课 程 设 计 报 告

题 目： 网上购物子系统

院（系）：

专 业:

班 级：

学 号:

姓 名：

指导老师：

成绩：

目录

[易购网上购物子系统 3](#_Toc13044382)

[1 开发环境与开发工具 3](#_Toc13044383)

[2 系统需求分析 3](#_Toc13044384)

[2.1 功能概述 3](#_Toc13044385)

[2.2 系统数据流图 3](#_Toc13044386)

[2.3 系统数据字典 4](#_Toc13044387)

[2.3.1 商品信息 4](#_Toc13044388)

[2.3.2 订单信息 7](#_Toc13044389)

[2.3.3 用户信息 9](#_Toc13044390)

[2.3.4日志信息 12](#_Toc13044391)

[3 系统功能分析 14](#_Toc13044392)

[3.1系统功能层次模块图 14](#_Toc13044393)

[3.2 各子模块功能的描述 17](#_Toc13044394)

[3.2.1首页-顶部导航栏（已登录） 17](#_Toc13044395)

[3.2.2首页-顶部导航栏（未登录） 17](#_Toc13044396)

[3.2.3首页-轮播图 17](#_Toc13044397)

[3.2.4首页-搜索框 17](#_Toc13044398)

[3.2.5首页-热销商品列表 18](#_Toc13044399)

[3.2.6首页-按类商品列表 18](#_Toc13044400)

[3.2.7购物车 18](#_Toc13044401)

[3.2.8个人中心 18](#_Toc13044402)

[3.2.9商品详情 19](#_Toc13044403)

[3.2.10订单列表 19](#_Toc13044404)

[4 系统设计 22](#_Toc13044405)

[4.1 数据库概念结构设计 22](#_Toc13044406)

[4.2 系统功能模块设计 24](#_Toc13044407)

[4.3 数据库逻辑结构设计 26](#_Toc13044408)

[4.4 数据库物理结构设计及其实现 26](#_Toc13044409)

[5 系统实现 30](#_Toc13044410)

[5.1 编写目的 30](#_Toc13044411)

[5.2 背景 30](#_Toc13044412)

[5.3 基线 30](#_Toc13044413)

[5.4 范围 30](#_Toc13044414)

[5.5 模块命名规则 30](#_Toc13044415)

[5.6 模块汇总 30](#_Toc13044416)

[5.6.1模块汇总表 30](#_Toc13044417)

[5.6.2模块关系图 31](#_Toc13044418)

[5.7子系统模块设计 32](#_Toc13044419)

[5.7.1 模块1 网上购物系统用户端 32](#_Toc13044420)

[5.7.2 模块2 系统后台管理——人员模块 40](#_Toc13044421)

[5.7.3 模块3 系统后台管理——商品模块 44](#_Toc13044422)

[5.7.4 模块4 系统后台管理——商品上架模块 48](#_Toc13044423)

[5.7.5 模块5 交易模块 54](#_Toc13044424)

[5.7.6 模块6 系统后台管理——订单管理模块 57](#_Toc13044425)

[6 系统测试 63](#_Toc13044426)

[6.1 测试方式 63](#_Toc13044427)

[6.2 测试结果 63](#_Toc13044428)

[7 体会与自我评价 63](#_Toc13044429)

[7.1 技能提升 63](#_Toc13044430)

[7.2 能力提升 65](#_Toc13044431)

**易购网上购物子系统**

**1 开发环境与开发工具**

开发环境：Win10

开发工具：Intellij IDEA Ultimate 和 WebStorm

**2 系统需求分析**

**2.1 功能概述**

本软件旨在实现一个包括商品的支付、发货、收货和评价的全流程商品交易过程，同时实现后台对用户信息、商品信息、订单信息和日志信息的实时查看和管理，真正做到全流程实时交易且可靠真实的网上购物系统。

**2.2 系统数据流图**



图2.1.1 网上购物系统中“商品交易全流程”分解后的数据流图



图2.2.2 网上购物系统后台“商品管理”分解后的数据流图

**2.3 系统数据字典**

**2.3.1 商品信息**

（1）“商品信息”数据结构

① 名字：商品信息。

② 别名：商品详细信息。

③ 描述：管理员通过编辑商品详细信息，添加或修改商品详细信息后入库。

④ 定义：商品详细信息 = 商品编号 + 名称 + 种类编号 + 价格 + 库存 + 图片 + 简介 + 详细介绍 + 销量 + 创建时间 + 更新时间 + 状态 + 上架时间。

⑤ 位置：保存至商品信息表或打印保存。

（2）“商品信息”数据结构之数据项

表1-1 “商品编号”数据项

|  |  |
| --- | --- |
| 名字 | 商品编号 |
| 别名 | 顺序号 |
| 描述 | 唯一标识某商品的数字编号 |
| 定义 | 整型数 |
| 位置 | 商品信息表、订单明细表 |

表1-2 “名称”数据项

|  |  |
| --- | --- |
| 名字 | 名称 |
| 别名 | 商品名称 |
| 描述 | 商品的名称 |
| 定义 | 字符型汉字名称，汉字数<=30 |
| 位置 | 商品信息表、订单明细表 |

表1-3 “种类编号”数据项

|  |  |
| --- | --- |
| 名字 | 种类编号 |
| 别名 | 种类编号 |
| 描述 | 用于连接商品种类标志所对应的商品种类字符串 |
| 定义 | 整型数 |
| 位置 | 商品信息表、商品种类表 |

表1-4 “价格”数据项

|  |  |
| --- | --- |
| 名字 | 价格 |
| 别名 | 商品价格 |
| 描述 | 商品的价格 |
| 定义 | 浮点数 |
| 位置 | 商品信息表、订单明细表 |

表1-5 “库存”数据项

|  |  |
| --- | --- |
| 名字 | 库存 |
| 别名 | 商品库存 |
| 描述 | 商品的库存，即当前系统中剩余商品的数量 |
| 定义 | 整型数 |
| 位置 | 商品信息表 |

表1-6 “图片”数据项

|  |  |
| --- | --- |
| 名字 | 图片 |
| 别名 | 商品描述图片 |
| 描述 | 商品的相关描述图片 |
| 定义 | 可变长度的字符型英文名称 |
| 位置 | 商品信息表 |

表1-7 “简介”数据项

|  |  |
| --- | --- |
| 名字 | 简介 |
| 别名 | 商品简介 |
| 描述 | 对于商品的简要描述 |
| 定义 | 字符型汉字名称，汉字数<=140 |
| 位置 | 商品信息表 |

表1-8 “详细介绍”数据项

|  |  |
| --- | --- |
| 名字 | 详细介绍 |
| 别名 | 商品详情 |
| 描述 | 对于商品的详细描述 |
| 定义 | 可变长度的字符型汉字名称 |
| 位置 | 商品信息表 |

表1-9 “销量”数据项

|  |  |
| --- | --- |
| 名字 | 销量 |
| 别名 | 商品销量 |
| 描述 | 记录商品的销量 |
| 定义 | 整型数 |
| 位置 | 商品信息表 |

表1-10 “状态”数据项

|  |  |
| --- | --- |
| 名字 | 状态 |
| 别名 | 商品状态 |
| 描述 | 记录商品的状态，主要包括“已上架”和“未上架”两种状态 |
| 定义 | 字符型英文名称，英文数<=10 |
| 位置 | 商品信息表 |

表1-11 “创建时间”数据项

|  |  |
| --- | --- |
| 名字 | 创建时间 |
| 别名 | 商品创建时间 |
| 描述 | 记录商品的创建时间 |
| 定义 | 时间类型 |
| 位置 | 商品信息表 |

表1-12 “更新时间”数据项

|  |  |
| --- | --- |
| 名字 | 更新时间 |
| 别名 | 商品更新时间 |
| 描述 | 记录商品的更新时间 |
| 定义 | 时间类型 |
| 位置 | 商品信息表 |

表1-13 “上架时间”数据项

|  |  |
| --- | --- |
| 名字 | 上架时间 |
| 别名 | 商品上架时间 |
| 描述 | 记录商品的上架时间 |
| 定义 | 时间类型 |
| 位置 | 商品信息表 |

（3）数据流

数据流名：商品信息数据流。

说明：“商品信息”数据结构在系统内的流向。

数据流来源：管理员创建、修改商品事务。

数据流去向：用户查看和管理员查看等事务。

平均流量：每天几十次。

高峰期流量：每天上百次。

（4）数据存储

① 数据库存储名：商品信息表

② 说明：商品信息数据，用于用户和管理员进行查看等操作。

③ 编号：商品信息为唯一标识，顺序整数，从1开始每次增加1。

④ 输入的数据流：创建和修改商品的数据流，来自管理员创建和修改商品。

⑤ 输出的数据流：查找商品的数据流，用于用户和管理员查看商品的相关信息。

⑥ 数据结构：“商品信息”“商品种类信息”

⑦ 数据量：一天，100 X 80 = 8 000字节。

⑧ 存取频度：每小时存取更新 12~18次，查询大于等于80次。

⑨ 存取方式：顺序插入、联机处理、检索与更新、顺序检索与随机检索。

（5）处理过程

① 处理过程名：实时商品分类查找

② 说明：根据外部提供的所要查找的商品种类名称，要能够实时的查找该种类的全部商品。

③ 输入：商品种类的名称。

④ 输出：属于该商品种类的全部商品。

⑤ 处理：通过数据库引擎先从商品种类信息表中查找当前种类名称所对应的种类编号，然后到商品信息表中根据商品种类编号等于刚刚查找到的商品种类编号来进行精确查找，处理的频度为每小时20~40次，每当有用户进入到首页或者需要根据商品种类来对商品进行分类时都要引发实时商品分类查找事务，实时商品分类查找所涉及的数据为每小时8~20KB，希望能够在查找时取得实时的查找反馈效果。

**2.3.2 订单信息**

（1）“订单信息”数据结构

① 名字：订单信息。

② 别名：订单信息。

③ 描述：用户支付后会创建订单入库，之后用户和管理员每次对订单状态的更新都会同步入库。

④ 定义：订单信息 = 订单编号 + 用户编号 + 状态 + 商品数量 + 商品总价 + 运单号 + 创建时间 + 更新时间。

⑤ 位置：保存至订单信息表或打印保存。

（2）“订单信息”数据结构之数据项

表2-1 “订单编号”数据项

|  |  |
| --- | --- |
| 名字 | 订单编号 |
| 别名 | 顺序号 |
| 描述 | 唯一标识某订单的数字编号 |
| 定义 | 整型数 |
| 位置 | 订单信息表、订单明细表 |

表2-2 “用户编号”数据项

|  |  |
| --- | --- |
| 名字 | 名称 |
| 别名 | 用户编号 |
| 描述 | 外连该订单的创建用户 |
| 定义 | 整型数 |
| 位置 | 用户信息表、订单信息表 |

表2-3 “状态”数据项

|  |  |
| --- | --- |
| 名字 | 状态 |
| 别名 | 订单状态 |
| 描述 | 记录订单的处理进度信息 |
| 定义 | 字符型英文名称，英文数<=20 |
| 位置 | 商品信息表、商品种类表 |

表2-4 “商品数量”数据项

|  |  |
| --- | --- |
| 名字 | 商品数量 |
| 别名 | 商品数量 |
| 描述 | 订单中所包含的各类商品的总数量 |
| 定义 | 整型数 |
| 位置 | 订单信息表 |

表2-5 “商品总价”数据项

|  |  |
| --- | --- |
| 名字 | 商品总价 |
| 别名 | 商品总价 |
| 描述 | 订单中所包含的各类商品的总金额 |
| 定义 | 浮点数 |
| 位置 | 订单信息表 |

表2-6 “运单号”数据项

|  |  |
| --- | --- |
| 名字 | 运单号 |
| 别名 | 快递单号 |
| 描述 | 记录订单的快递单号 |
| 定义 | 可变长度的字符型数字名称 |
| 位置 | 订单信息表 |

表2-7 “创建时间”数据项

|  |  |
| --- | --- |
| 名字 | 创建时间 |
| 别名 | 订单创建时间 |
| 描述 | 记录订单的创建时间 |
| 定义 | 时间类型 |
| 位置 | 订单信息表 |

表2-8 “更新时间”数据项

|  |  |
| --- | --- |
| 名字 | 更新时间 |
| 别名 | 订单更新时间 |
| 描述 | 记录订单状态的更新时间 |
| 定义 | 时间类型 |
| 位置 | 订单信息表 |

（3）数据流

数据流名：订单信息数据流。

说明：“订单信息”数据结构在系统内的流向。

数据流来源：用户创建订单事务，管理员和用户共同修改订单状态事务。

数据流去向：用户创建、查看、更新和管理员查看、更新等事务。

平均流量：每天几十次。

高峰期流量：每天上百次。

（4）数据存储

① 数据库存储名：订单信息表

② 说明：订单信息数据，用于用户和管理员进行创建、查看和更新等操作。

③ 编号：订单信息为唯一标识，顺序整数，从1开始每次增加1。

④ 输入的数据流：创建和修改订单的数据流，来自用户和管理员创建和更新订单。

⑤ 输出的数据流：查找订单的数据流，用于用户和管理员查看订单的相关信息。

⑥ 数据结构：“订单信息”“订单明细信息”

⑦ 数据量：一天，80 X 80 = 1 600字节。

⑧ 存取频度：每小时存取更新 12~18次，查询大于等于80次。

⑨ 存取方式：顺序插入、联机处理、检索与更新、顺序检索与随机检索。

（5）处理过程

① 处理过程名：实时订单状态更新

② 说明：根据外部提供的所要更新的订单状态，要能够实时的实时的对订单的状态进行更新。

③ 输入：订单编号和订单所要更新的状态。

④ 输出：更新后的订单的相关信息。

⑤ 处理：通过数据库引擎从订单信息列表根据订单号来查找订单，之后再根据所要更新的订单状态来对订单的状态进行更新，处理的频度为每小时20~40次，每当有用户或管理员要对订单的状态进行更新时就会触发实时订单状态更新事务，实时订单状态更新所涉及的数据为每小时8~20KB，希望能够在更新后取得实时的查找反馈效果。

**2.3.3 用户信息**

（1）“用户信息”数据结构

① 名字：用户信息。

② 别名：用户详细信息。

③ 描述：用户通过创建、修改和管理员进行锁定/解锁和删除/恢复用户信息后入库。

④ 定义：用户信息 = 用户编号 + 姓名 + 电话 + 邮箱 + 状态 + 地址 + 昵称 + 密码 + 创建时间 + 最后登陆时间。

⑤ 位置：保存至用户信息表或打印保存。

（2）“用户信息”数据结构之数据项

表3-1 “用户编号”数据项

|  |  |
| --- | --- |
| 名字 | 商品编号 |
| 别名 | 顺序号 |
| 描述 | 唯一标识某用户的数字编号 |
| 定义 | 整型数 |
| 位置 | 用户信息表、订单信息表、订单处理记录表 |

表3-2 “姓名”数据项

|  |  |
| --- | --- |
| 名字 | 名称 |
| 别名 | 用户姓名 |
| 描述 | 记录用户的姓名 |
| 定义 | 字符型汉字名称，汉字数<=10 |
| 位置 | 用户信息表 |

表3-3 “电话”数据项

|  |  |
| --- | --- |
| 名字 | 电话 |
| 别名 | 用户电话 |
| 描述 | 记录用户的电话 |
| 定义 | 字符型数字名称，数字数=11 |
| 位置 | 用户信息表 |

表3-4 “邮箱”数据项

|  |  |
| --- | --- |
| 名字 | 邮箱 |
| 别名 | 用户邮箱 |
| 描述 | 记录用户的邮箱 |
| 定义 | 可变长度的字符型 |
| 位置 | 用户信息表 |

表3-5 “状态”数据项

|  |  |
| --- | --- |
| 名字 | 状态 |
| 别名 | 用户状态 |
| 描述 | 记录用户的状态 |
| 定义 | 字符型英文名称，英文数<=10 |
| 位置 | 用户信息表 |

表3-6 “地址”数据项

|  |  |
| --- | --- |
| 名字 | 地址 |
| 别名 | 用户地址 |
| 描述 | 记录用户的地址信息 |
| 定义 | 可变长度的字符型汉字名称 |
| 位置 | 用户信息表 |

表3-7 “昵称”数据项

|  |  |
| --- | --- |
| 名字 | 昵称 |
| 别名 | 用户昵称 |
| 描述 | 记录用户的昵称用于评价的相关信息的显示 |
| 定义 | 字符型汉字名称，汉字数<=20 |
| 位置 | 用户信息表 |

表3-8 “密码”数据项

|  |  |
| --- | --- |
| 名字 | 密码 |
| 别名 | 密码 |
| 描述 | 记录非对称加密后的用户密码信息 |
| 定义 | 可变长度的字符型 |
| 位置 | 用户信息表 |

表3-9 “创建时间”数据项

|  |  |
| --- | --- |
| 名字 | 创建时间 |
| 别名 | 用户创建时间 |
| 描述 | 记录用户账号的创建时间 |
| 定义 | 时间类型 |
| 位置 | 用户信息表 |

表3-10 “最后登录时间”数据项

|  |  |
| --- | --- |
| 名字 | 最后登陆时间 |
| 别名 | 用户最后登录时间 |
| 描述 | 记录用户的最后登陆时间 |
| 定义 | 时间类型 |
| 位置 | 用户信息表 |

（3）数据流

数据流名：用户信息数据流。

说明：“用户信息”数据结构在系统内的流向。

数据流来源：用户创建、修改用户信息和管理员更新用户状态的事务。

数据流去向：用户查看、修改信息和管理员查看、修改状态等事务。

平均流量：每天几十次。

高峰期流量：每天上百次。

（4）数据存储

① 数据库存储名：用户信息表

② 说明：用户信息数据，用于用户和管理员进行查看、修改等操作。

③ 编号：用户信息为唯一标识，顺序整数，从1开始每次增加1。

④ 输入的数据流：创建和修改用户信息的数据流，来自用户创建和修改用户信息、管理员修改用户状态。

⑤ 输出的数据流：查找用户信息的数据流，用于用户和管理员查看用户的相关信息。

⑥ 数据结构：“用户信息”“订单信息”“订单处理记录”

⑦ 数据量：一天，100 X 80 = 8 000字节。

⑧ 存取频度：每小时存取更新 12~18次，查询大于等于80次。

⑨ 存取方式：顺序插入、联机处理、检索与更新、顺序检索与随机检索。

（5）处理过程

① 处理过程名：用户状态实时更新

② 说明：根据外部提供的所要更新的用户的编号和状态，要能够实时的更新该用户的状态。

③ 输入：用户编号和所要更新的用户状态。

④ 输出：更新后的用户信息。

⑤ 处理：通过数据库引擎先从用户信息表中根据用户编号查找用户信息，并根据外部提供的所要更新的状态对其进行状态更新，处理的频度为每小时20~40次，当管理员在后台对用户状态进行操作时就会触发用户状态实时更新事务，用户状态实时更新所涉及的数据为每小时8~20KB，希望能够在查找时取得实时的查找反馈效果。

**2.3.4日志信息**

（1）“日志信息”数据结构

① 名字：日志信息。

② 别名：日志记录信息。

③ 描述：系统通过自动记录管理员的登陆和操作以及系统的故障后生成日志信息入库。

④ 定义：日志信息 = 日志编号 + 类型 + 管理员编号 + 操作 + 模块 + 结果 + 时间。

⑤ 位置：保存至日志记录信息表或打印保存。

（2）“日志信息”数据结构之数据项

表4-1 “日志编号”数据项

|  |  |
| --- | --- |
| 名字 | 日志编号 |
| 别名 | 顺序号 |
| 描述 | 唯一标识某日志的数字编号 |
| 定义 | 整型数 |
| 位置 | 日志信息表 |

表4-2 “类型”数据项

|  |  |
| --- | --- |
| 名字 | 类型 |
| 别名 | 操作类型 |
| 描述 | 该日志的具体操作类型 |
| 定义 | 字符型英文呢名称，英文数<=10 |
| 位置 | 日志信息表 |

表4-3 “管理员编号”数据项

|  |  |
| --- | --- |
| 名字 | 管理员编号 |
| 别名 | 操作者编号 |
| 描述 | 用于记录操作者的编号 |
| 定义 | 整型数 |
| 位置 | 日志信息表、管理员信息表 |

表4-4 “操作”数据项

|  |  |
| --- | --- |
| 名字 | 操作 |
| 别名 | 具体操作 |
| 描述 | 日志所记录的具体操作 |
| 定义 | 字符型英文名称，英文数<=30 |
| 位置 | 日志信息表 |

表4-5 “模块”数据项

|  |  |
| --- | --- |
| 名字 | 模块 |
| 别名 | 操作模块 |
| 描述 | 记录日志所操作的模块信息 |
| 定义 | 字符型英文名称，英文数<=20 |
| 位置 | 日志信息表 |

表4-6 “结果”数据项

|  |  |
| --- | --- |
| 名字 | 结果 |
| 别名 | 操作结果 |
| 描述 | 记录具体操作的结果 |
| 定义 | 可变长度的字符型英文名称 |
| 位置 | 日志信息表 |

表4-7 “创建时间”数据项

|  |  |
| --- | --- |
| 名字 | 创建时间 |
| 别名 | 商品创建时间 |
| 描述 | 记录商品的创建时间 |
| 定义 | 时间类型 |
| 位置 | 商品信息表 |

表4-8 “时间”数据项

|  |  |
| --- | --- |
| 名字 | 时间 |
| 别名 | 操作时间 |
| 描述 | 记录日志的生成时间 |
| 定义 | 时间类型 |
| 位置 | 日志信息表 |

（3）数据流

数据流名：日志信息数据流。

说明：“日志信息”数据结构在系统内的流向。

数据流来源：系统自动记录管理员操作和系统错误日志。

数据流去向：管理员查看等事务。

平均流量：每天几十次。

高峰期流量：每天上百次。

（4）数据存储

① 数据库存储名：日志信息表

② 说明：日志信息数据，用于管理员进行查看等操作。

③ 编号：日志信息为唯一标识，顺序整数，从1开始每次增加1。

④ 输入的数据流：系统自动记录管理员操作和系统错误日志。

⑤ 输出的数据流：查找日志的数据流，用于管理员查看后台操作和系统错误的相关信息。

⑥ 数据结构：“日志信息”

⑦ 数据量：一天，100 X 80 = 8 000字节。

⑧ 存取频度：每小时存取更新 12~18次，查询大于等于80次。

⑨ 存取方式：顺序插入、联机处理、检索与更新、顺序检索与随机检索。

（5）处理过程

① 处理过程名：实时日志分类查找

② 说明：根据外部提供的所要查找的日志种类名称，要能够实时的查找该种类的全部日志。

③ 输入：日志种类的名称。

④ 输出：属于该日志种类的全部日志。

⑤ 处理：通过数据库引擎先从日志信息表中通过日志类型来精确查找该类型的全部日志信息，处理的频度为每小时20~40次，每当有管理员进入到日志查看页面时就会触发实时日志分类查找事务，实时日志分类查找所涉及的数据为每小时8~20KB，希望能够在查找时取得实时的查找反馈效果。

**3 系统功能分析**

**3.1系统功能层次模块图**



图3.1.1 网站总模块图



图3.1.2 网站用户端页面结构



图3.1.3 网站管理端页面结构



图3.1.4 网站功能结构

**3.2 各子模块功能的描述**

**3.2.1首页-顶部导航栏（已登录）**

**级别：**重要

**内容：**“首页”“个人中心”“我的订单”“购物车”“退出登录”

**位置：**网站所有页面顶部

**功能：**为网站各页面的导航入口，提供至“首页”、“个人中心”、“我的订单”、“购物车”和“退出登录”的入口链接

其中“我的订单”中又提供了“待支付”，“待发货”，“待收货”，“待评价”，“已完成”和“全部”几种不同进度订单的页面入口，每个页面进入后会显示对应进度的订单列表，并提供不同进度订单的对应操作

**3.2.2首页-顶部导航栏（未登录）**

**级别：**主要

**内容：**“首页”“登陆”“注册”

**位置：**网站所有页面顶部

**功能：**为网站各页面的导航入口，提供至“首页”、“登陆”和“注册”的入口链接

**3.2.3首页-轮播图**

**级别：**重要

**内容：**购物系统中的热销商品和近期活动介绍

**位置：**首页中上部

**功能：**为用户提供当前系统中关于热销商品和近期活动的相关信息。

**3.2.4首页-搜索框**

**级别：**重要

**内容：**商品搜索框

**位置：**首页中上部

**功能：**可以选择不同的商品种类作为搜索的范围，通过商品的名称进行精确查找或者模糊查找。

**3.2.5首页-热销商品列表**

**级别：**主要

**内容：**展示购物系统中当前月销量前三的商品

**位置：**首页中下部

**功能：**按照商品的当月销售量对商品进行排序，选取较高的前三位放置在当前列表中，用户可以通过热销商品列表查看热销商品的图片和简介，并可以通过其下方对应的“立即购买”和“详情”链接分别跳转至支付页面和商品详情页面。

同时在标题右上方提供“查看更多”链接，用户可以通过该链接进入到商品列表页面查看更多按月销量进行排序的商品。

**3.2.6首页-按类商品列表**

**级别：**主要

**内容：**按不同商品种类选取销量前三的商品进行展示

**位置：**首页中下部

**功能：**按照商品种类对商品进行分类，其左上方提供当前种类的名称，右上方提供“查看更多”的链接，用户通过该链接可以进入到该种类商品的列表，查看所有的当前类别的商品。

中下部提供三个按照月销量进行排序的当前种类的商品，每个商品提供一张商品的图片、商品的名称和商品的简介供用户进行参考，同时每个商品的正下方提供“立即购买”和“详情”两个链接，通过这两个链接分别可以直接选取一个当前商品并直接跳转至支付页面和进入当前商品的详情页面。

**3.2.7购物车**

**级别：**主要

**内容：**显示已加入购物车的相关商品信息

**入口：**首页导航栏“购物车”

**功能：**在当前页面可以对已加入购物车中的商品的名称、商品单价、商品数量和商品总价进行查看，并支持对购物车中的商品进行操作。

具体的操作为增加当前商品的购买数量、减少当前商品的购买数量和移除当前商品。在购物车页面的底部提供当前购物车内商品的总件数和总价格，并提供支持了“提交订单”操作的按钮。

**3.2.8个人中心**

**级别：**主要

**内容：**查看和修改个人信息

**入口：**首页导航栏“个人中心”

**功能：**在个人中心页面中可以查看个人信息、修改部分个人信息、查看个人不同进度订单的数量和查看个人订单的详细记录。

其中个人订单记录又可按订单的进度分为：“待支付”，“待发货”，“待收货”，“待评价”，“已完成”，“全部”。

不同进度的订单提供不同的操作，例如“待支付”订单提供支付和取消订单两种操作，“待发货”订单提供查看详情和催单两种操作，“待收货”订单提供查看详情和收货两种操作，“待评价”订单提供查看详情和评价两种操作，“已完成”订单只提供查看详情操作。

**3.2.9商品详情**

**级别：**主要

**内容：**查看和购买商品

**入口：**首页商品列表中的“详情”，订单列表页面的“详情”等

**功能：**商品详情页面左上部提供商品的名称和相关图片，右上部提供商品的价格、销量、评价、库存、简介并支持相关的商品操作，具体的操作包括：增加或减少选购商品的数量、“立即购买”和“加入购物车”。

通过“立即购买”可以选购上方所确定的商品数量的商品并直接跳转到支付页面进行支付，“加入购物车”可以将指定数量的商品加入到购物车，此时购物车中商品订单的进度状态为待结算。

页面的下部主要为该商品的详细介绍、商品评价和销售记录。“商品评价”中所提供的信息为评价者的昵称并隐匿后几位、评价的星级、评价的文字和图片以及评价创建的时间。“销售记录”中提供购买当前商品的用户的昵称并隐匿后几位、购买商品的数量、订单的当前进度状态、订单创建的时间以及跳转订单评价的链接。

**3.2.10订单列表**

**级别：**主要

**内容：**显示特定进度的全部订单

**入口：**首页顶部导航栏“我的订单”

**功能：**按照订单的不同进度“待支付”，“待发货”，“待收货”，“待评价”，“已完成”，“全部”来进行分类显示。

每个订单提供订单编号、订单状态、订单金额、运单号、创建时间、更新时间和订单操作。

不同进度的订单提供不同的操作，例如“待支付”订单提供支付和取消订单两种操作，“待发货”订单提供查看详情和催单两种操作，“待收货”订单提供查看详情和收货两种操作，“待评价”订单提供查看详情和评价两种操作，“已完成”订单只提供查看详情操作。

不同进度的订单此处显示的提示不同，例如：“待发货”进度订单列表中此处显示为“支付时间”，“待收货”进度订单列表中此处显示为“发货时间”，“待评价”进度订单列表中此处显示为“收货时间”，“已完成”进度订单列表此处显示为“完成时间”，“全部”订单列表中显示为“更新时间”。

**订单详情**

**级别：**主要

**内容：**显示订单的详细信息

**入口：**订单列表的订单操作中的“详情”

**功能：**页面的上部显示订单的描述信息订单编号、订单状态、订单总金额、订单运单号、订单的创建时间和订单的最后更新时间。

页面的中部显示订单明细，即订单中所包含的商品的名称、单价、数量、总金额以及更多操作（查看商品详情），同时还包括订单中所包含的商品的总件数和总金额。

页面的下部显示订单的处理记录，主要提供的信息就是订单的状态和状态改变的时间。

**后台管理**

**“会员管理”：**页面提供用户的相关信息：用户编号、用户姓名、用户状态、用户电话、用户邮箱、最后登录时间以及创建时间，同时提供了查看用户更加详细信息、锁定/解锁用户和删除/恢复用户使用的相关操作。

“详情”中提供了用户更加详细的信息，包括更加详细的个人信息以及用户的订单信息。

“锁定”用户会限制用户的部分功能，“解锁”反之。

“删除”会假删除用户，使用户无法登陆，“恢复”反之。

**“管理员管理”：**页面提供了关于管理员的相关的信息：姓名、手机、权限等，并提供了查看详情和查看操作记录等相关操作。

“详情”页面中可以查看到管理员更加详细的个人信息，以及如果当前操作的管理员的权限高于此管理员，则还可以对当前管理员的权限进行提升和降级操作。

**“上架商品”**：通过此页面可以在填写商品的相关信息，如名称、类别、价格、库存、简介、上架类型（立即上架或定时上架）、详细介绍和商品图片等，之后完成对商品的上架操作。

如果商品的上架类型为“立即上架”，则商品会被立即上架。

如果商品的上架类型为“定时上架”，则商品会根据管理员设置的时间而定时上架。

**“商品列表”**：这一模块主要包括三个类别，即全部商品、已上架商品和未上架商品，进行分类的依据是商品的商品的状态。

在商品列表页面可以查看到商品的部分信息：商品编号、商品名称、商品价格、商品库存、商品销量、商品状态和上架时间，以及可以对商品进行更多的操作：查看详情、修改和上架/下架。

“详情”：可以查看到商品的全部详细信息，以及部分商品的相关评价，同时也提供了修改商品信息的入口。

“修改”：可以修改商品的库存和价格等部分信息。

“上架/下架”：通过修改商品的状态来控制商品是否在用户端显示。

**“订单管理”：**这部分主要根据订单的处理进度来将订单分为：待发货、待收货、待评价、已完成和全部订单。且每个模块都提供了订单的编号、状态、总件数、总金额、运单号（未发货订单此处为“暂未发货”提示占位符）、创建时间和最后更新时间等信息，同时针对不同模块提供了不同的更多操作选项，支持查看订单的更多相关信息。

“待发货”订单中管理员可以完成订单发货步骤，即填写运单号然后确认订单已发货来更新订单状态。

**“日志管理”：**日志管理模块主要包含三大部分：登陆日志、操作日志和系统日志，每个分模块都提供操作人姓名、操作时间、操作模块、操作结果和操作时间等相关信息。

“登陆日志”：记录管理员的登陆情况。

“操作日志”：记录管理员在后台的各个模块的操作情况。

“系统日志”：记录系统在运行时所发生的故障信息。

**4 系统设计**

**4.1 数据库概念结构设计**

图4.1.1 网上购物系统E-R图

**4.2 系统功能模块设计**



图3.1.1 网站总模块图



图3.1.2 网站用户端页面结构



图3.1.3 网站管理端页面结构



图3.1.4 网站功能结构

**4.3 数据库逻辑结构设计**

按照E-R图到逻辑关系模式的转换规则和根据系统的数据库分表优化，可得到系统有如下9个关系，其中带下划线的为关系关键字（即主码）。

（1） 用户信息（用户编号，姓名，电话，邮箱，状态，地址，昵称，密码，创建时间，最后登录时间）

（2） 管理员信息（管理员编号，姓名，电话，密码，权限）

（3） 商品种类信息（顺编号，种类名称）

（4） 商品信息（商品编号，名称，种类编号，价格，库存，图片，简介，详细介绍，销量，创建时间，更新时间，状态，上架时间）

（5） 订单信息（订单编号，用户编号，状态，商品数量，商品总价，运单号，创建时间，更新时间）

（6） 订单明细信息（订单明细编号，订单编号，商品编号，商品名称，商品价格，商品数量）

（7） 日志信息（顺序号，种类，管理者编号，具体操作，操作模块，操作结果，时间）

（8） 订单处理记录信息（顺序号，订单编号，管理者编号，更新后状态，时间）

（9） 评价信息（顺序号，订单明细编号，星级，内容，图片，时间）

注：

① 为增加的系统的响应速度和QPS，因此采用了适当数量的冗余字段，例如订单信息中商品数量和商品总价字段的设置是为了避免在仅需查询订单中商品的数量和总价时而多次遍历订单明细表而带来的响应延迟。

② 为维护订单内信息的不变性和可查询订单内相关产品的需要，在借鉴淘宝等大型网上在线购物平台的基础上，选择了在订单明细表中保存商品的名称和价格这样的必要字段来供用户在购买后进行回顾（因为管理者可能从后台对商品的价格等信息进行修改，所以需要将商品的重要信息进行存储），同时选择外联所购买的商品连接，用户可以通过外联在商家没有修改商品描述的情况下查看商品更多的详细信息。

③ 为完成商品的按种类进行分类的需求，因此我们创建商品种类信息冗余表用于存储商品的种类的信息，而商品信息中对此表中的属性进行外联，从而可以实现对商品的分类，也可以便于后续的管理者对分类名称的修改操作。

**4.4 数据库物理结构设计及其实现**

① 用户表（user），其属性对应的含义：编号，姓名，电话，邮箱，状态，地址，昵称，密码，创建时间，最后登录时间。

create table user

(

id int not null auto\_increment primary key,

name varchar(20) not null,

phone varchar(11) not null,

email varchar(100),

status varchar(20) not null,

address varchar(200) not null,

nickname varchar(50) not null,

password varchar(200) not null,

create\_time datetime,

last\_login datetime

) ENGINE = InnoDB AUTO\_INCREMENT=1 DEFAULT CHARACTER SET utf8;

② 管理员表（administrator），其属性对应的含义：编号，姓名，电话，密码，权限。

create table administrator

(

id int not null auto\_increment primary key,

name varchar(20) not null,

phone varchar(11) not null,

password varchar(200) not null,

permission int default 1 not null

) ENGINE = InnoDB AUTO\_INCREMENT=1 DEFAULT CHARACTER SET utf8;

③ 商品种类表（commodity\_category）：其属性对应的含义：编号，种类名称。

create table commodity\_category

(

id int not null auto\_increment PRIMARY KEY,

name varchar(20) not null

) ENGINE = InnoDB AUTO\_INCREMENT=1 DEFAULT CHARACTER SET utf8;

④ 商品表（commodity），其属性对应的含义：编号，名称，种类编号，价格，库存，图片，简介，详细介绍，销量，创建时间，更新时间，状态，上架时间。

create table commodity

(

id int not null auto\_increment PRIMARY KEY,

name varchar(50) not null,

type\_id int not null ,

price float not null ,

stock int not null,

image varchar(200),

introduction varchar(200),

detailed\_introduction text,

sales\_volume int default 0,

create\_time datetime,

update\_time datetime,

status varchar(10),

added\_time datetime

) ENGINE = InnoDB AUTO\_INCREMENT=1 DEFAULT CHARACTER SET utf8;

alter table commodity add constraint FK\_ID foreign key(type\_id) REFERENCES commodity\_category(id);

⑤ 订单表，其属性对应的含义：订单编号，用户编号，状态，商品数量，商品总价，运单号，创建时间，更新时间。

create table commodity\_order

(

id int not null auto\_increment primary key,

user\_id int not null,

status varchar(20) not null,

commodity\_quantity int not null,

total\_amount float not null,

waybill\_number varchar(100),

create\_time datetime,

update\_time datetime

) ENGINE = InnoDB AUTO\_INCREMENT=1 DEFAULT CHARACTER SET utf8;

alter table commodity\_order add constraint FK\_ID\_1 foreign key(user\_id) REFERENCES user(id);

⑥ 订单明细表，其属性对应的含义：编号，订单编号，商品编号，商品名称，商品价格，商品数量。

create table commodity\_order\_detail

(

id int not null auto\_increment primary key,

order\_id int not null,

commodity\_id int not null,

commodity\_name varchar(50) not null,

commodity\_price float not null,

commodity\_quantity int not null

) ENGINE = InnoDB AUTO\_INCREMENT=1 DEFAULT CHARACTER SET utf8;

alter table commodity\_order\_detail add constraint FK\_ID\_2 foreign key(order\_id) REFERENCES commodity\_order(id);

alter table commodity\_order\_detail add constraint FK\_ID\_3 foreign key(commodity\_id) REFERENCES commodity(id);

⑦ 日志表，其属性对应的含义：顺序号，种类，管理者编号，具体操作，操作模块，操作结果，时间。

create table log

(

id int not null auto\_increment primary key,

type varchar(20) not null,

operator\_id int not null,

operating varchar(30) not null,

module varchar(20) not null,

result varchar(100) not null,

time datetime not null

) ENGINE = InnoDB DEFAULT CHARACTER SET utf8;

alter table log add constraint FK\_ID\_4 foreign key(operator\_id) REFERENCES administrator (id);

⑧ 订单处理记录表，其属性对应的含义：顺序号，订单编号，管理者编号，更新后状态，时间。

create table order\_processing\_record

(

id int not null auto\_increment primary key,

order\_id int not null,

operator\_id int not null,

status varchar(20) not null,

time datetime

) ENGINE = InnoDB DEFAULT CHARACTER SET utf8;

alter table order\_processing\_record add constraint FK\_ID\_5 foreign key(order\_id) REFERENCES commodity\_order(id);

alter table order\_processing\_record add constraint FK\_ID\_6 foreign key(operator\_id) REFERENCES administrator (id);

⑨ 评价表，其属性对应的含义：顺序号，订单明细编号，星级，内容，图片，时间。

create table evaluation

(

id int not null auto\_increment primary key,

order\_detail\_id int not null,

grade int,

content text,

image varchar(200),

time datetime

) ENGINE = InnoDB AUTO\_INCREMENT=1 DEFAULT CHARACTER SET utf8;

alter table evaluation add constraint FK\_ID\_7 foreign key(order\_detail\_id) REFERENCES commodity\_order\_detail(id);

**5 系统实现**

**5.1 编写目的**

本说明书目的在于明确说明系统各功能的实现方式，指导开发员进行编码。

本说明书的预期读者为：客户项目经理、开发项目经理、测试项目经理、系统测试人员、系统开发人员等。

**5.2 背景**

待开发软件系统的名称：易购网上购物子系统

此软件系统任务开发者：杨帆

此软件系统任务用户：有网上购物需求的群体

**5.3 基线**

易购网上购物系统需求分析说明书 v1.0

**5.4 范围**

系统包括的范围：网上购物系统用户端和后台数据管理

**5.5 模块命名规则**

1. 局部变量应尽量易懂简洁；

2、禁止使用汉语拼音来命名；

3、在代码中尽量不用具体的大小数值，定义成宏，便于以后维护；

4、变量在定义时赋初值，类析构时或程序退出时判断释放所有变量。

**5.6 模块汇总**

**5.6.1模块汇总表**

***提示：****这里模块是指相对独立的软件设计单元，例如对象类、函数包等等。*

|  |  |
| --- | --- |
| **子系统A** | |
| 模块名称 | 功能简述 |
| 网上购物系统用户端 | 主要提供用户登录、注册和找回密码，即用户个人信息的相关维护操作，以及提供了用户浏览选购商品的入口，用户可以通过此模块完成商品购买的全部流程操作。 |
| **子系统B** |  |
| 模块名称 | 功能简述 |
| 后台管理——人员模块 | 采后台管理子模块，实现人员信息的查看，人员信息的相关修改操作。 |
| 后台管理——商品管理模块 | 后台管理的子模块，实现商品的信息查看、信息修改以及状态更新等操作。 |
| 后台管理——订单管理模块 | 后台管理的子模块，实现订单的查看以及订单状态的更新。 |
| 后台管理——商品上架 | 后台管理的子模块，实现商品的上架功能。 |
| 购物交易模块 | 主要进行商品的购买交易操作。 |

**5.6.2模块关系图**

商品交易

人员管理

商品上架

商品管理

订单管理

**5.7子系统模块设计**

**5.7.1 模块1 网上购物系统用户端**

|  |  |
| --- | --- |
| 模块名称 | *网上购物系统用户端* |
| 功能描述 | 用户进入网站后，可以选择以游客的身份来浏览和选购商品，可以以游客的身份将选购的商品加入到购物车里，但当结算时用户必须选择进行账号登陆。  如果用户尚未注册，则引导用户进行账号的注册，如果用户已经注册了账号则直接引导用户登陆账号，当用户连续五次输入错误密码后系统会自动冻结账号，禁止用户进行登陆操作。  如果用户忘记密码，则提供了找回密码的服务，可以通过密保问题或者邮箱找回的方式来找回密码。  当用户登陆后可以在网站的首页的商品列表中选购商品，同时根据结算方式的不同，又可以分为“立即购买”——立即支付，选择此操作则会直接跳转至支付界面进行支付，这时默认是选择一件该商品进行结算。如果用户选择的是添加到购物车，则将当前该物品按选择的数量加入到购物车中，此时购物车中商品的订单状态为待结算状态，当用户选购完全部商品时，则可以进入到购物车界面查看已选购商品的具体数量和当前购物车内商品的总件数和总金额等一些信息，并可以选择进行商品结算，此后会跳转至支付界面对已选购的商品进行支付。  当商品支付后，订单会交由后台管理员操作，当管理员更新订单状态为已发货——待收货状态时，用户在收到商品后可以进行确认收货的操作，然后对订单中的商品进行评价，当用户完成全部评价后，则该笔订单的状态更新为已完成状态。  当用户登陆后，用户可以在个人中心页面内查看自己的个人信息，并可以进行相关的信息(如地址等)修改。  还可以对当前自己的订单进行管理，这个管理既包括查看订单的相关详细信息，即购买的商品名称、数量、商品的总价和商品的总件数等信息，还有订单的处理记录，即每次状态更新的时间。还包括对订单进行相关状态的更新，比如收货后的确认收货状态等。 |
| 接口与属性 | // 订单和购物车相关操作接口  public interface OrderBiz {  // 添加订单  void add(CommodityOrder commodityOrder);  // 修改订单  void edit(CommodityOrder commodityOrder);  // 订单发货  void deliver(Integer orderId, String waybillNumber);  // 立即购买  Integer buyNow(CommodityOrderDetail commodityOrderDetail, Integer userId);  // 加入购物车  void addToShoppingCart(CommodityOrderDetail commodityOrderDetail, Integer userId);  // 结算订单  CommodityOrder settlement(Integer userId);  // 支付订单  void pay(Integer orderId);  // 确认收货  void confirmReceipt(Integer orderId);  // 已完成  void completed(Integer orderId);  // 删除订单  void remove(Integer id);  // 增加购物车中商品的数量  void increaseCommodityQuantity(Integer userId, Integer commodityId);  // 减少购物车中商品的数量  void decreaseCommodityQuantity(Integer userId, Integer commodityId);  // 移除购物车中的商品  void removeCommodity(Integer userId, Integer commodityId);  // 根据订单号获取订单  CommodityOrder get(Integer id);  // 获取购物车  CommodityOrder getCart(Integer userId);  // 清空购物车  void clearCart(Integer userId);  // 通过订单明细获取商品编号  String getCommodityNameByOrderDetail(Integer id);  // 通过订单状态获取订单列表  List<CommodityOrder> getOrderListByStatus(String status);  // 获取订单列表  List<CommodityOrder> getOrderList(Integer userId, String status);  // 通过用户编号获取订单列表  List<CommodityOrder> getByUserId(Integer userId);  // 获取商品的销售订单  List<CommodityOrderDetail> getCommoditySalesRecord(Integer commodityId);  // 获取所有订单  List<CommodityOrder> getAll();  }  // 用户相关操作接口  public interface UserBiz {  // 添加用户  void add(User user);  // 修改用户信息  void edit(User user);  // 修改密码  void changePassword(String phone, String password);  // 修改状态  void changeStatus(Integer id, String status);  // 修改个人信息  User verifyInfo(String phone, String password);  // 删除用户  void remove(Integer id);  // 通过用户编号获取用户信息  User get(Integer id);  // 通过用户编号从缓存中获取用户信息  User getDetailFromCache(Integer id);  // 通过认证获取用户列表  List<User> getByIdentity(String identity);  // 获取所有用户列表  List<User> getAll();  }  // 评论相关操作接口  public interface EvaluationBiz {  // 添加评论  void add(Evaluation evaluation);  // 通过订单明细编号查找评论  List<Evaluation> getByOrderId(Integer id);  // 通过商品编号查找评论  List<Evaluation> getByCommodityId(Integer id);  } |
| 数据结构  与算法 | *// 订单类*  *public class CommodityOrder {*  *// 订单编号*  *private Integer id;*  *// 用户编号*  *private Integer userId;*  *// 订单状态*  *private String status;*  *// 商品总数量*  *private Integer commodityQuantity;*  *// 商品总金额*  *private Float totalAmount;*  *// 运单号*  *private String waybillNumber;*  *// 创建时间*  *private Date createTime;*  *// 更新时间*  *private Date updateTime;*  *// 订单明细列表*  *private List<CommodityOrderDetail> detailList;*  *… 省略相关get/set方法*  *}*  *// 订单明细类*  *public class CommodityOrderDetail {*  *// 订单明细编号*  *private Integer id;*  *// 订单编号*  *private Integer orderId;*  *// 商品编号*  *private Integer commodityId;*  *// 商品名称*  *private String commodityName;*  *// 商品价格*  *private Float commodityPrice;*  *// 商品数量*  *private Integer commodityQuantity;*  *// 商品实体*  *private Commodity commodity;*  *// 订单实体*  *private CommodityOrder commodityOrder;*  *… 省略相关get/set方法*  *}*  *// 评价类*  *public class Evaluation {*  *// 评价编号*  *private Integer id;*  *// 订单明细编号*  *private Integer orderDetailId;*  *// 评价星级*  *private Integer grade;*  *// 评价内容*  *private String content;*  *// 评价图片*  *private String image;*  *// 评价时间*  *private Date time;*  *// 用户昵称*  *private String userName;*  *… 省略相关get/set方法*  *}*  *// 用户类*  *public class User {*  *// 用户编号*  *private Integer id;*  *// 用户姓名*  *private String name;*  *// 用户电话*  *private String phone;*  *// 用户邮箱*  *private String email;*  *// 用户状态*  *private String status;*  *// 用户地址*  *private String address;*  *// 用户昵称*  *private String nickname;*  *// 用户密码*  *private String password;*  *// 创建时间*  *private Date createTime;*  *// 最后登录时间*  *private Date lastLogin;*  *… 省略相关get/set方法*  *}* |
| 补充说明 |  |
| 流程图 |  |

**5.7.2 模块2 系统后台管理——人员模块**

|  |  |
| --- | --- |
| 模块名称 | *人员管理* |
| 功能描述 | 此模块属于后台管理模块中的子模块，其主要的功能是对人员进行信息查看和管理，同时人员的类别又分为管理人员和用户。  当管理员需要进入此模块的时候需要首先登陆后台管理系统，后台管理系统具有自动管理用户验证的安全机制，即当连续输入错误五次密码后，此管理员的账号将被冻结，暂时不能够再进行登陆后台等操作。  人员管理模块中的管理员管理子模块，主要提供了查看管理员的相关信息，比如管理员姓名和电话等信息，以及其相关操作的操作日志和登录日志，同时还可以在此模块中对管理员的权限进行更改。  人员管理模块中的用户管理子模块，主要提供了查看用户的相关信息，比如姓名、电话、地址等个人信息，以及其相关的订单信息，并可以对其相关的订单进行操作，同时还可以在此模块中对用户进行锁定/解锁和删除/恢复等相关操作。 |
| 接口与属性 | // 用户相关操作接口  public interface UserBiz {  // 添加用户  void add(User user);  // 修改用户信息  void edit(User user);  // 修改密码  void changePassword(String phone, String password);  // 修改状态  void changeStatus(Integer id, String status);  // 修改个人信息  User verifyInfo(String phone, String password);  // 删除用户  void remove(Integer id);  // 通过用户编号获取用户信息  User get(Integer id);  // 通过用户编号从缓存中获取用户信息  User getDetailFromCache(Integer id);  // 通过认证获取用户列表  List<User> getByIdentity(String identity);  // 获取所有用户列表  List<User> getAll();  }  // 管理员相关操作接口  public interface AdminBiz {  // 添加管理员  void add(Admin admin);  // 修改管理员信息  void edit(Admin admin);  // 修改密码  void changePassword(String phone, String password);  // 修改状态  void changeStatus(Integer id, String status);  // 修改个人信息  Admin verifyInfo(String phone, String password);  // 删除管理员  void remove(Integer id);  // 通过管理员编号获取管理员信息  Admin get(Integer id);  // 通过管理员编号从缓存中获取管理员信息  Admin getDetailFromCache(Integer id);  // 通过认证获取管理员列表  List<Admin> getByIdentity(String identity);  // 获取所有管理员列表  List<Admin> getAll();  } |
| 数据结构  与算法 | *public class User {*  *// 用户编号*  *private Integer id;*  *// 用户姓名*  *private String name;*  *// 用户电话*  *private String phone;*  *// 用户邮箱*  *private String email;*  *// 用户状态*  *private String status;*  *// 用户地址*  *private String address;*  *// 用户昵称*  *private String nickname;*  *// 用户密码*  *private String password;*  *// 创建时间*  *private Date createTime;*  *// 最后登录时间*  *private Date lastLogin;*  *… 省略相关get/set方法*  *}*  *public class Admin {*  *// 管理员编号*  *private Integer id;*  *// 管理员姓名*  *private String name;*  *// 管理员电话*  *private String phone;*  *// 用户状态*  *private String status;*  *// 用户权限*  *private Integer permission;*  *// 用户昵称*  *private String nickname;*  *// 管理员密码*  *private String password;*  *// 创建时间*  *private Date createTime;*  *// 最后登录时间*  *private Date lastLogin;*  *… 省略相关get/set方法*  *}* |
| 补充说明 |  |
| 流程图 |  |

**5.7.3 模块3 系统后台管理——商品模块**

|  |  |
| --- | --- |
| 模块名称 | *商品管理* |
| 功能描述 | 商品管理模块位于系统后台管理模块中，属于其子模块，其主要功能是查看商品的相关信息并进行相关的商品操作。  首先是管理员登陆系统的相关的逻辑，同上一个人员管理模块类似，所有的后台管理登陆模块都提供了登陆验证功能，如果连续输入错误五次密码，则账号将会被系统自动冻结，禁止操作。  商品管理模块主要包含两个子模块，一个是查看商品信息，另一个是操作商品。  通过进入到商品列表页面，可以查看不同类别的商品，同时可以进一步进入到商品详情页面查看商品的详细信息，例如商品的名称、商品价格、商品库存、商品简介等信息。  随后可以通过列表页面为每个商品提供修改信息入口，进入到修改商品信息和状态的子页面，可修改的具体的商品信息包括库存和价格等次要信息，而对于商品状态的修改主要是通过修改商品的状态来实现商品的上架和下架操作。  对商品的管理还提供了创建商品模块，可以通过此模块填写要新创建的商品信息，例如商品名称、商品价格、商品库存和商品简介等信息，之后在选择立即上架或者定时上架后对商品进行上架操作。 |
| 接口与属性 | // 商品接口  public interface CommodityBiz {  // 添加商品  void add(Commodity commodity);  // 修改商品  void edit(Commodity commodity);  // 修改商品状态  void changeStatus(Integer id);  // 移除商品  void remove(Integer id);  // 根据商品编号获取商品  Commodity get(Integer id);  // 获取每种商品的销量前三位商品  List<Commodity> getEveryTypeTopThree();  // 通过商品的类型类获取商品列表  List<Commodity> getByType(Integer typeId);  // 通过商品的状态来获取商品列表  List<Commodity> getByStatus(String status);  // 从缓存中获取商品  Commodity getDetailFromCache(Integer id);  // 获取全部商品  List<Commodity> getAll();  }  // 商品种类接口  public interface CommodityCategoryBiz {  // 添加商品种类  void add(CommodityCategory commodityCategory);  // 编辑商品种类  void edit(CommodityCategory commodityCategory);  // 移除商品种类  void remove(Integer id);  // 通过商品种类编号获取商品种类信息  CommodityCategory get(Integer id);  // 获取所有商品种类  List<CommodityCategory> getAll();  } |
| 数据结构  与算法 | *// 商品类*  *public class Commodity {*  *// 商品编号*  *private Integer id;*  *// 商品库存*  *private Integer stock;*  *// 商品种类编号*  *private Integer typeId;*  *// 商品销售量*  *private Integer salesVolume;*  *// 商品价格*  *private Float price;*  *// 商品名称*  *private String name;*  *// 商品种类*  *private String type;*  *// 商品图片*  *private String image;*  *// 商品状态*  *private String status;*  *// 商品介绍*  *private String introduction;*  *// 商品详细介绍*  *private String detailedIntroduction;*  *// 商品上架时间*  *private Date addedTime;*  *// 商品创建时间*  *private Date createTime;*  *// 商品最后更新时间*  *private Date updateTime;*  *… 省略相关get/set方法*  *}* |
| 补充说明 |  |
| 流程图 |  |

**5.7.4 模块4 系统后台管理——商品上架模块**

|  |  |
| --- | --- |
| 模块名称 | *商品上架* |
| 功能描述 | 商品上架模块是系统后台管理的子模块，主要执行的功能为商品上架功能。  首先是登陆时的安全验证机制，即连续五次输入错误密码则会直接冻结管理员账户，同时在成功登录和登陆失败的同时都会有对应的登陆日志记录，用于高层级管理员排查异常和事故。  然后进入到商品上架模块，填写商品的相关信息，如商品的名称、商品的价格、商品的库存、商品的简介和详细介绍等信息，最后选择商品的上架类型，即“立即上架”和“定时上架”。  “立即上架”后用户可以在第一时间浏览和选购该商品。  “定时上架”商品会在特定的时间更新商品状态，到特定的时间后用户即可以浏览和选购该商品。  以上的全部操作的相关信息都会记录在操作日志中，包括操作的时间、操作人员和操作的具体执行操作等信息。 |
| 接口与属性 | // 商品接口  public interface CommodityBiz {  // 添加商品  void add(Commodity commodity);  // 修改商品  void edit(Commodity commodity);  // 修改商品状态  void changeStatus(Integer id);  // 移除商品  void remove(Integer id);  // 根据商品编号获取商品  Commodity get(Integer id);  // 获取每种商品的销量前三位商品  List<Commodity> getEveryTypeTopThree();  // 通过商品的类型类获取商品列表  List<Commodity> getByType(Integer typeId);  // 通过商品的状态来获取商品列表  List<Commodity> getByStatus(String status);  // 从缓存中获取商品  Commodity getDetailFromCache(Integer id);  // 获取全部商品  List<Commodity> getAll();  }  // 商品种类接口  public interface CommodityCategoryBiz {  // 添加商品种类  void add(CommodityCategory commodityCategory);  // 编辑商品种类  void edit(CommodityCategory commodityCategory);  // 移除商品种类  void remove(Integer id);  // 通过商品种类编号获取商品种类信息  CommodityCategory get(Integer id);  // 获取所有商品种类  List<CommodityCategory> getAll();  }  // 日志类  public interface LogBiz {  // 记录系统日志  void addSystemLog(Log log);  // 记录登录日志  void addLoginLog(Log log);  // 记录操作日志  void addOperatingLog(Log log);  // 根据日志类型获取日志列表  List<Log> getLogByType(String type);  // 根据管理员编号获取日志列表  List<Log> getLogByUserId(Integer id);  } |
| 数据结构  与算法 | *// 商品类*  *public class Commodity {*  *// 商品编号*  *private Integer id;*  *// 商品库存*  *private Integer stock;*  *// 商品种类编号*  *private Integer typeId;*  *// 商品销售量*  *private Integer salesVolume;*  *// 商品价格*  *private Float price;*  *// 商品名称*  *private String name;*  *// 商品种类*  *private String type;*  *// 商品图片*  *private String image;*  *// 商品状态*  *private String status;*  *// 商品介绍*  *private String introduction;*  *// 商品详细介绍*  *private String detailedIntroduction;*  *// 商品上架时间*  *private Date addedTime;*  *// 商品创建时间*  *private Date createTime;*  *// 商品最后更新时间*  *private Date updateTime;*  *… 省略相关get/set方法*  *}*  *// 日志类*  *public class Log {*  *// 日志类型*  *private String type;*  *// 管理员编号*  *private Integer operatorId;*  *// 管理员姓名*  *private String operator;*  *// 具体操作*  *private String operating;*  *// 操作模块*  *private String module;*  *// 操作结果*  *private String result;*  *// 操作结果*  *private Date time;*  *… 省略相关get/set方法*  *}* |
| 补充说明 |  |
| 流程图 |  |

**5.7.5 模块5 交易模块**

|  |  |
| --- | --- |
| 模块名称 | *交易模块* |
| 功能描述 | 交易模块主要是指交易的全流程操作的模块，主要包括商品交易的选购、结算、支付、发货、收货、评价和已完成七种进度状态。  首先是用户端通过系统的用户端浏览和选购商品，如果选择立即购买则会直接默认选中一个该种商品跳转支付界面，而如果选择添加购物车则会默认添加至购物车中，此时的商品为待结算状态。  当用户从购物车中选择支付订单时则会跳转到支付页面，在确认商品后即可对商品订单进行支付操作。  当用户支付完成后，订单的状态即变为待发货状态，此时管理员可以对订单进行发货操作。具体的发货操作应为管理员填写订单号，同时更新订单的状态为已发货状态。  管理员发货后，订单状态待收货，此时用户在收到商品后即可选择确认收货操作，然后对订单中的商品进行逐项的评价。评价的操作包括对商品的星级评价，图片评价和文字评价。当用户评价完商品后，管理员和其他用户可以从商品的相关信息中查看到该条评价，包括这条评价的用户和产生时间。  当用户完成评价后，即整个交易完成，订单状态更新为已完成。 |
| 接口与属性 | // 订单处理记录接口  public interface OrderProcessingRecordBiz {  // 添加支付记录  void addPayRecord(OrderProcessingRecord orderProcessingRecord);  // 添加发货记录  void addDeliverRecord(OrderProcessingRecord orderProcessingRecord);  // 添加确认收货记录  void addConfirmReceiptRecord(OrderProcessingRecord orderProcessingRecord);  // 添加评论记录  void addCommentOrderRecord(OrderProcessingRecord orderProcessingRecord);  // 根据订单明细编号查找订单处理流程记录列表  List<OrderProcessingRecord> getOrderProcessingRecord(Integer orderId);  } |
| 数据结构  与算法 | *// 订单处理记录类*  *public class OrderProcessingRecord {*  *// 订单详情编号*  *private Integer orderId;*  *// 操作人编号*  *private Integer operatorId;*  *// 订单更新状态*  *private String status;*  *// 操作时间*  *private Date time;*  *// 操作人姓名*  *private String operator;*  *… 省略相关get/set方法*  *}* |
| 补充说明 |  |
| 流程图 |  |

**5.7.6 模块6 系统后台管理——订单管理模块**

|  |  |
| --- | --- |
| 模块名称 | *订单管理* |
| 功能描述 | 订单管理模块为系统后台管理的子模块，其主要功能是对订单进行管理，其主要的子模块为查看订单和更新订单状态等。  与其它后台管理模块类似，它也具备安全验证机制，即当管理员连续输入错误五次密码后，系统将自动冻结该管理员账号。  此模块可以根据订单处理进度和用户来查看不同的订单，并可以查看到订单的详细处理流程记录。  同时可以对订单进行更新操作，具体的操作为当订单的状态为待发货时，管理员可以填写订单的运单号，并确认已发货 |
| 接口与属性 | // 订单接口  public interface OrderBiz {  // 添加订单  void add(CommodityOrder commodityOrder);  // 修改订单  void edit(CommodityOrder commodityOrder);  // 订单发货  void deliver(Integer orderId, String waybillNumber);  // 立即购买  Integer buyNow(CommodityOrderDetail commodityOrderDetail, Integer userId);  // 加入购物车  void addToShoppingCart(CommodityOrderDetail commodityOrderDetail, Integer userId);  // 结算订单  CommodityOrder settlement(Integer userId);  // 支付订单  void pay(Integer orderId);  // 确认收货  void confirmReceipt(Integer orderId);  // 已完成  void completed(Integer orderId);  // 删除订单  void remove(Integer id);  // 增加购物车中商品的数量  void increaseCommodityQuantity(Integer userId, Integer commodityId);  // 减少购物车中商品的数量  void decreaseCommodityQuantity(Integer userId, Integer commodityId);  // 移除购物车中的商品  void removeCommodity(Integer userId, Integer commodityId);  // 根据订单号获取订单  CommodityOrder get(Integer id);  // 获取购物车  CommodityOrder getCart(Integer userId);  // 清空购物车  void clearCart(Integer userId);  // 通过订单明细获取商品编号  String getCommodityNameByOrderDetail(Integer id);  // 通过订单状态获取订单列表  List<CommodityOrder> getOrderListByStatus(String status);  // 获取订单列表  List<CommodityOrder> getOrderList(Integer userId, String status);  // 通过用户编号获取订单列表  List<CommodityOrder> getByUserId(Integer userId);  // 获取商品的销售订单  List<CommodityOrderDetail> getCommoditySalesRecord(Integer commodityId);  // 获取所有订单  List<CommodityOrder> getAll();  }  // 订单处理记录接口  public interface OrderProcessingRecordBiz {  // 添加支付记录  void addPayRecord(OrderProcessingRecord orderProcessingRecord);  // 添加发货记录  void addDeliverRecord(OrderProcessingRecord orderProcessingRecord);  // 添加确认收货记录  void addConfirmReceiptRecord(OrderProcessingRecord orderProcessingRecord);  // 添加评论记录  void addCommentOrderRecord(OrderProcessingRecord orderProcessingRecord);  // 根据订单明细编号查找订单处理流程记录列表  List<OrderProcessingRecord> getOrderProcessingRecord(Integer orderId);  } |
| 数据结构  与算法 | *// 订单类*  *public class CommodityOrder {*  *// 订单编号*  *private Integer id;*  *// 用户编号*  *private Integer userId;*  *// 订单状态*  *private String status;*  *// 商品总数量*  *private Integer commodityQuantity;*  *// 商品总金额*  *private Float totalAmount;*  *// 运单号*  *private String waybillNumber;*  *// 创建时间*  *private Date createTime;*  *// 更新时间*  *private Date updateTime;*  *// 订单明细列表*  *private List<CommodityOrderDetail> detailList;*  *… 省略相关get/set方法*  *}*  *// 订单明细类*  *public class CommodityOrderDetail {*  *// 订单明细编号*  *private Integer id;*  *// 订单编号*  *private Integer orderId;*  *// 商品编号*  *private Integer commodityId;*  *// 商品名称*  *private String commodityName;*  *// 商品价格*  *private Float commodityPrice;*  *// 商品数量*  *private Integer commodityQuantity;*  *// 商品实体*  *private Commodity commodity;*  *// 订单实体*  *private CommodityOrder commodityOrder;*  *… 省略相关get/set方法*  *}*  *// 订单处理记录类*  *public class OrderProcessingRecord {*  *// 订单详情编号*  *private Integer orderId;*  *// 操作人编号*  *private Integer operatorId;*  *// 订单更新状态*  *private String status;*  *// 操作时间*  *private Date time;*  *// 操作人姓名*  *private String operator;*  *… 省略相关get/set方法*  *}* |
| 补充说明 |  |
| 流程图 |  |

**6 系统测试**

**6.1 测试方式**

因为本系统的体积过于庞大，所以系统测试大多是在开发过程中进行的，而且大多数都采用了手动操作测试的方式来进行。

**6.2 测试结果**

在开发的过程中有遇到过一些系统问题，但是在一段时间的DEBUG之后本系统当前无BUG运行正常，各个模块的功能均可以正常的运行，且能够达到预期的运行效果。

**7 体会与自我评价**

**7.1 技能提升**

通过这次课程设计的实践，自己加深了对于之前很多知识的理解，这次的实践是对自己之前所学习过的知识的一次总结和应用，同时也让自己学习到了更多新的技术。在这次课程设计中，自己独立完成了数据库系统的需求分析、数据库概念结构设计、数据库逻辑结构设计、数据库物理结构设计等一些流程的开发，大幅度的提升了自己的编程能力、独立思考能力、软件开发能力等一些综合素质。

在系统的开发中，我首次实践了之前所学习过的Spring + SpringMVC + MyBatis组合框架来进行系统的开发，在这套开发模式中，Spring提供了IOC的系统容器，SpringMVC提供了结构更加清晰、更易维护的MVC开发架构模式，MyBatis提供了封装JDBC后更易操作的数据库操作方式。

在之前的学习过程中，自己更多的就是读一些相关的资料和书籍，或者做一些小demo来进行实践，但其实在这个过程中自己对这个技术的掌握是不到位的，大多数情况下是仅仅限于了解的，古话说得好“纸上得来终觉浅，绝知此事要躬行”，实践是获得真知的唯一途径，许多的技术只有当你真正的亲身去实践过了，才会有更加深入的理解。因此这次的系统开发给了我一个很好的机会对其进行实践，而且对于这样较大规模的系统开发，在开发过程中会遇到许多在做小demo中没有遇到过的问题，在不断的遇到问题，解决问题的过程中使自己对于这项的技术又更深了一步。

同时在这次的系统的开发中，我也通过SpringMVC来使用了MVC的这种架构模式，其实之前在做安卓开发的时候，也有了解过MVC、MVP和MVVM这三种开发模式，并在一个较大规模的安卓开发过程中实践使用了MVP这种架构模式，并在小程序的开发过程中实践过MVVM这种架构模式，但是对于后端开发，使用MVC这种架构模式来进行开发还是第一次，因此这次的系统开发也是自己对于MVC架构模式的又一次深入探索与学习。

在这个系统的开发过程中，我是秉承一种大规模的系统开发的思想来进行开发，并集中的考虑了关于并发问题、容灾问题和数据读并发读写性能等一些高并发场景下的问题。在购物车的设计过程中，我的第一个版本是使用数据库来直接进行存储的，每次购物车状态的更新，我都会同步到数据库。但是在后来的思考和资料查询的过程中，我发现这样的方式在高并发的场景下存在着巨大的性能瓶颈，因此我选择了使用Redis内存来进行购物车的数据存储，并在特定的时间来使用单独的工作线程将Redis内存中数据同步至数据来进行持久化存储，保证用户在每次退出登录后再重新登录时可以再次获取到自己数据库中的数据，同时也保证了当主机发生了意外关闭或其他重启现象而清空内存时，用户的购物车信息也能够被持久化存储至数据库中。

针对商品列表和商品和商品详情这一系列页面的加载，我也采用了Spring集成Redis的方式来进行优化，通过测试可以发现没有优化之前，商品的信息直接从数据进行获取时每次操作的时间大概在500ms左右，而这还只是在数据量较少的情况下，那如果当数据量高达至上万时，这样操作的延迟会给用户带来很不好的体验。但是当我使用Redis内存进行优化后，成功的将这一操作的延迟降低到10ms~40ms性能优化进百分之几百，页面的相应速度明显得到大幅度的提升。

图7.1.1

上图即该操作的整个流程，通过Spring对于Redis的集成，我们在每次需要获取商品信息之前都先到Redis内存中进行查找，如果Redis内存中存在的话则直接返回数据，不存在时我们就到数据库中去进行查询，并将查询后的数据返回同时保存至Redis内存中，当管理员在后台进行修改商品信息的操作时，我们就同步的更新Redis内存中的数据。

我之所以想到这种方式来对数据存储进行优化，其主要原因就是对于商品详情这个数据查询的次数是明显的多于修改的次数的，所以我们可以将其保存至Redis内存中，以对内存的读取来代替对于文件的读取。

对于商品的定时上架功能，我采用了Spring集成Quartz的方式来进行实现，并通过管理员设置的上架时间来动态的生成CronExpression表达式，并将生成的任务动态的添加至Scheduler中进行管理。而我之所以选择Quartz而没有选择Timer的主要考虑是，Quartz相对来说使用更加的灵活，同时它是以多线程并发的方式来进行任务的执行，也就是当多个任务同时需要执行时，他会采用多开多个线程的方式来并发的进行执行，而对于Timer则是单线程的来进行的，会出现阻塞操作。因此使用Quartz可以大幅度的提升系统的性能。

同时对于用户注册和图片文件上传等一些可以进行异步操作的功能模块，我都使用RabbitMQ消息队列方式来进行异步的执行，通过这样的方式可以大幅度的提升系统的性能同时提升用户的使用体验。

**7.2 能力提升**

通过这次的系统开发，我觉得对于我最大的提升除了技术方面，更多的是在能力方面的提升。在开发的过程中，不断的遇到问题，遇到挫折，解决问题，收获成功的喜悦。在开发的过程中养成了独立思考、发现问题和解决问题的能力。

在开发的过程中，无数次的在漆黑的夜晚遇到莫名其妙的BUG，开始的时候整个人都会很焦躁不安，很惶恐，手足无措不知道怎么去解决它，但是随着开发的进行，经验的积累，自己解决问题的能力逐渐上升，DEBUG的能力也不断的提升，遇到问题的时候更加的冷静，能够在更短的时间里定位问题、发现问题并在思考后解决问题，我觉得这才是对于最大的影响。

同时在开发的过程中，随着自己学习能力的不断提升，面对新的技术时有了更好的学习方法，学习的效率也越来越高。