数，直接存储前面格式化好的XML数据：

图5.35 运维人员运行环境配置模块前端页面

## 5.9杂

5.9.1语言全球化（i18n）

又称国际化，i18n是internationalization的缩写，用于在系统中，能够实现支持不同语言和地区的需求，在本项目中使用vue-i18n这个一个库来实现，所有的语言的提示都将放置在一个单独的文件中，在项目运行时进行调用不同的语言文件，下面将讲述其安装和使用方法：

**官网地址：** <https://vue-i18n.intlify.dev/>

kanna@kannadeMacBook-Pro ~/D/c/nxc-frontend> npm install vue-i18n@10 # 安装依赖

安装完成后在项目的main中引用该插件即可，随后在src中单独新建一个文件夹来存放所

// locale是获取当前语言名称 非必须

console.log(t('commonHeader.userData')) // 将打印出"用戶資料"

下图为切换为英语模式后的页面，切换后无需刷新网页：

图5.36 切换为英文后的示例页面

5.9.2 pinia数据持久化

Pinia中实现数据持久化，使状态在浏览器刷新后仍然保留，这里使用pinia-plugin-persistedstate插件。这个插件将Pinia状态持久化到localStorage中，使得在页面刷新后状态能够恢复。

**项目地址：**<https://github.com/prazdevs/pinia-plugin-persistedstate>

kanna@kannadeMacBook-Pro ~/D/c/nxc-frontend> npm i pinia-plugin-persistedstate #安装依赖

在每一个store 中，可以使用 persistedState 配置来指定需要持久化的状态。你可以控制是否将整个状态持久化，或者只持久化部分状态，如下所示：

persist: {

enabled: true, // 启用持久化

strategies: [

{

key: 'user', // 存储的 key，默认会使用 store 的名称

storage: localStorage, // 数据持久化到 localStorage，或者你也可以使用 sessionStorage

paths: ['username', 'preferences'], // 指定持久化的字段

},

],

}

配置成功后在Application中即可看到数据：

图5.37 pinia数据持久化

第六章 系统测试

系统测试是系统完成的过程中不可或缺的步骤，它贯穿于编程过程中，是检验每个单元、每个模块以及整个系统是否合格的方法。

## 6.1 测试方法

软件测试方法的种类繁多，一般来说只要能够有效检测软件运行状况的方法都可以称为软件测试方法。然而在系统测试中严谨来有三种广泛应用且被公认为最可靠的软件测试方法。接下来将简要介绍这三种方法：

1、单元测试

单元是一个系统中的最小单元，一个完整的复杂的软件由大量精细的单元组成。因此，单元测试的过程是极其细致入微的，也是系统运行后期几乎没有或没有BUG的重要前提。通过逐一细致地检测系统中的每个模块，每一个模块都经过精确的测试验证，从而保证其良好运行。在功能实现的过程中，只要每一个模块都经过详尽的测试确保正常运行，通常所期望的功能实现就能顺利进行，这样就节省了大量寻找BUG的时间，为系统后期的测试节约了宝贵的时间。

2、集成测试

集成测试在软件开发进程中属于关键的测试部分，其主要作用是校验不同模块之间的集成是否顺畅。在集成测试中使用各种测试方法来检查系统的正确性和稳定性。集成测试的作用主要体现在发现和修复模块之间的错误和BUG。通过集成测试可以及早发现模块间的接口问题、模块之间的兼容性问题和数据传输问题从而降低系统开发过程中出现问题的风险。

3、确认测试

确认测试是在集成测试完成后开展的一种普遍应用的测试形式，主要用于进一步明确系统是否存在缺陷。作为一种有效性测试方式来确认测试旨在通过多种运行场景和特定因素的测试来验证系统的功能和性能与开发需求的一致性。通过确认测试可以确保系统已经达到可投入实际使用的标准并证明系统的功能和性能符合需求。

## 6.2测试用例

由于该项目模块数量较多，此处只列出几个比较重要的测试用例，保证该项目的核心功能不会出现异常错误，即用户的下单、支付、激活、工单等功能，其余的如邀请注册、文档阅览将在后续完善中的使用文档中提到。

6.2.1 登录模块

表6.1 登录测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试模块 | 登录模块 | 测试概述 | 测试用户使用登录功能的情况 |
| 测试环境 | 微服务架构高性能售卡系统 | | |
| 测试目的 | 验证各个注册用户均能登录到改系统，不会出现越级现象。 | | |
| 测试步骤 | 在登录框中，对应输入先前注册成功的账号以及密码以及两步验证码  检查输入的内容无误，点击登录的按钮 | | |
| 测试要求 | 分别采用不同权限的用户账号登录系统 | | |
| 期望结果 | 不同权限的用户均成功登录，进入到指定账户的系统界面中 | | |
| 实际结果 | 完成 | | |
| 出现问题 | 第三方oauth2.0登录 | | |
| 修改方案 | 后续由社区完善 | | |
| 开发者 | 沈海洋 | 开发日期 | 2024年7月10日 |
| 测试者 | 沈海洋 | 测试日期 | 2025年3月6日 |

6.2.2 充值模块

表6.2 充值测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试模块 | 充值模块 | 测试概述 | 测试用户是否能顺利完成支付 |
| 测试环境 | 微服务架构高性能售卡系统 | | |
| 测试目的 | 验证是否能正确对接支付网关并完成完整的支付流程 | | |
| 测试步骤 | 使用普通用户账户进入充值页面  点击或输入充值金额后点击确认按钮以获取支付链接和二维码  使用支付平台如支付宝等进行支付，包括桌面端二维码和移动端链接 | | |
| 测试要求 | 不管是桌面端还是移动端都可以成功完成支付 | | |
| 测试者 | 沈海洋 | 测试日期 | 2025年3月6日 |

## 6.3测试总结

在本次模拟测试，系统在界面加载和交互过程中没有出现明显的卡顿或延迟，充分体现了其高效的性能和用户友好的设计。整体而言系统在高并发负载下稳定运行同时提供了清晰、整洁的界面，提升了用户体验，并满足了实际业务需求。

总 结

经过近

求。

在系统开发过程中，深刻认识到了微服务架构和高并发系统设计的挑战与魅力。首先系统的稳定性和高效性离不开对架构的深度理解与技术选型。其次注重代码质量和模块化设计，以及细节的优化，是实现系统高性能的关键。在未来的开发中将继续提升系统的功能和性能，尤其是在高并发处理和界面设计方面来争取为用户提供更好的体验。

参考文献

1. 李俊君,刘永亮,王维,等.Golang软件包API演进情况研究[J].电脑知识与技术,2024,20(11):39-41+52.DOI:10.14004/j.cnki.ckt.2024.0576.
2. 左向山,曾文果.利用Grpc构建多重安全性微服务研究与应用[J].电子技术与软件工程,2023,(04):5-8.
3. Clément J .gRPC Go for Professionals:Implement, test, and deploy production-grade microservices[M].Packt Publishing Limited:2023-07-14.DOI:10.0000/9781837634095.
4. 黄磊.基于微服务的运维平台设计与异常检测研究[D].电子科技大学,2021.DOI:10.27005/d.cnki.gdzku.2021.002797
5. 学,2024.DOI:10.27005/d.cnki.gdzku.2024.005281.
6. 向啟苗.基于Golang的文件存储优化程序实现[J].电脑编程技巧与维护,2023,(05):77-79.DOI:10.16184/j.cnki.comprg.2023.05.041.
7. 王明松,于营,秦永佩.基于协程(goroutine)的并发编程模型及应用[J].电子质量,2022,(06):95-98.
8. 仵春林.基于容器和微服务的自适应流媒体传输系统设计与优化[D].电子科技大学,2024.DOI:10.27005/d.cnki.gdzku.2024.001771.
9. 李方方,周亚凤,王校建,等.高并发条件下消息队列的设计与实现[J].黑龙江科学,2023,14(24):72-74+78.
10. 徐智勇,戴祖旭.基于WebSocket协议的消息安全推送系统设计与实现[J].武汉工程大学学报,2024,46(03):310-316.DOI:10.19843/j.cnki.CN42-1779/TQ.202306003.

致 谢

连雨不知春去，一晴方觉夏深。

欲买桂花同载

“我会成为什么样的人呢？” 我曾无数次问自己。

“你想成为的任何人。” 这是我给自己的答案。

是的，自由之风一路吹拂，我将一路前行，直至灯火通明。坎坷之路，终抵星空。感谢走得很慢但一直在前进的自己。