|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **<道具交易平台>**  **系统概要设计报告**  设计人员：      指导教师   |  |  |  | | --- | --- | --- | | **文件状态：**  **[ ] 草稿**  **[ ] 正式发布**  **[√ ] 正在修改** | **文件标识** | **1.0** | | **当前版本** | **V1.0** | | **拟 稿 人** |  | | **拟稿日期** |  | | **审 核 人** |  | | **审核日期** |  | |

编写说明

标题：系统概要设计报告书

类别：文档

存放位置：项目文档\02、项目需求\蟹螃游戏道具交易平台-软件需求规格说明书-V1.0.2.doc

编辑软件：WPS OFFICE

版本历史：

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **作者** | **日期** | **备注** |
| V1.0.1 |  | 2025/3/21 | 无 |
| V1.0.2 |  | 2025/4/27 | 无 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

完成情况分工：

|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | **姓名** | **工作量** | **完成工作内容** |
|  |  |  | 系统体系结构设计 |
|  |  |  | 接口设计 |
|  |  |  | 数据库设计及系统出错处理 |

目录

[编写说明 1](#_Toc14036)

[2.2系统概要设计报告 3](#_Toc17754)

[1、引言 3](#_Toc14896)

[1.1 编写目的 3](#_Toc16806)

[1.2 项目背景 3](#_Toc25549)

[1.3相关术语 3](#_Toc21615)

[1.4参考文献 4](#_Toc15158)

[2、系统体系结构设计 5](#_Toc25359)

[2.1 系统特点分析 5](#_Toc1134)

[2.2 系统体系结构设计 5](#_Toc3394)

[3、接口设计 9](#_Toc17634)

[3.1 外部接口 9](#_Toc7391)

[3.2 内部接口 11](#_Toc30911)

[4、系统数据库设计 13](#_Toc22600)

[4.1 概念数据库设计 13](#_Toc17445)

[4.2 逻辑数据库设计 13](#_Toc30040)

[5、系统出错处理设计 13](#_Toc1745)

[5.1 出错信息 13](#_Toc6368)

[5.2 补救措施 14](#_Toc11824)

**2.2****系统概要设计报告**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1、引言**  1.1 编写目的  本报告旨在将《蟹螃游戏道具交易平台软件需求规格说明书》中定义的功能与非功能需求转化为可落地的技术框架，明确系统分层架构（前端、业务逻辑、数据服务层）、核心模块划分（用户管理、交易引擎、风控审核等）、关键技术选型及非功能性保障方案，为后续详细设计、开发及测试提供统一技术标准，确保系统实现与需求目标的高度一致性。  1.2 项目背景  随着中国游戏产业规模突破3000亿元，虚拟道具交易需求激增，但现有平台面临欺诈率高（32%）、跨游戏检索效率低（平均15秒/次）、合规风险突出等痛点。蟹螃平台通过区块链技术实现道具交易全链路追溯，集成智能风控引擎（欺诈拦截率98.7%）保障资金安全，并设计跨游戏统一检索、价格波动预警等模块，构建安全合规、响应高效（<500ms）的虚拟资产交易基础设施，预计首年服务500万用户并占据15%市场份额，推动行业规范化发展。  1.3相关术语   |  |  | | --- | --- | | **解耦** | 通过技术手段降低系统各模块之间的依赖关系，让每个模块可以独立开发、修改或扩展 | | **部署** | 将应用程序部署到目标计算机环境中的过程 | | **Web系统** | 一种基于Web基数的应用程序，通过浏览器访问和使用 | | **事务交互** | 一种处理数据的方式，将相关操作作为单个逻辑单元执行 | | **数据库** | 用于存储和组织数据的电子化系统 | | **E-R图** | 实体关系图，用于表示应用程序实体和它们之间关系的图形表示 | | **SQL Server** | 一种关系型数据库管理系统​，用于数据存储、处理、事务管理和安全性 | | **区块链技术** | 分布式账本技术，实现交易记录的不可篡改和全程追溯 | | **智能风控引擎** | 基于规则引擎和机器学习模型，实时检测并拦截欺诈交易（文档中拦截率98.7%） | | **MVC设计模式** | 模型-视图-控制器架构模式，实现业务逻辑、数据展示和用户输入的分离 | | **高可用集群** | 通过冗余部署确保系统在节点故障时持续提供服务 | | **阿里云OSS** | 阿里云对象存储服务，用于存储图片、文档等非结构化数据 | | **gRPC协议** | 高性能远程过程调用协议，采用HTTP/2和Protobuf实现高效通信 | | **Redis哨兵集群** | 基于Redis的内存数据库集群，通过哨兵机制实现自动故障转移 | | **MinIO** | 开源分布式对象存储系统，支持海量文件存储和多副本冗余 | | **跨游戏检索** | 支持用户通过统一搜索接口查询多款游戏的虚拟道具 | | **身份验证** | 通过手机验证码、生物识别等技术确认用户身份 |   1.4参考文献  [1]Gamma. E,Helm.R,Johnson.R&Vlissides.J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.  [2].Alibaba Cloud.(2023).对象存储OSS开发指南[EB/OL].https://help.aliyun.com/product/31815.html  [3].NIST.(2020).FIPS180-4SecureHashStandard[EB/OL].https://csrc.nist.gov/publications/detail/fips/180/4/final（SHA-256加密标准）  [4]. popywei. 聊天系统的设计与实现概要设计报告[EB/OL]. CSDN博客. (日期不详)[2024-07-18]. https://blog.csdn.net/xxx/article/details/xxx.  **2、系统体系结构设计**  2.1 系统特点分析  本系统为典型的基于事务交互的交易系统，由用户交互、业务管理和数据库存储组成。系统术语一个典型的数据库数据分析的Web系统。具有以下特点：   1. 提高用户体验：通过合理设计模块分类，使得用户从进入系统到目标界面所需步骤≤5次，提高用户使用系统的体验。 2. 实现业务解耦：降低系统各模块间的依赖关系，让每个模块可以独立开发、修改或扩展，实现业务解耦。 3. 提高安全性：系统实现身份验证，保护用户的个人信息安全。并且对使用系统的用户请求实现区分，确保处理多个请求时不会混淆用户身份，提高系统运行时的安全性。 4. 提高数据支持性：使用阿里云OSS对象存储管理数据，存入数据库后可以对数据进行统计。   2.2 系统体系结构设计  2.2.1系统体系结构模式  本系统采用MVC设计模式与多层次分布式架构构建，由前端交互层、部署于Web服务器的业务逻辑层以及独立运行的数据库服务层三大部分组成。前端作为视图层与后端通信，专注于用户交互行为捕捉与数据可视化呈现；后端作为控制器与模型层，负责处理核心业务逻辑、数据校验转换及API接口调度；数据库层采用关系型数据库管理系统，以高可用集群模式部署在专用数据库服务器，承担结构化数据的持久化存储、索引优化及事务管理。各层级通过标准化协议进行跨服务器协同工作，既实现了功能解耦与独立扩展，又保障了系统整体的高并发处理能力与数据一致性。  2.2.2系统体系结构设计  （1）逻辑视图设计    图一 系统逻辑视图  系统以用户操作流程为核心构建逻辑架构，整体呈现清晰的层级递进关系。从用户注册与登录为起点，用户需填写包含实名姓名（字符型）、手机号（数字型）、身份证号（字符串）等关键信息的注册表单，完成验证后生成用户档案，记录信用等级与交易历史。已认证用户可通过购买/出售道具触发交易流程：购买时系统校验道具库存（整型数值），生成包含订单ID（字符串）、交易金额（整型）、创建时间（日期型）的订单记录，同时更新道具的上架时间（日期型）与库存状态。  交易流程与后台管理模块紧密联动：审核员基于预设规则（如价格浮动阈值、敏感词过滤）对道具信息进行核验，操作数据关联至系统管理员的全局监控体系。所有实体通过数据流贯通：用户操作驱动道具状态变更，订单记录映射交易关系，审核与管理员模块形成监管闭环。核心数据属性严格匹配类型约束（如字符型名称、整型价格），流程箭头明确指向各环节输入输出，构成完整的业务逻辑链条。  （2）开发及运行视图设计    图二 系统开发及运行视图  系统采用典型的三层架构设计，通过表现层、应用层、数据层的垂直分层实现功能解耦与职责分离。  表现层作为用户交互入口，包含 Web前端（React/Vue框架）、移动端（Flutter跨平台应用）及统一的 API网关，通过 HTTP/HTTPS协议 与应用层通信，承担请求路由、身份验证与协议转换职责。  应用层横向扩展为五大核心服务：用户服务处理认证鉴权，商品服务管理库存与价格策略，订单服务依托分布式事务保障交易完整性，支付服务对接第三方支付渠道，统计服务通过实时计算引擎生成业务洞察。各服务间采用 gRPC协议 进行高性能通信，利用长连接与二进制编码提升传输效率。  数据层纵向整合多样化存储方案：结构化数据由MySQL集群承载核心业务表，文件存储基于对象存储（OSS/MinIO）管理图片与文档，缓存数据通过Redis集群加速热点访问，日志数据借助ELK技术栈实现全链路追踪。三层之间通过标准化协议（HTTP/HTTPS→gRPC→存储引擎API）逐级下沉数据，形成从界面交互到持久化存储的完整闭环，既保障了模块化开发的灵活性，又通过分层隔离实现了安全性与性能优化。  （3）部署视图设计    图三 部署视图  该架构图采用分层设计，清晰展现用户请求从接入到数据存储的全流程，共分为三层：  1. 用户请求接入层  互联网用户发起请求，通过内容分发网络（CDN缓存静态资源（如图片、文件），动态请求路由至负载均衡器（如Nginx集群），实现请求分发。  2. 应用服务层  核心组件：  Web服务器：处理HTTP请求解析与基础业务逻辑  API网关：统一路由、鉴权与流量控制  业务集群：执行具体业务处理（如订单生成、支付逻辑）  数据交互：通过内部调用链连接各组件，形成完整业务链路  3. 数据基础设施层  存储服务：  MySQL主从集群：持久化核心业务数据，支持高可用读写  Redis哨兵集群：缓存热点数据，提升查询性能  MinIO分布式存储：存储图片/文件，支持多副本冗余  数据流向：业务集群通过箭头与存储组件交互，完成数据读写   1. 接口设计   3.1 外部接口  3.1.1第三方支付接口   |  |  | | --- | --- | | **接口名称** | 支付请求接口 | | **接口用途** | 与支付宝/微信支付系统对接，完成订单支付和退款操作 | | **协议** | HTTPS | | **数据格式** | JSON | | **请求参数** | - order\_id（订单号） - amount（金额，单位：分） - user\_id（用户标识） | | **返回参数** | - payment\_id（支付流水号） - status（支付状态：成功/失败 | | **安全机制** | - 请求参数签名（SHA256） - 敏感信息加密传输（AES-256） |   3.1.2游戏数据库同步接口   |  |  | | --- | --- | | **接口名称** | 道具库存验证接口 | | **接口用途** | 验证游戏内道具库存有效性 | | **协议** | WebSocket | | **数据格式** | Protobuf | | **请求参数** | - game\_id（游戏ID） - item\_id（道具ID） - user\_token（用户令牌） | | **返回参数** | - valid（有效性：true/false） - expire\_time（道具有效期） |   3.1.3通知服务接口   |  |  | | --- | --- | | **接口名称** | 短信/邮件通知接口 | | **接口用途** | 进行消息通知 | | **协议** | HTTP | | **数据格式** | XML | | **请求参数** | - template\_id（模板ID） - recipients（接收者列表） - content（内容） | | **返回参数** | - send\_status（发送状态码） | | **触发场景** | - 用户注册验证码 - 订单状态变更通知 - 安全风险告警 |   3.1.4游戏内交易API   |  |  | | --- | --- | | **接口名称** | 道具转移确认接口 | | **接口用途** | 当交易完成后，通过游戏厂商API执行游戏内道具所有权转移 | | **协议** | RESTful API | | **认证方式** | OAuth 2.0 |     3.2 内部接口  3.2.1用户服务接口   |  |  | | --- | --- | | **接口名称** | 用户认证接口 | | **协议** | gRPC | | **请求参数** | - username - password（HMAC加密） - device\_id | | **返回参数** | - access\_token（JWT令牌） - role（用户角色） | | **QPS要求** | ≥1000次/秒 |   3.2.2商品服务接口   |  |  | | --- | --- | | **接口名称** | 商品上架审核接口 | | **协议** | RESTful API | | **请求参数** | - item\_data（商品元数据） - seller\_id - game\_id | | **返回参数** | - audit\_status（审核状态） - reject\_reason（驳回原因）） | | **审核规则** | 敏感词过滤/价格合理性检查/图片合规性验证 |   3.2.3订单服务接口   |  |  | | --- | --- | | **接口名称** | 订单创建接口 | | **协议** | RabbitMQ（消息队列） | | **消息结构** | // 订单创建消息示例  { "user\_id": "U123",  "items": [{"item\_id": "I456", "quantity": 1}],  "payment\_method": "alipay"} | | **事务保证** | 分布式事务（Saga模式）： - 预扣库存 - 生成订单流水 - 支付状态跟踪 |   3.2.4支付服务接口   |  |  | | --- | --- | | **接口名称** | 支付状态回调接口 | | **协议** | Webhook | | **验证机制** | 双重验证： 1. 签名验证 2. 订单状态幂等性检查 | | **处理逻辑** | - 更新订单状态 - 触发资金托管操作 - 通知买卖双方 |   3.2.5数据统计接口   |  |  | | --- | --- | | **接口名称** | 热榜生成接口 | | **计算周期** | 每15分钟增量更新 每日00:00全量计算 | | **算法** | 热度值 = 交易量×0.6 + 搜索量×0.3 + 收藏量×0.1 | | **输出形式** | Redis缓存（ZSET结构）  // 热榜数据示例  { "game\_id": "G001",  "top\_items": [{"item\_id": "I123", "heat\_value": 95.6}]  } |   3.2.6消息服务接口   |  |  | | --- | --- | | **接口名称** | 实时通知推送接口 | | **协议** | WebSocket | | **推送场景** | - 价格变动提醒 - 库存预警（<5件） - 订单状态变更 | | **性能指标** | 同时在线连接数≥10万 消息延迟<500ms |   **4、系统数据库设计**  4.1 概念数据库设计    表 1:概念数据库ER图  4.2 逻辑数据库设计  用PowerDesigner等工具进行数据库逻辑建模，完成数据库的逻辑设计，导出SQL，创建数据库。  **5、系统出错处理设计**  5.1 出错信息   | **错误类型** | **错误代码** | **表现形式** | **用户提示信息** | **日志记录内容** | | --- | --- | --- | --- | --- | | 支付超时 | ERR\_5001 | 支付页面卡顿 | "支付通道繁忙，请稍后重试（代码5001）" | [TIMESTAMP] Payment timeout: order\_id={}, user\_id={} | | 库存冲突 | ERR\_6003 | 交易失败弹窗 | "该道具已被其他玩家购买，正在刷新库存..." | [TIMESTAMP] Inventory conflict: item\_id={}, game\_id={} | | 身份验证失败 | ERR\_4003 | 登录界面红字提示 | "安全验证未通过（生物识别/短信验证码错误）" | [TIMESTAMP] Auth failure: ip={}, device\_id={} | | 数据同步异常 | ERR\_8002 | 系统公告栏警告 | "游戏数据同步延迟，部分功能暂不可用" | [TIMESTAMP] GameAPI sync failed: code={}, response={} | | 风控拦截 | ERR\_9001 | 交易冻结提示 | "您的交易存在风险，客服将在30分钟内联系您" | [TIMESTAMP] Risk control triggered: rule\_id={}, score={} |   5.2 补救措施  1. 实时交易类错误  自动重试机制：对支付超时、库存校验失败等情况实施指数退避重试（最多3次）  备用通道切换：当主支付接口不可用时自动切换至备用支付通道（微信→支付宝→银联）   1. 数据一致性错误​​   区块链补偿：通过智能合约自动校验交易链，对未完成的交易执行回滚操作  定时对账任务：每30分钟执行一次订单-库存-资金的三方对账   1. 系统级故障   服务降级方案：  核心交易模块：保持最低限度的下单/支付功能  非核心功能：临时关闭搜索推荐、热榜更新等  灾备切换流程：     1. 安全应急方案​​   当检测到大规模撞库攻击时：  自动启用临时验证码+图形验证双因素认证  对异常IP段实施15分钟流量清洗  通过短信/邮件通知受影响用户修改密码   1. 监控与通知​​    * 分级告警机制：  | **严重级别** | **响应时间** | **通知方式** | | --- | --- | --- | | P0（交易中断） | 5分钟 | 电话+企业微信+邮件 | | P1（部分功能异常） | 15分钟 | 企业微信+邮件 | | P2（性能下降） | 30分钟 | 邮件+监控看板 | |