**概要设计规约**

**StarGames平台软件概要设计规约（说明书）**

目录

[1.  **引言** 3](#_Toc123677797)

[1.1  **概要设计依据** 3](#_Toc123677798)

[1.2  **参考资料** 3](#_Toc123677799)

[1.3  **假定和约束** 3](#_Toc123677800)

[2. **概要设计** 3](#_Toc123677801)

[2.1 **系统总体架构设计** 3](#_Toc123677802)

[2.2 **系统软件结构设计** 4](#_Toc123677803)

[2.3 **接口设计** 4](#_Toc123677804)

[2.3.1 **登录注册接口** 4](#_Toc123677805)

[2.3.2 **管理交易接口** 5](#_Toc123677806)

[2.3.3 **个人中心接口** 6](#_Toc123677807)

[2.3.4 **游戏库管理接口** 6](#_Toc123677808)

[2.3.5 **游戏发布接口** 7](#_Toc123677809)

[2.3.6 **信息管理接口** 7](#_Toc123677810)

[2.3.7 **游戏管理接口** 8](#_Toc123677811)

[2.3.8 **好友管理接口** 9](#_Toc123677812)

[2.4 **界面设计** 10](#_Toc123677813)

[2.4.1 **登录界面** 10](#_Toc123677814)

[2.4.2 **注册界面** 10](#_Toc123677815)

[2.4.3 **首页界面** 11](#_Toc123677816)

[2.4.4 **游戏库界面** 12](#_Toc123677817)

[2.4.5 **愿望单界面** 12](#_Toc123677818)

[2.4.6 **购物车界面** 13](#_Toc123677819)

[2.4.7 **搜索界面** 13](#_Toc123677820)

[2.4.8 **个人信息界面** 13](#_Toc123677821)

[2.4.9 **好友界面** 14](#_Toc123677822)

[2.4.10 **发行商信息管理界面** 14](#_Toc123677823)

[2.4.11 **发行商发布新游戏界面** 15](#_Toc123677824)

[2.4.12 **管理员封禁用户界面** 15](#_Toc123677825)

[2.4.13 **管理员下架游戏界面** 15](#_Toc123677826)

[2.5 **数据库设计** 16](#_Toc123677827)

[2.5.1 **数据库逻辑设计** 16](#_Toc123677828)

[2.5.2 **数据库物理设计** 20](#_Toc123677829)

[2.5.3 **数据库设计考虑** 27](#_Toc123677830)

[2.6 **系统出错处理设计** 28](#_Toc123677831)

[2.6.1 **出错信息** 28](#_Toc123677832)

[2.6.2 **补救措施** 31](#_Toc123677833)

[2.6.3 **系统维护设计** 31](#_Toc123677834)

1.  **引言**

1.1  **概要设计依据**

a．杜庆峰老师上课所讲的概要设计基本原则；

b. StarGames-需求分析规约

1.2  **参考资料**

1. 《SpringBoot 更好的开发》

2. 《HTML5+CSS3 从入门到精通》

3. 《更好的软件架构，更好的设计》

4. 《Design Pattern》

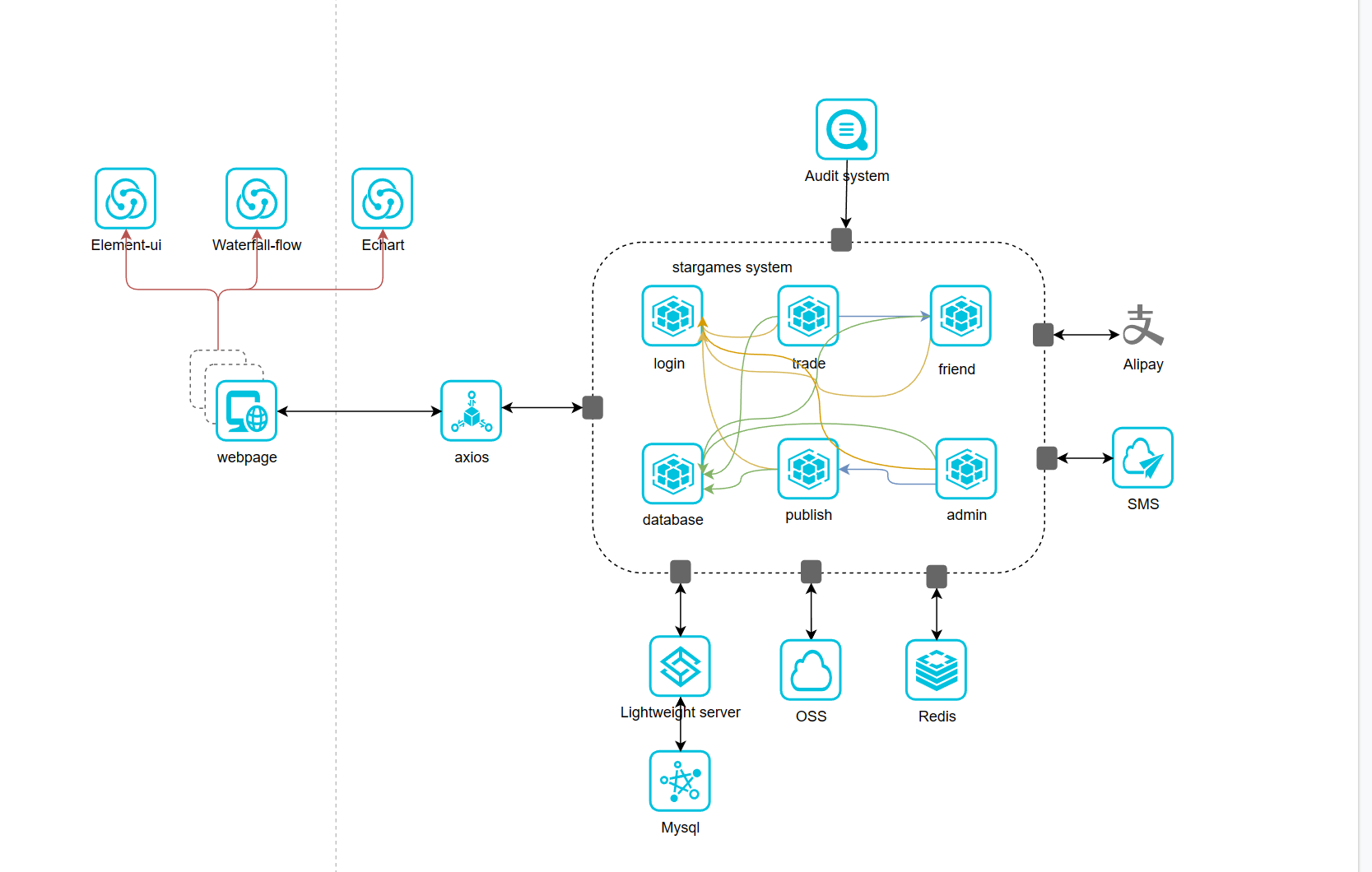
1.3  **假定和约束**

本项目开发主要受软件工程课程的约束，因此本项目的假定和约束如下所示：

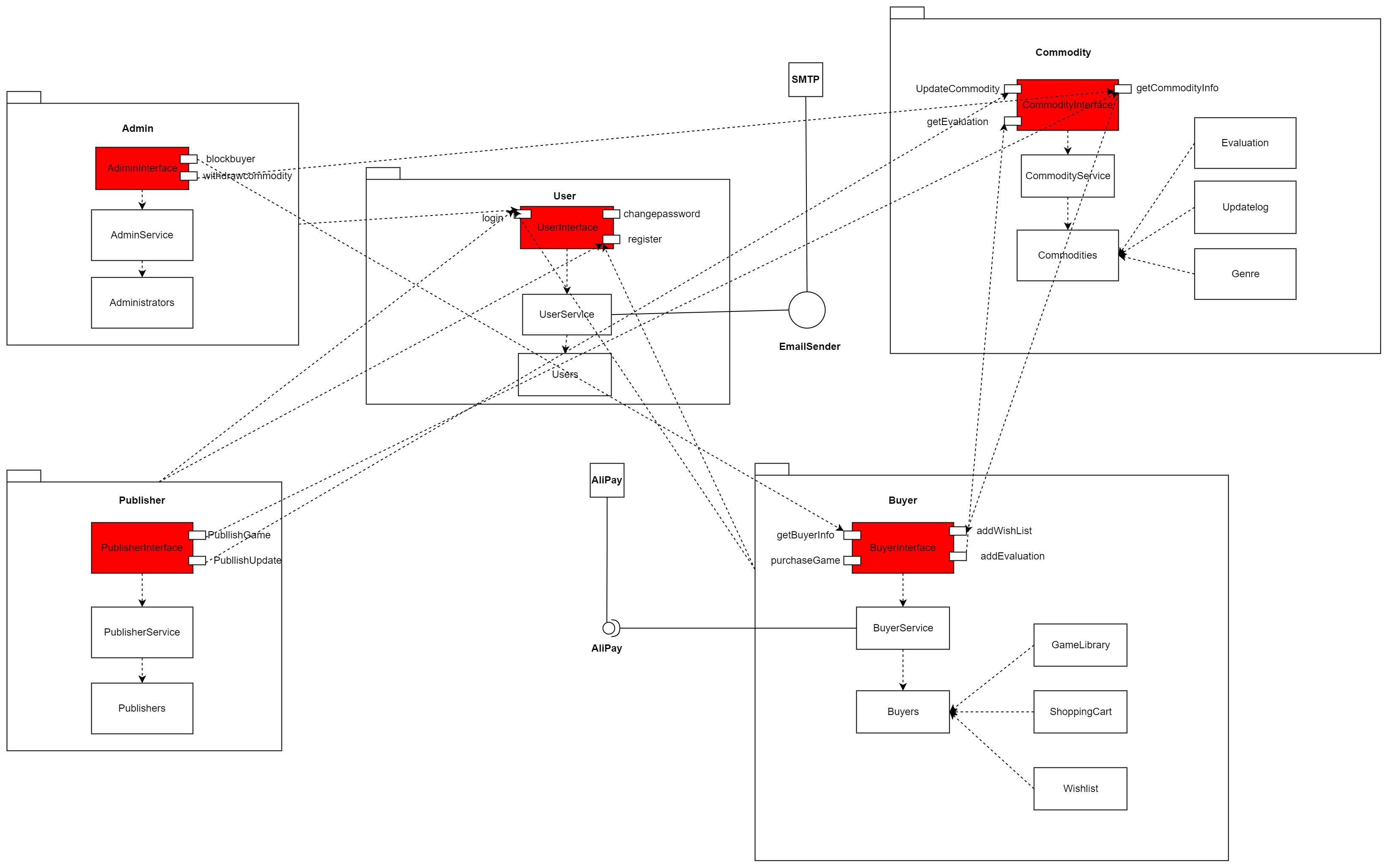
1. 项目开发期限为2个月，时间为2022年11月~12月；
2. 项目开发无经费，设备条件为4台Windows操作系统电脑以及腾讯云云平台等；
3. 项目在开发前通过线上问卷调研的方式收集了187份问卷，并据此我们制定了用户画像；
4. 在交流过程中，我们每五天线下汇报工作进度，同时通过Github进行代码协作管理。

2. **概要设计**

2.1 **系统总体架构设计**



2.2 **系统软件结构设计**



2.3 **接口设计**

2.3.1 **登录注册接口**

2.3.1.1 **login**

方法签名：public int checkIdentity(String UserID, String Password)

接口调用：public User checkIdentity(String UserID, String Password)判断该用户是否可以登录

实际调用方法:

loginManager.checkIdentity(UserID, Password)

2.3.1.1 **register**

方法签名：public boolean addNewUser(String PhoneNumber, String Password, int UserType)

接口调用：public User loginService.addNewUser(PhoneNumber, Password,UserType)注册一个新用户

实际调用方法:

loginManager.register(String PhoneNumber,String Password,int UserType)

2.3.2 **管理交易接口**

2.3.2.1 **page**

方法签名：public String page(String subject, String total)

接口调用：alipayClient.excutePage(tradeRequest) 通过阿里客户端执行相应的提交

实际调用方法:

paymentManger.page(tradeRequest)

2.3.2.2 **giveGift**

方法签名：public boolean giveGift(String commodityID, String UserID)

接口调用：payment.giveGift(String commodityID, String UserID) 通过阿里客户端执行相应的提交

实际调用方法:

paymentManger.giveGift(String commodityID, String UserID)

2.3.2.3 **getOrderList**

方法签名：public List getOrderList(String UserID)

接口调用：payment.getOrderList(String UserID)获得用户所有的交易的订单

实际调用方法：

paymentManger.getOrderList(String UserID)

2.3.3 **个人中心接口**

2.3.3.1 **addToWishList**

方法签名：public Map AddWishList(String CommodityID,String UserID)

接口调用：wishListService.AddWishList(CommodityID,UserID)将指定的游戏加入愿望单

实际调用方法：

wishListManager.AddWishList(CommodityID,UserID)

2.3.3.2 **addToShoppingCart**

方法签名：public Map AddShoppingCart(String CommodityID,String UserID)

接口调用：shoppingCartService.AddShoppingCart(CommodityID,UserID)将指定的游戏加入愿望单

实际调用方法：

shoppingCartManager.AddShoppingCart(CommodityID,UserID)

2.3.4 **游戏库管理接口**

2.3.4.1 **getLibrary**

方法签名：public List GetLibItems(String BuyerID)

接口调用：gameLibService.getGameIDForLibrary(BuyerID)得到指定用户的所有游戏

实际调用方法：

gameLibManager.getGameIDForLibrary(BuyerID)

2.3.4.2 **removeGame**

方法签名：public boolean removeGame(String commodityID)

接口调用：gameLibService.removeGame(commodityID)得到指定用户的所有游戏

实际调用方法：

gameLibManager.removeGame(commodityID)

2.3.5 **游戏发布接口**

2.3.5.1 **publishGenres**

方法签名：public String PublishGames(String PublisherID,String Name,String Description,String Url,int price,int genreID)

接口调用：sellerService.PublishAddGenre(PublisherID,Name,Description,Url,price,genreID)

实际调用方法：sellerManager.PublishAddGenre(PublisherID,Name,Description,Url,price,genreID)

2.3.5.2 **publishUpdate**

方法签名：public String PublishUpdate(UpdateItem updateItem)

接口调用：sellerService.PublishUpdateGame(UpdateItem updateItem)

实际调用方法：sellerManager.PublishUpdateGame(UpdateItem updateItem)

2.3.5.3 **deleteGame**

方法签名：public boolean DeleteGame(String GameID)

接口调用：sellerService.deleteGame(String GameID)

实际调用方法：sellerManager.deleteGame(String GameID)

2.3.6 **信息管理接口**

2.3.6.1 **changeSellerInfo**

方法签名：public boolean changeSellerInfo(String SellerID,String Name,String Description,String HomePageUrl)

接口调用：sellerService.changeSellerInfo(SellerID,Name,Description,HomePageUrl)

实际调用方法：sellerManager.changeSellerInfo(SellerID,Name,Description,HomePageUrl)

2.3.6.2 **getBuyerInfo**

方法签名：public Map<String, Object> GetBuyerInfo(String BuyerID)

接口调用：buyerInfoService.getBuyerInfo(BuyerID)

实际调用方法：buyerInfoManager.getBuyerInfo(BuyerID)

2.3.6.3 **changeBuyerInfo**

方法签名：public boolean changeBuyerInfo(BuyerInfo buyerInfo)

接口调用：buyerInfoService.changeBuyerInfo(BuyerInfo buyerInfo)

实际调用方法：buyerInfoManager.changeBuyerInfo(BuyerInfo buyerInfo)

2.3.6.4 **blockBuyer**

方法签名：public boolean changeBuyerInfo(BuyerInfo buyerInfo)

接口调用：adminService.blockBuyer(BuyerID)

实际调用方法：adminManager.blockBuyer(BuyerID)

2.3.7 **游戏管理接口**

2.3.7.1 **searchGame**

方法签名：public List searchGame(String name)

接口调用：searchService.searchGame(string name)

实际调用方法:

searchManager.searchGame(String name)

2.3.7.2 **getCommodityInfo**

方法签名：public map getCommodityInfo(String CommodityID)

接口调用：commodityService.getCommodityInfo(string CommodityID)

实际调用方法:

commodityManager.getCommodityInfo(String CommodityID)

2.3.7.3 **addEvaluation**

方法签名：public map addEvaluation(String CommodityID,String UserID,String Evaluation)

接口调用：evaluationService.addEvaluation(String CommodityID,String UserID,String Evaluation)

实际调用方法:

evaluationManager.addEvaluation(String CommodityID,String UserID,String Evaluation)

2.3.7.4 **recommendGames**

方法签名：public map Recommendgames(String CommodityID,String UserID)

接口调用：adminService.Recommendgames(String CommodityID,String UserID)

实际调用方法:

adminManager.Recommendgames(String CommodityID,String UserID)

2.3.7.5 **deletePostedGames**

方法签名：public map deletePostedGames(String CommodityID,String UserID)

接口调用：adminService.deletePostedGames(String CommodityID,String UserID)

实际调用方法:

adminManager.deletePostedGames(String CommodityID,String UserID)

2.3.8 **好友管理接口**

2.3.8.1 **addFriends**

方法签名：public boolean addFriends(String FriendID,String UserID)

接口调用：friendService.addFriends(String FriendID,String UserID)

实际调用方法:

friendManager.addFriends(String FriendID,String UserID)

2.3.8.2 **deleteFriends**

方法签名：public boolean deleteFriends(String FriendID,String UserID)

接口调用：friendService.deleteFriends(String FriendID,String UserID)

实际调用方法:

friendManager.deleteFriends(String FriendID,String UserID)

2.3.8.3 **chatFriends**

方法签名：public string chatFriends(String FriendID,String UserID，string message)

接口调用：friendService.chatFriends(String FriendID,String UserID，string message))

实际调用方法:

friendManager.chatFriends(String FriendID,String UserID，string message)

2.4 **界面设计**

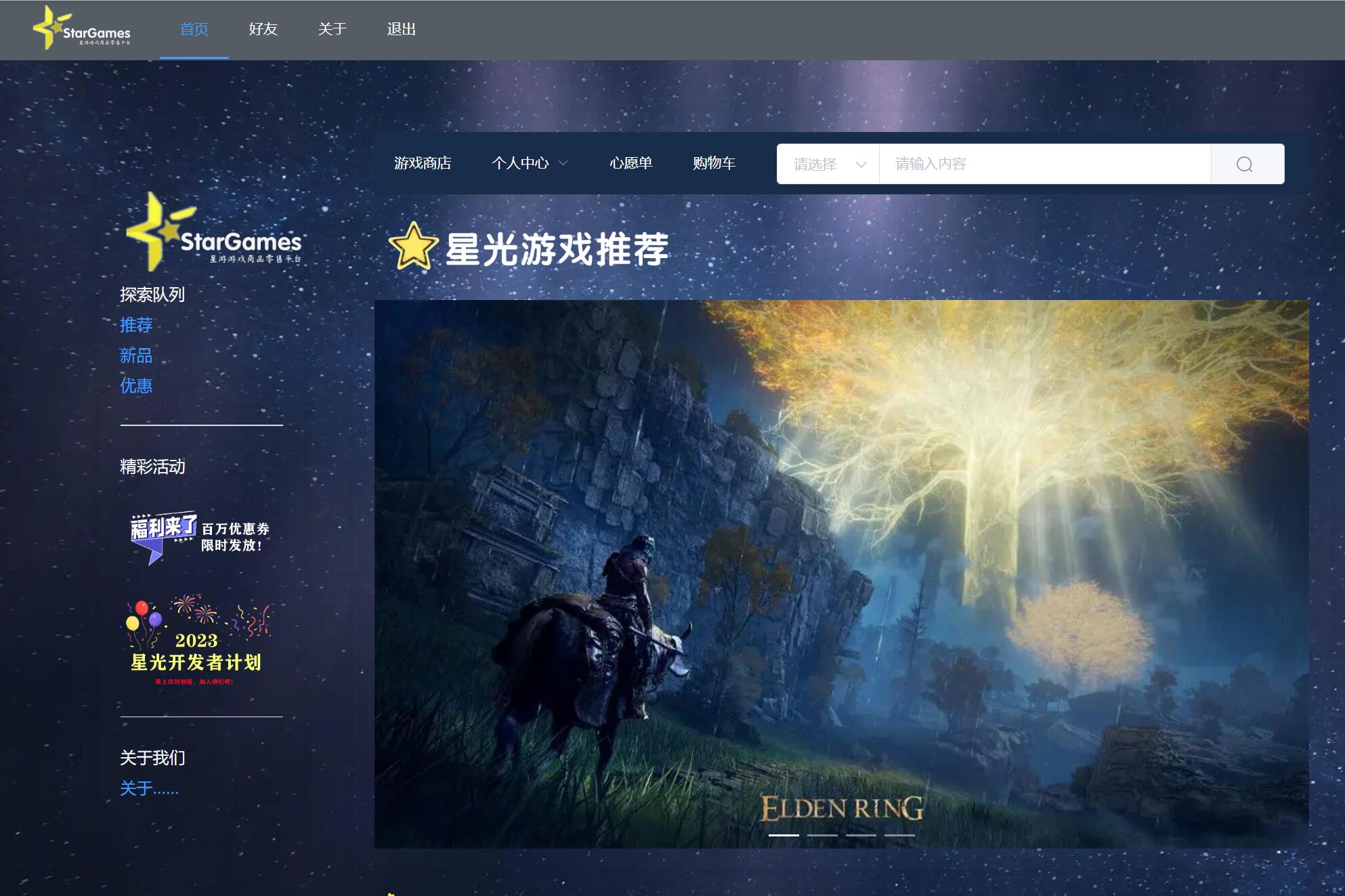
2.4.1 **登录界面**



2.4.2 **注册界面**



2.4.3 **首页界面**



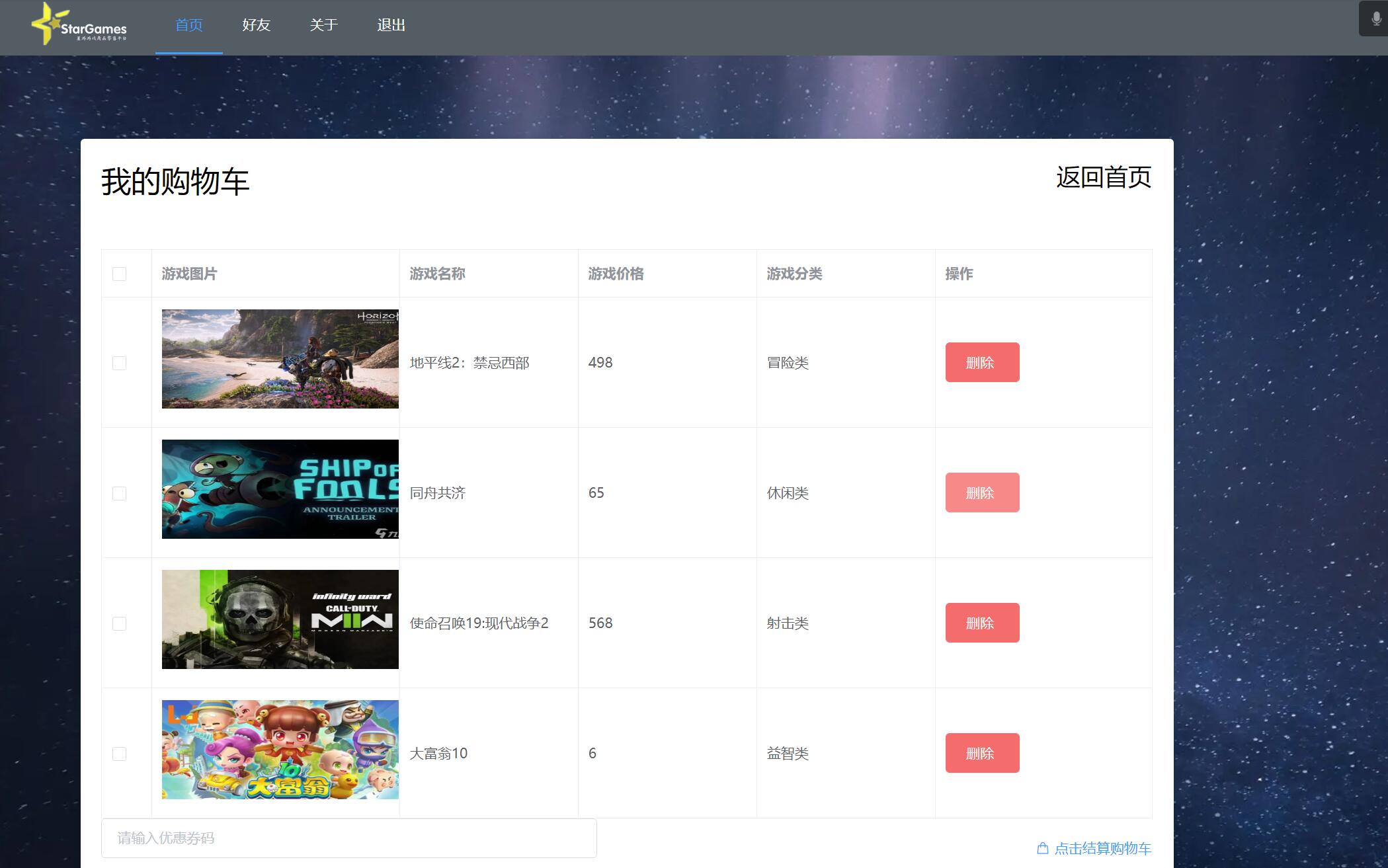
2.4.4 **游戏库界面**



2.4.5 **愿望单界面**



2.4.6 **购物车界面**

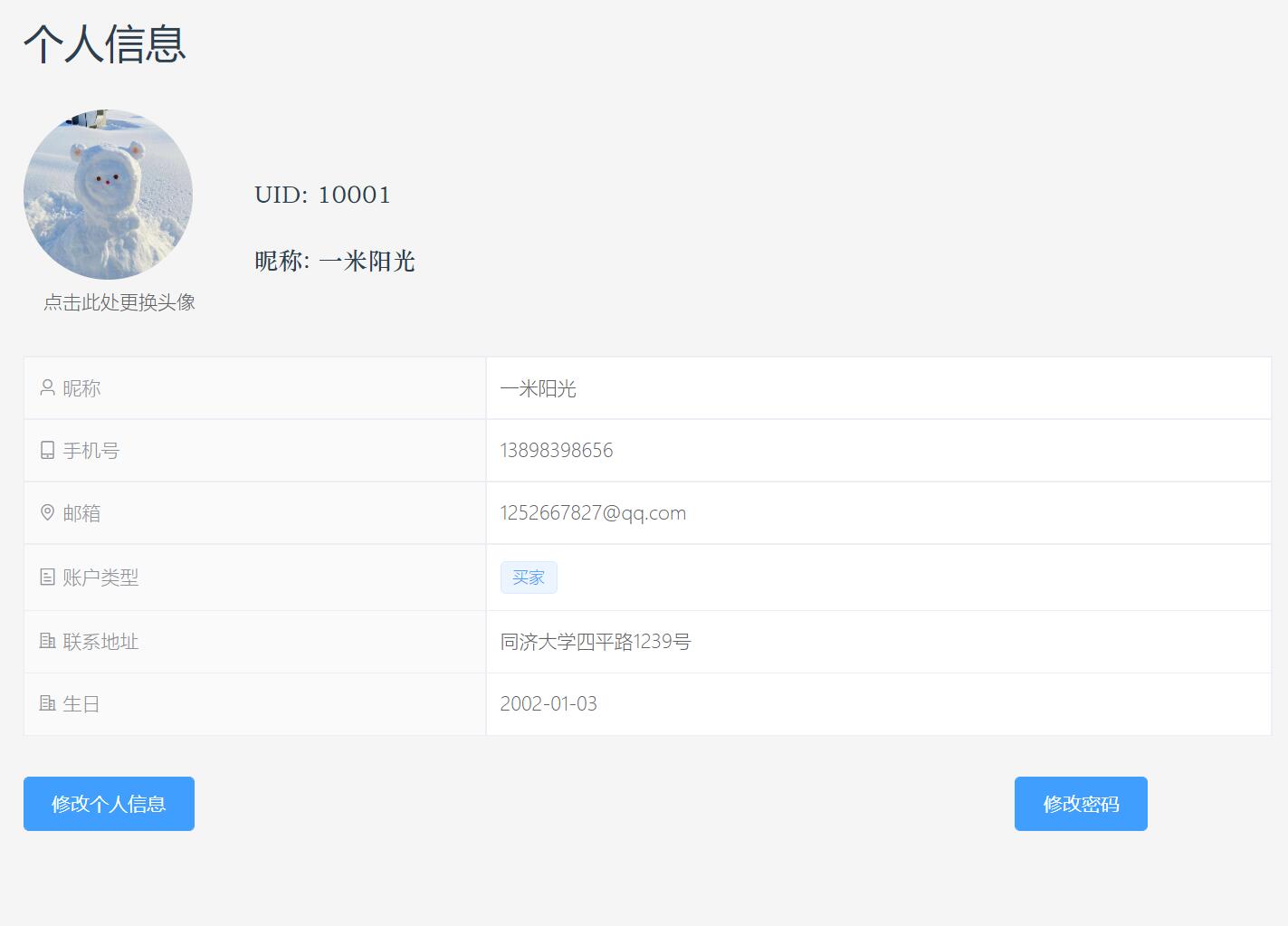


2.4.7 **搜索界面**





2.4.8 **个人信息界面**



2.4.9 **好友界面**



2.4.10 **发行商信息管理界面**



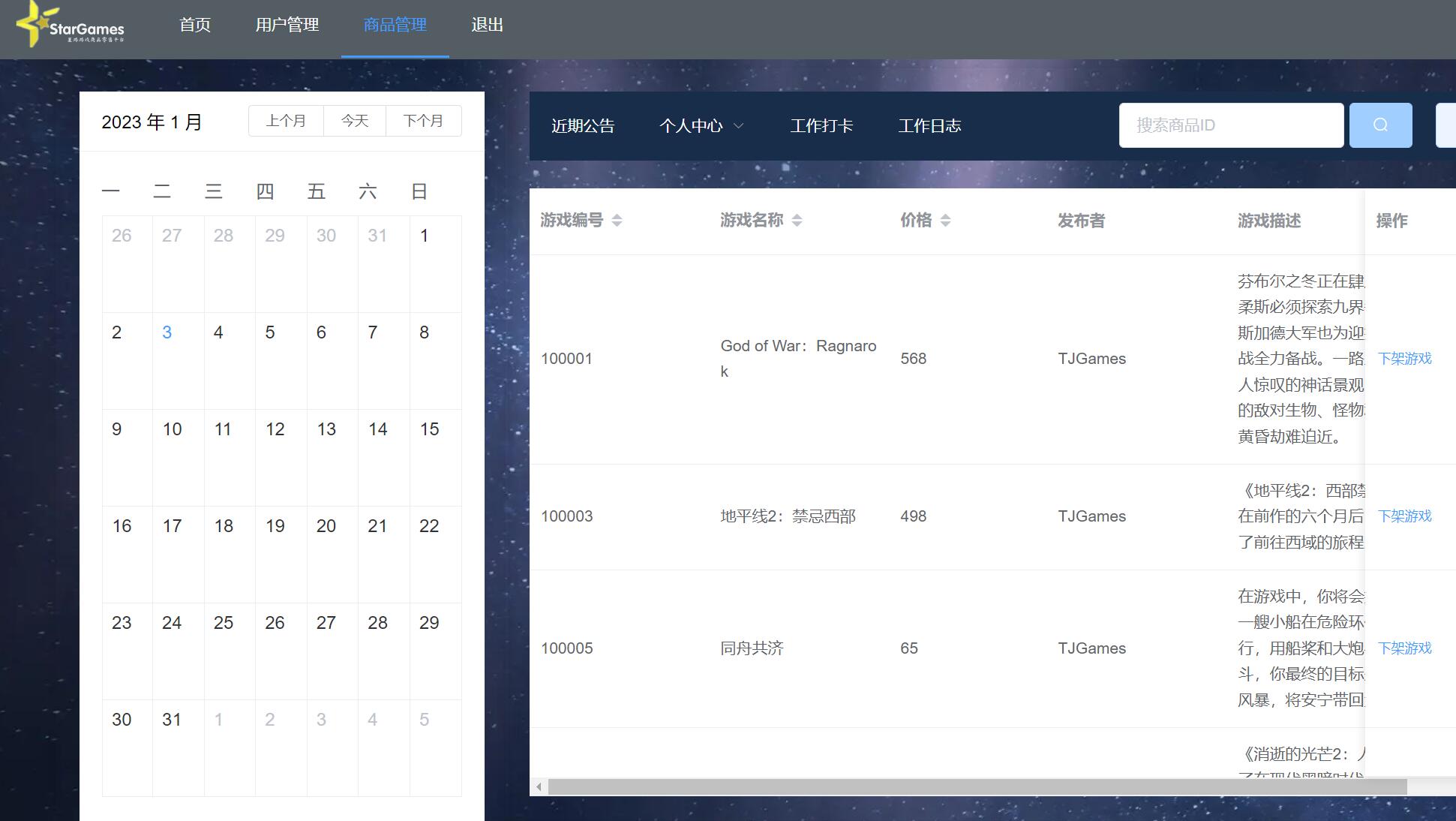
2.4.11 **发行商发布新游戏界面**



2.4.12 **管理员封禁用户界面**



2.4.13 **管理员下架游戏界面**



2.5 **数据库设计**

2.5.1 **数据库逻辑设计**

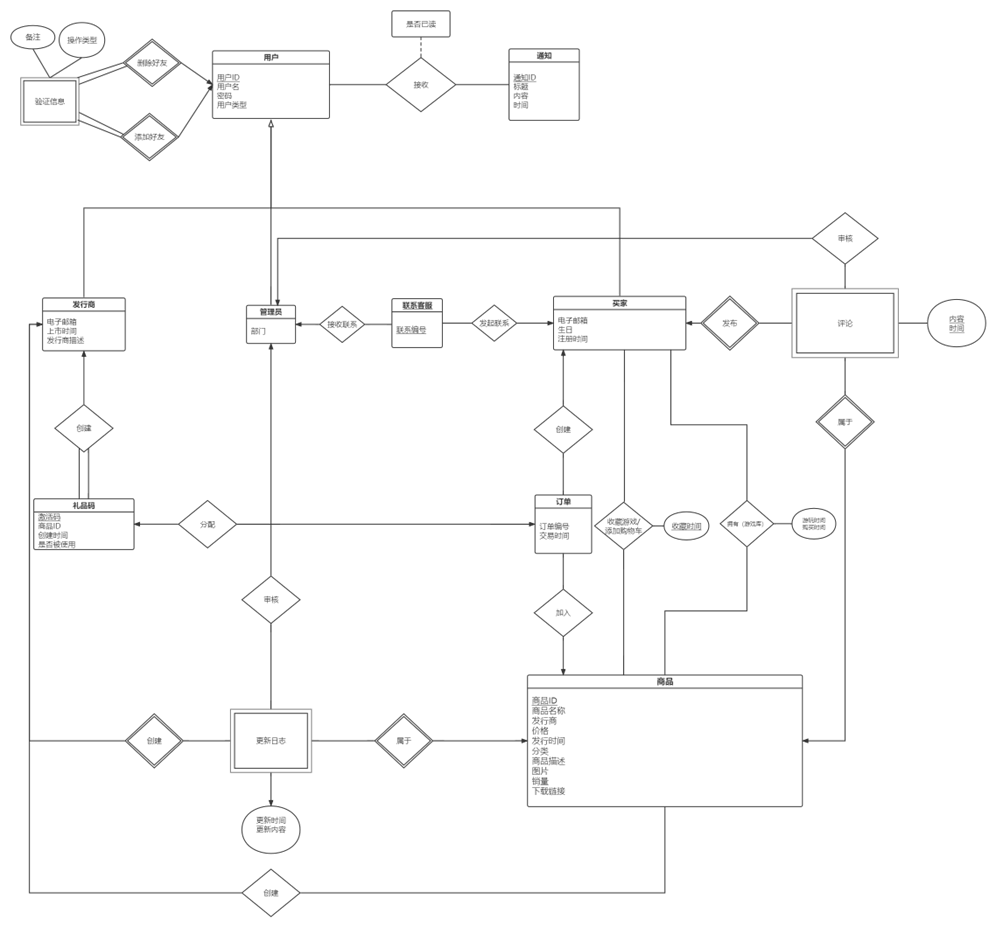
2.5.1.1 **数据库实体设计**

* *<entity>***用户**：是指本电子游戏交易系统的所有使用者，具体分为买家、卖家、管理员三种不同身份。每位用户拥有一个独一无二的账号，通过输入自己设置的密码进行登录。属性包括：用户 ID，用户名，密码，用户类型。
* *<entity>* **商品**：是指获得授权在本电子游戏交易系统中出售的所有游戏，每种商品由商品 ID 来唯一区分。商品的属性包括：商品 ID，商品名称，发行商，价格，发行时间，分类，商品描述，图片，下载链接，销量。
* *<entity>* **买家**：买家是用户的一种类型，拥有用户的全部属性以及收藏购买游戏、发表评论、添加好友等一系列专有权限。买家的额外属性有：联系方式、注册时间、生日。
* *<entity>* **游戏发行商**：游戏发行商是用户的一种类型，拥有用户的全部属性以及发售旗下游戏、停售旗下游戏，更新旗下游戏信息等一系列专有权限。游戏发行商的额外属性有：开店时间、卖家简介。
* *<entity>* **管理员**：管理员是用户的一种类型，拥有用户的全部属性以及修改整个系统游戏库、发布通知、删除商品评论等一系列专有权限。管理员的额外属性有：部门。
* *<entity>* **通知**：通知是指管理员向用户发送的信息。用户关注、收藏或购买的游戏上架、更新时，或是平台举行促销活动时，会向用户发送通知进行宣传。通知的属性包括：通知 ID，标题，内容，时间。
* *<entity>* **联系客服**：联系客服是指买家在系统使用过程中遇到问题需要向客服反应时的联系渠道，买家发起联系信息，管理员接受联系信息。联系客服具有的属性包括：联系编号。
* *<weak entity>* **评论**：评论是买家针对某款发售商品发表的评论性话语，并展示在商品页面中，对所有人可见，这是一个弱实体，它依赖于买家实体集和商品实体集。评论的属性包括：评论内容和评论时间。
* *<entity>* **订单**：订单是由买家发起的商品交易请求，在买家完成支付后由系统自动生成交易信息。订单的属性包括：订单编号，交易时间。
* *<weak entity>* **更新日志**：更新日志是指，在每次游戏发行商对在售游戏进行更新维护之后记录的本次更新具体信息。更新日志是一个弱实体，其存在依赖于发行商和商品实体集。更新日志的属性包括：更新时间、更新内容。
* *<entity>* **礼品码**：礼品码是指由游戏发行商创建的，可以用于兑换某款游戏的一串字符，它会在订单发布后发送给下单的买家。礼品码的属性包括：激活码，生成时间，使用情况。
* *<weak entity>* **验证信息**：验证信息是指在用户申请对另一名用户添加或删除好友时，系统向另一名用户发送的验证信息。验证信息是一个弱实体，它依赖于用户实体。验证信息包含的属性有备注和操作类型。

2.5.1.2 **数据库关系设计**

* **用户与验证信息之间的删除好友联系集**：用户通过验证信息来备注要删除的好友和选择删除的操作类型来实现删除好友的操作，每一条验证信息都对应一种操作类型并返回给用户。
* **用户与验证信息之间的添加好友联系集**：用户通过验证信息来备注要添加的好友和选择添加好友的操作类型来实现添加好友的操作，每一条验证信息都对应一种操作类型并返回给用户。
* **用户和通知之间的接收联系集**：用户能够接收到后台编辑的通知，一个用户可以收到多条通知，一个通知可以有多个用户收到，属于多对多的关系。通知联系集中还有是否已读的信息。
* **发行商与礼品码之间的创建联系集**：礼品码由发行商创建，用来买给买家来激活游戏，一个发行商可以创建多个商品的礼品码，一个礼品码只能由一个发行商创建，属于多对一的关系，并且礼品码全部由发行商发布，是全部参与的。
* **管理员和联系客服之间的接收联系关系集**：买家联系客服人员后，管理员通过联系客服实体集接收联系后安排客服人员，一个管理员可以接受多个客服申请，一个客服申请对应由一个管理员进行处理，属于多对一的联系。
* **买家和联系客服之间的发起联系联系集**：买家通过发起客服联系来申请客服，由管理员来审核处理，一个买家可以发起多个客服申请，一个客服申请对应一个买家，属于多对一的联系。
* **买家和评论之间的发布关系**：买家通过礼品码购买游戏后可以体验游戏，体验完成后可以对商品进行发布评论，一个买家可以发布多条评论，一个评论对应一个买家，属于多对一的关系。
* **评论和商品之间的属于联系集**：对于发行商发行的每一款商品都有一个评论区，评论属于商品，一个商品可以有多个买家发布的评论，一个评论只属于一款产品，属于多对一的关系。
* **管理员和评论之间的审核联系集**：买家发布评论之后，其中会有一些不合理或者不合规的言论，这时候就需要管理员来对评论进行审核，管理员有删除不当言论的评论，一个管理员可以审核多条评论，一条评论只需要一个管理员审核，属于多对一的联系。
* **买家和订单之间的创建联系集**：买家通过礼品码购买商品的同时，系统会自动生成订单，该订单内容买家创建，一个订单内只属于一个买家，一个买家可以购买多个游戏创建多个订单，属于多对一的关系。
* **订单和商品之间的加入联系集**：一个订单内的具体内容包含一款商品，商品信息在创建订单时会加入到订单信息中，一个订单只包含一款游戏，一个游戏可以被创建多个订单，属于多对一的关系。
* **买家和商品之间的收藏游戏联系集**：买家可以通过浏览商品列表，选中心仪的游戏可以对游戏进行收藏，一个买家可以收藏多款游戏，一款游戏也可以被多个买家收藏属于多对多的关系。
* **礼品码和订单之间的分配联系集**：买家是通过礼品码来购买游戏的，来创建订单的，所以在创建订单时将买家的礼品码分配给订单信息，一个订单对应一个礼品码，属于一对一的关系。
* **管理员和更新日志之间的审核联系集**：对于商品发布更新日志会由管理员进行审核，一个更新日志只由一个管理员审核，一个管理员可以审核多个更新日志，属于多对一的关系。
* **发行商与更新日志之间的创建联系集**：对于游戏的更新日志由发行商自己创建，一个发行商可以创建多个更新日志，一个更新日志只由一个发行上发布，属于多对一的关系。
* **商品与更新日志的属于联系集**：对于每一款游戏都会有后续的更新，一个游戏可以有多个更新日志来进行后续更新，一个更新日志只属于一款游戏，属于多对一的关系。
* **商品和发行商之间的创建联系集**：所有的游戏由发行商发行，供买家来通过礼品码购买，一个发行商可以发行多个游戏，一个游戏只能属于一个发行商，属于多对一的关系。
* **商品和买家之间的拥有联系集**：玩家购买游戏后会将游戏存储到自己的游戏库中，属于拥有关系，一个买家可以拥有多个游戏，一个游戏可以被多个买家拥有，属于多对多的关系，所以包含一个游戏库表。

2.5.1.3 **数据库E-R图设计**



2.5.2 **数据库物理设计**

关系模型的物理结构是一组关系模式的集合。为得到数据库的物理结构，我们需要将E-R图中的实体、实体中的属性和实体与实体的联系转换为关系模式。对此我们采用如下的转换原则：

* 实体类型的转换：一个实体集转换成一个关系模式。实体的属性就是关系的属性，实体的码就是关系的码。
* 对于一对一的联系而言：转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，每个实体的码均是该关系的候选码。如果与某一端实体对应的关系模式合并，则需要在该关系模式的属性中加入另一个关系模式的码和联系本身的属性。
* 对于一对多的联系而言：转换为一个独立的关系模式，也可以与多端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为N端实体的码。
* 对于多对多的联系而言：转换为一个关系模式。与该联系相连的各实体的码为各实体码的组合。
* 对于三个或三个以上的多元联系而言：转换成一个关系模式。与该多元联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为各实体码的组合。
* 具有相同键的关系模式可以进行合并

根据以上原则，我们将E-R转换为了如下的数据库结构，现将所有的数据

库表的名称、标识符、数据项、记录、存储要求、存储类型、索引、存储区域以表格的形式列举如下：

1. **用户表 *(Users)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| User\_ID | number | 20 | 用户编号 | PK，用户个人编号 |
| UserName | varchar | 20 | 用户名 | 表中存储用户名 |
| Password | varchar | 20 | 密码 | 用户登录密码 (HASH 值) |
| UserType | number | 20 | 用户类型 | 可分为买家、卖家两种类型，可使用不同的功能。 |

1. **买家表 *(Buyers)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| Buyer\_ID | number | 20 | 买家用户名 | PK，FK，参照用户表中的 UserId |
| BuyerName | varchar | 20 | 注册时间 | FK，参照用户表中的 UserName |
| RegistrationTime | date |  | 用户名 | 买家注册账号的时间 |
| Birthday | date |  | 买家生日 | 可以在生日时享受特殊优惠等 |
| Phone | varchar | 20 | 联系方式 | 买家的联系方式，多为手机 |

1. **发行商表 *(Publishers)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| Publisher\_ID | number | 20 | 发行商 ID | PK，FK，参照用户表中的 UserId |
| Publisher\_Name | varchar | 20 | 发行商用户名 | FK，参照用户表中的 UserName |
| StartTime | date |  | 开店时间 | 发行商开始售卖的时间 |
| Description | varchar | 2000 | 发行商描述 | 可以包括发行商售卖产品类型等等 |
| Mail | varchar | 50 | 电子邮箱 | 买家可通过电子邮箱与发行商交流 |

1. **管理员表 *(Adiministrators)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| Admin\_ID | number | 20 | 管理员编号 | PK，FK，参照了 user 表的 UserId 属性 |
| Admin\_Name | varchar | 20 | 管理员姓名 | FK，参照用户表中的 UserName，存储管理员的真实姓名 |
| Department | varchar | 100 | 部门 | 管理员所在部门名称 |

1. **商品表 *(Commodities)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| Commodity\_ID | number | 20 | 商品编号 | PK |
| Commodity\_Name | varchar | 20 | 商品名 | 表中存储用户名 |
| Publisher | varchar | 20 | 发行商 | 发行商名 |
| Price | number | 10 | 商品价格 |  |
| Publish\_Time | varchar | 50 | 发行时间 |  |
| Classification | varchar | 20 | 商品类别 | 游戏的类别，如休闲类、棋牌类、动作类等等 |
| Description | varchar | 100 | 商品描述 |  |
| Picture | blob |  | 商品附图 |  |
| DownloadURL | varchar | 100 | 下载地址 |  |
| SalesVolume | Number | 10 | 销量 |  |

1. **愿望单表 *(Wishlist)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| ID | number | 20 | 用户 ID | PK，FK，参照用户表中的 ID |
| Commodity\_ID | number | 20 | 商品编号 | PK，FK，参照商品表中的 ID |
| Promote\_Message | varchar | 100 | 促销信息 |  |

1. **购物车表 *(ShoppingCart)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| ID | number | 20 | 用户 ID | PK，FK，参照用户表中的 ID |
| Commodity\_ID | number | 20 | 商品编号 | PK，FK，参照商品表中的 ID |
| Join\_Time | varchar | 50 | 加入时间 | 商品加入购物车的时间 |

1. **游戏库表 *(GameLibrary)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| ID | number | 20 | 用户 ID | PK，FK，参照用户表中的 ID |
| Commodity\_ID | number | 20 | 商品编号 | PK，FK，参照商品表中的 ID |
| Purchase\_Time | varchar | 50 | 购买时间 | 购买游戏的时间 |
| Game\_Time | varchar | 50 | 游戏时长 | 游玩此游戏的时长 |

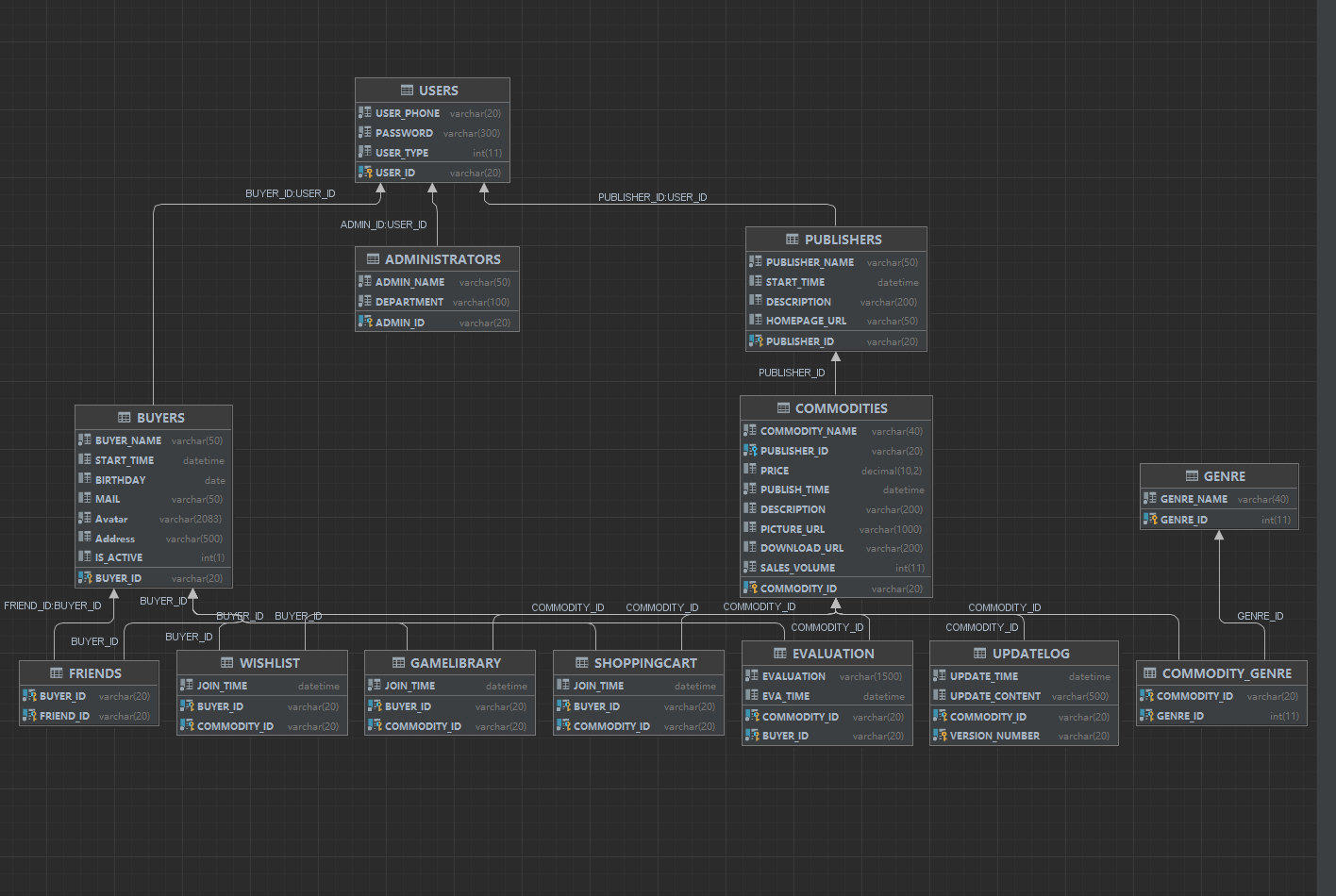
1. **通知表 *(Message)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| Message\_ID | number | 20 | 通知编号 | PK |
| Message\_title | varchar | 20 | 通知标题 | 表中存储通知标题 |
| Message\_content | varchar | 1000 | 通知内容 | 表中存储通知内容 |
| Message\_Time | date | 7 | 通知时间 | 表中存储发送通知的时间 |
| Receiver\_ID | number | 20 | 接收方 ID | FK |
| Is\_read | number | 1 | 是否已读 | 表中存储是否已读 |

1. **更新日志表 *(Update log)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 说明 | 备注 |
| Commodity\_ID | number | 20 | 商品编号 | PK，FK，参照用户表中的 ID |
| Version\_number | varchar | 20 | 更新版本号 | PK |
| Update\_Time | date | 7 | 更新时间 | 表中存储更新时间 |
| Update\_content | varchar | 1000 | 更新内容 | 表中存储更新内容 |

以上是系统数据库每张表的详细结构。综合分析所有表的相互关系后，我们得出了如下图所示的物理模型：



2.5.3 **数据库设计考虑**

2.5.3.1 **数据库范式**

由于该系统数据表较多，且业务需求设计到各张表频繁的增删改查，因此满足一定的范式，以保证不会发上插入、删除和更新的操作异常，同时降低数据的冗余。对于本系统设计的关系模式而言，满足了如下范式：

* 第一范式（1NF）。在归宿的关系模式中，所有的域都是原子性的，数据表的每一项都是不可分割的原子项。
* 第二范式（2NF）。归宿数据库表中的每个实例或记录均可以被唯一地区分。
* 第三范式（3NF）。归宿的关系中每个关系都不包含已在其他关系出现的非主关键字信息，任何非主属性不得传递依赖于主属性。

2.5.3.2 **存储查询优化**

随着系统发布日期的延长，数据库中的数据项将会越来越多，负载也会也会越来越大，较慢的查询效率必然会影响整体系统的响应时间。对此，我们对数据库表建立了必要的索引，以加快数据库的查询速度。对此我们制定了以下原则来建立简单索引和组合索引：

* 默认对表的外键建立索引
* 对查询中频繁出现的查询条件对应的字段建立索引
* 对于非必要的字段尽量不建立索引，以避免增加数据库的存储

除此之外，我们还对于某些复杂查询建立视图，以简化查询，方便管理。

2.5.3.3 **函数和触发器**

为了保证系统功能的正常运行，我们相应设计了函数和触发器来自动维护这些功能。

2.6 **系统出错处理设计**

2.6.1 **出错信息**

本系统涉及用户的个人隐私、财产和支付等重要信息，因此对于一些严重的错误建立完备的出错处理机制和系统故障发生后的补救措施是十分重要的。对此我们列举了如下的常见出错信息和预计的系统处理方法。由于某些错误可以避免，因此对于特定的软错误而言，我们直接在前端予以提示和修正。对于一些可能人为造成的错误而言，如误删误改操作而言，我们提供完备的确认机制；对于一些关键信息，如密码等，为防止黑客攻击系统盗用损坏用户信息，系统提相应的保密和加密措施。

以下以一览表的形式说明每种可能出现的软错误和硬错误发生时，系统输出信息的形式，含义及处理方法。

|  |  |  |  |
| --- | --- | --- | --- |
| 错误类型 | 错误信息 | 处理方法 | 系统输出信息 |
| 数据库错误 | 数据库连接失败 | 尝试重连 | 前端输出错误提示信息 |
| 数据库操作失败 | 事务回滚，输出错误信息 | 前端输出更新失败信息 |
| 数据库溢出 | 事务回滚，输出错误信息 | 前端输出更新失败信息 |
| TCP连接错误 | 连接超时或断开 | 尝试重新连接，特定时间后输出错误信息 | 尝试失败后输出对应的网络错误信息 |
| 系统部分自定义错误 | 用户名密码错误 | 不予登录 | 系统输出输入错误提示信息，登陆失败 |
| 身份证输入错误 | 不予注册 | 系统输出错误的身份证提示信息，登陆失败 |
| 验证码错误/为空 | 不予登录，切换验证码 | 系统输出验证码错误提示信息，登陆失败 |
| 修改个人信息错误 | 不予修改 | 系统输出不满足条件的错误修改提示信息，修改失败 |
| 游戏信息错误 | 不予发布游戏，空操作 | 系统标红不满足输入条件的游戏信息，发布失败，等待修改 |
| 帖子发布信息错误 | 不予发布帖子，空操作 | 系统标红不满足输入条件的帖子信息，发布失败，等待修改 |
| 支付错误 | 系统回滚事务 | 系统输出支付失败信息并跳转回待支付界面 |
| 链接错误 | 图片链接错误 | 抛出异常 | 系统展示占位图片信息，提示图片展示失败 |
| 外部链接错误 | 抛出异常 | 系统提示外部链接错误信息 |
| 程序错误 | 空指针错误 | 服务停止重启 | 系统提示内部错误并重启服务 |
| 内部程序错误 | 服务重启 | 系统提示内部错误并重启服务 |
| 其他错误 | 服务器故障 | 无 | 系统提示服务器故障信息并等待人为修复重启 |

2.6.2 **补救措施**

由于错误类型不尽相同，错误的原因也各有差异，因此对于不同的错误我们采取不同的变通措施，列举如下：

* **对于软错误**，如用户输入非法信息，可能会导致后端数据库的操作错误和程序内部发生错误，系统直接在前端的输入或修改操作中对数据本身进行验证，分析错误类型，并给出相应的错误提示语句，从用户输入层面避免输入信息的非法，从而从一定程度上避免因输入非法带来的故障。
* **对于硬错误**，如数据库连接错误和网络连接错误而言，由于错误类型较少且原因明确，可在前端输出对应的提示信息；而对于程序内部的错误，可以在程序编写阶段设置对应的异常捕获程序和抛出异常语句，在出错时输出相应的错误语句，将服务或程序重启，避免整个业务的故障停滞。维护人员可在输出日志中查看错误信息及时修正。
* 采取适当的后备技术，如当数据库受到攻击或删库时，通过定期转储数据库，对数据库定期备份避免非法攻击带来的不可逆故障。
* 制定完备的故障恢复和重启技术，对于因程序内部错误导致的服务故障，采取服务重启并从头开始运行来保证业务逻辑的正常执行。

2.6.3 **系统维护设计**

2.6.3.1 **数据库维护**

数据库存储了用户的全部信息，涉及到用户的安全和隐私，对于数据库的维护是重中之重。对此我们建立了完备的数据库维护方案，主要包含如下几点：

* 数据库定期备份。为防止数据库操作失误、服务器宕机、服务器数据丢失带来的严重后果，定期对数据库进行转储备份来避免数据丢失的发生。
* 定期检查数据库的一致性。数据的不一致可能会导致某些业务功能的失败和财产的流失，因此需要定期对于数据库事务的一致性和数据一致性进行检查，特别是涉及到房源订单数据的一致性检查。
* 定期删除冗余数据。由于本系统对于房源的删除机制是采用标志位的形式标识房源的删除与否，因此为了防止数据出现过多冗余，管理员需要定期删除一些不再有用的冗余数据。
* 定期查看操作日志。定期查看操作日志以检查数据库设计可能存在的缺陷，及时修复和更改。

2.6.3.2 **后端维护**

除数据库外，后端程序的错误和失败也会严重影响整个系统业务的正常运行。因此，需要专门指定维护的监测点和模块来对后端代码进行维护。

* 异常抛出模块。对于每个服务实现类，我们均根据特定的错误情形加入相应的try-catch块，以保证当错误发生时不会影响到程序的正常运行。
* 单元测试模块。为保证后端定期维护作出的修改的正确性，每个控制类和业务实现类均加入了单元测试模块，方便快速测试和维护。