影院管理系统（CMS）体系结构设计文档

文档修改历史

修改人员	日期	修改原因	版本号
Agent_67	2019.5.22	初始化	v0.0
Agent_67	2019.5.25	第一次整合	v1.0
Agent_67	2019.6.17	添加包设计图	v2.0
Agent_67	2019.6.19	修复描述性错误	v2.1

1.引言

1.1 编制目的

本报告详细完成对影院管理系统的概要设计，达到指导详细设计和开发的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户编写，是了解系统的导航。

1.2 词汇表

词汇名称	词汇含义	备注
CMS	影院管理系统	Cinema Management System

1.3参考资料

1. IEEE标准。
2. 影院管理系统用例文档。
3. 影院管理系统需求规格说明文档。
4. 骆斌，丁二玉，刘钦 - 软件工程与计算 . 卷二 , 软件开发的技术基础 : Software engineering and computing . Volume II , Fundamentals of software development technology。

2.产品概述

参考影院管理系统用例文档和需求规格说明文档中对产品的概括描述。

3.逻辑视图

影院管理系统中，选择了分层体系结构风格，将系统分为4层（展示层、控制层、业务逻辑层、数据层）能够很好地示意整个高层抽象。展示层包含UI界面，控制层包含UI界面的跳转和业务的访问，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图1和图2所示。

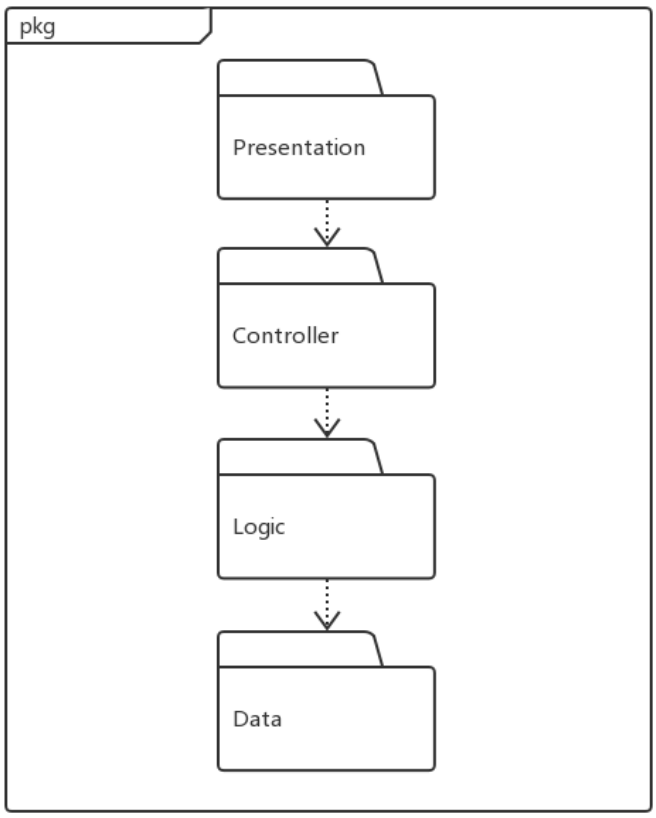

图1 参照体系风格结构的包图表达逻辑

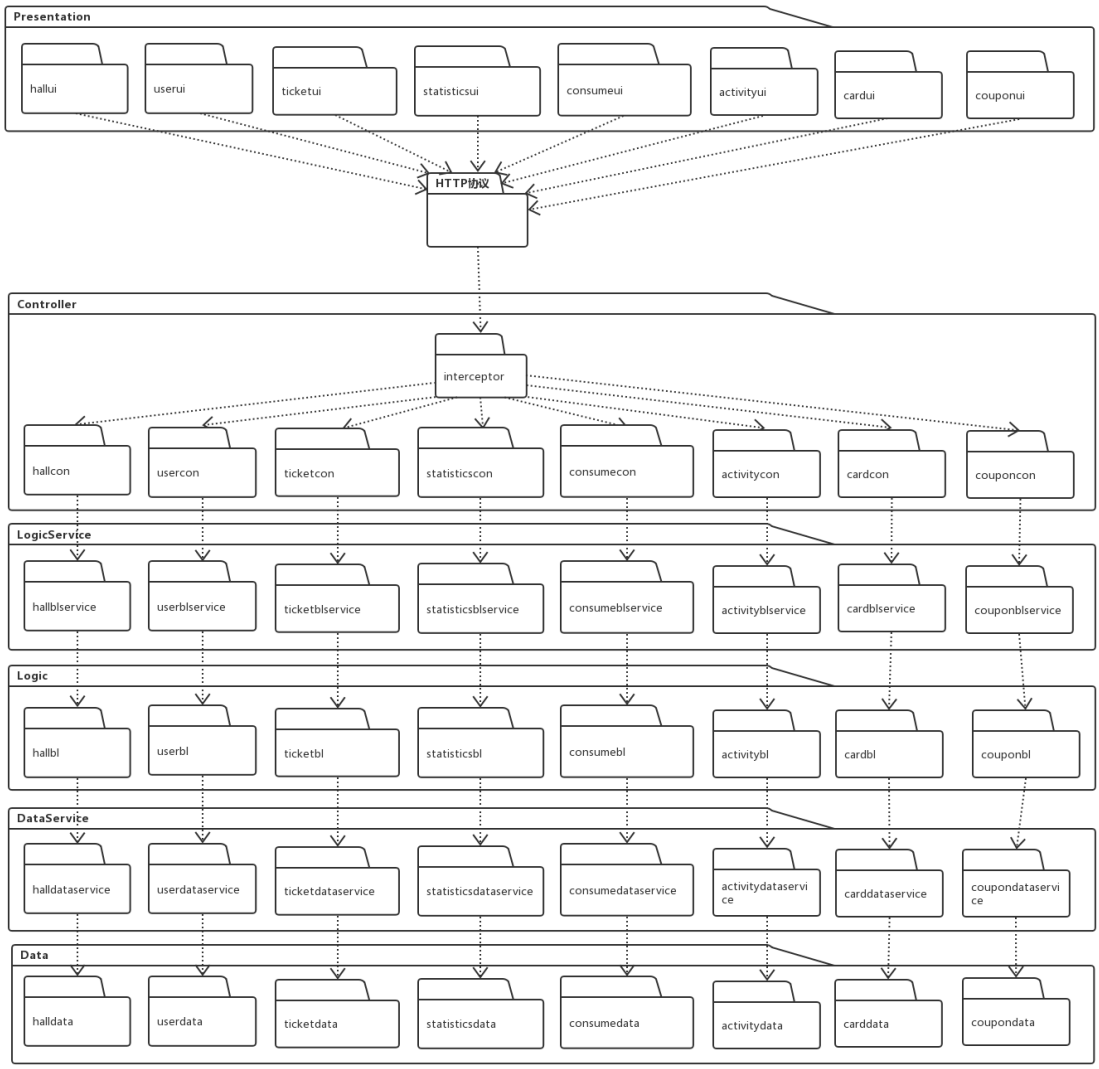

图2 软件体系结构逻辑设计方案

4.组合视图

4.1 开发包图

影院管理系统的第三阶段开发包设计如表1所示。

表1 影院管理系统的第三阶段开发包设计

开发（物理）包	依赖的其他开发包
interceptor	usercon,hallcon,ticketcon,cardcon, activitycon,consumecon,couponcon,statisticscon
viewcon	
userui	usercon,vo
usercon	userblservice,vo

开发（物理）包	依赖的其他开发包
userblservice	
userbl	userdataservice,po,vo
userdataservice	po
userdata	mybatis,po
hallui	hallcon,vo
hallcon	hallblservice,vo
hallblservice	
hallbl	halldataservice,po,vo
halldataservice	po
halldata	mybatis,po
ticketui	ticketcon,vo
ticketcon	ticketblservice,vo
ticketblservice	
ticketbl	ticketdataservice,po,vo
ticketdataservice	po
ticketdata	mybatis,po
cardui	cardcon,vo
cardcon	cardservice,vo
cardblservice	
cardbl	carddataservice,po,vo
carddataservice	po
carddata	mybatis,po
couponcon	couponservice,vo
couponblservice	
couponbl	coupondataservice,po,vo
coupondataservice	po
coupondata	mybatis,po
activityui	activitycon,vo
activitycon	activityblservice,vo
activityblservice	

开发（物理）包	依赖的其他开发包
activitybl	activitydataservice,po,vo
activitydataservice	po
activitydata	mybatis,po
statisticsui	statisticscon,vo
statisticscon	statisticsblservice,vo
statisticsbl	
statisticsblservice	statisticsdataservice,po,vo
statisticsdataservice	po
statisticsdata	mybatis,po
consumeui	consumecon,vo
consumecon	consumeblservice,vo
consumeblservice	
consumebl	consumedataservice,po,vo
consumedataservice	po
consumedata	mybatis,po
vo	
po	
mybatis	

影院管理系统客户端开发包图如图3所示，服务器端开发包图如图4所示。

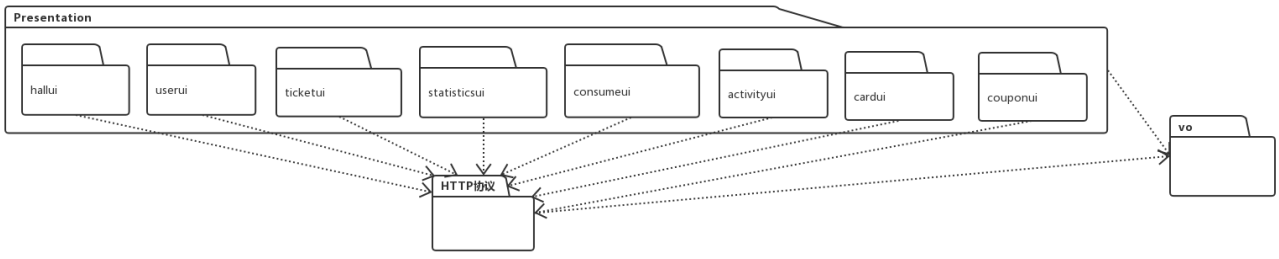

图3 影院管理系统客户端开发包图

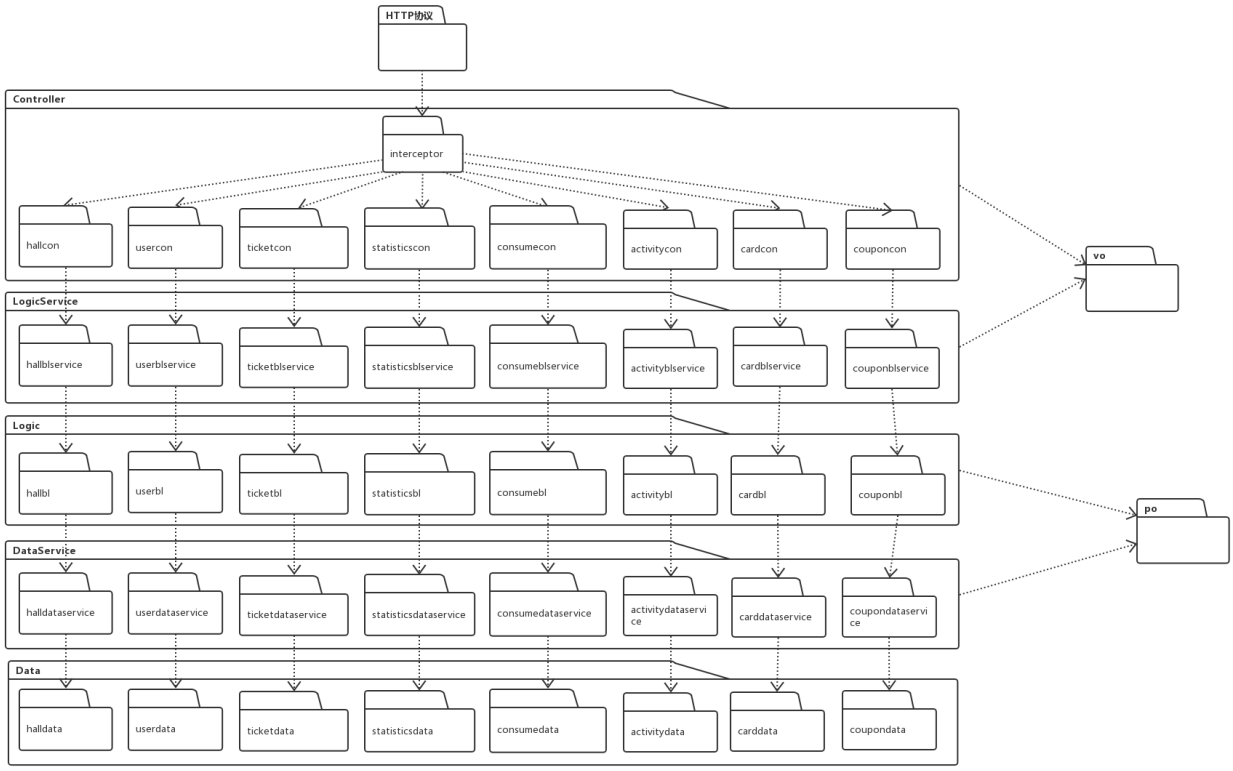

图4 影院管理系统服务器端开发包图

4.2 运行时进程

在影院管理系统中，会有一个服务器进程，其进程如图5所示。结合部署图，客户端是客户端机器上的浏览器，服务端进程是在服务端机器上运行。

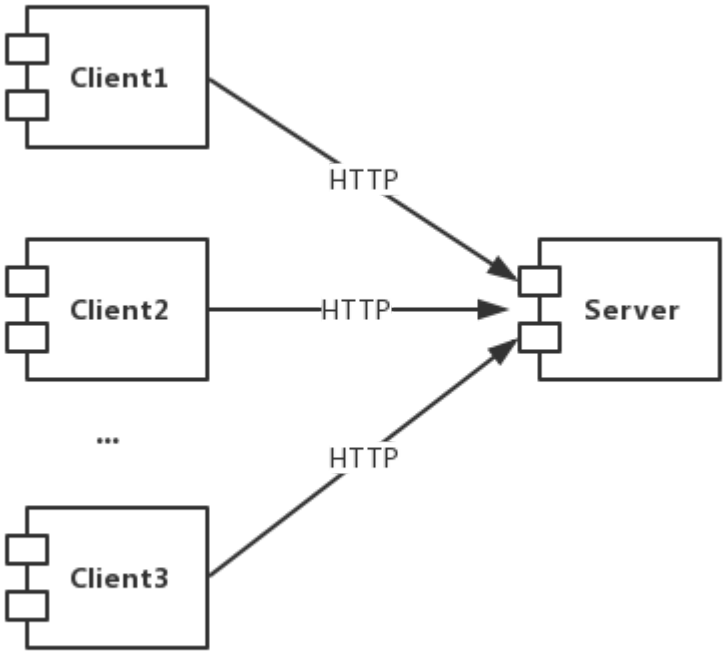

图5 进程图

4.3 物理部署

影院管理系统中服务器端构件是放在服务器端机器上，客户端通过HTTP协议与服务器进行交互，不需要在独立部署。部署图如图6所示。

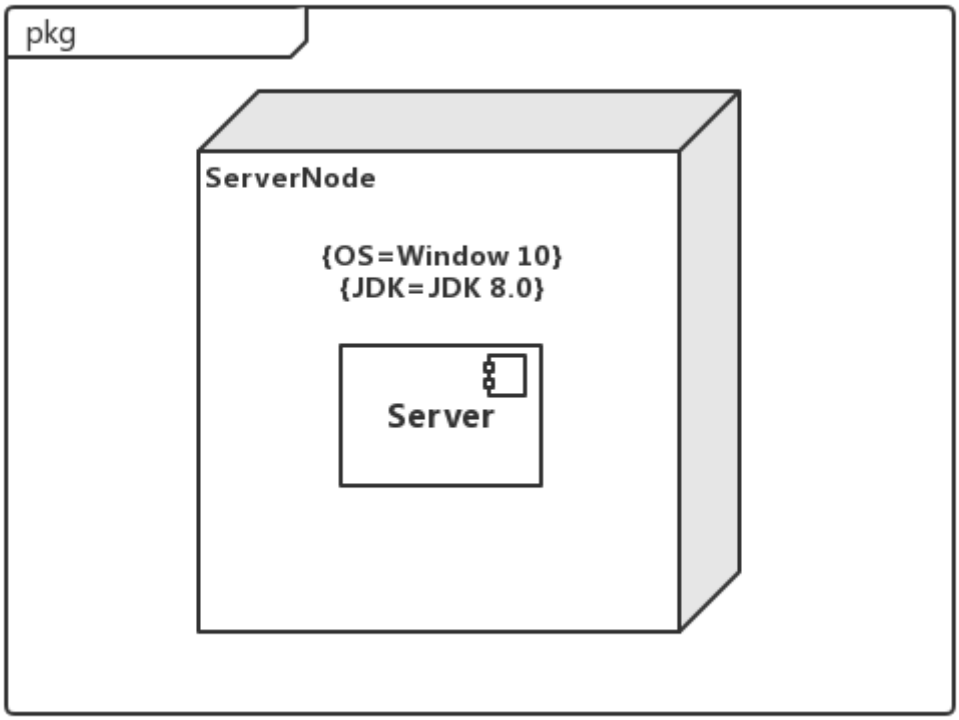

图6 部署图

5.架构设计

5.1 模块职责

服务器端模块视图如图7所示。服务器端各层的职责如表2所示。

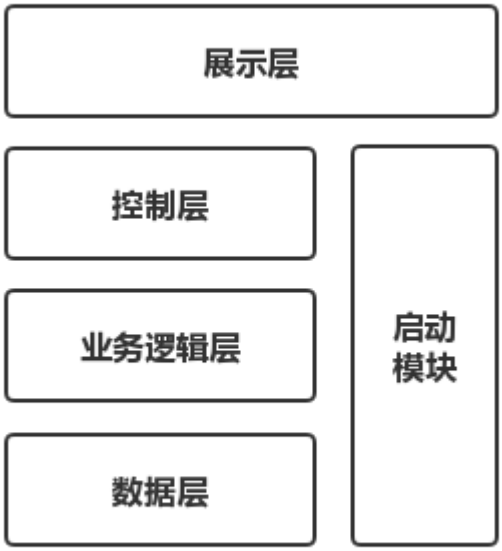

图7 服务器端模块视图

表2 服务器端各层的职责

层	职责
展示层	显示在客户端浏览器中的用户界面
控制层	基于用户请求的URL调用相应的服务
业务逻辑层	进行业务逻辑处理
数据层	数据的持久化及数据访问接口
启动模块	负责启动SpringBoot项目进程

每一层只是使用下方直接接触的层。除了展示层与控制层之间通过URL进行调用，层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表3所示。

表3 层之间调用的接口

接口	服务调用方	服务提供方
UserBLService HallBLService SalesBLService CardBLService UserDataService ActivityBLService StatisticsBLSerivce ConsumeBLService	控制层	业务逻辑层
HallDataService SalesDataServie	业务逻辑层	数据层

借用User模块来说明层之间的调用，如图8所示。控制层、业务逻辑层、数据层之间都由上层依赖了一个接口（需接口），而下层实现这个接口（供接口）。UserBLService提供了UserController所要调用的服务。

UserDataBlService提供了对数据库的增、查等操作。这样的实现就大大降低了层与层之间的耦合。

CouponBLService

ActivityDataService

StatisticsDataService

Cons

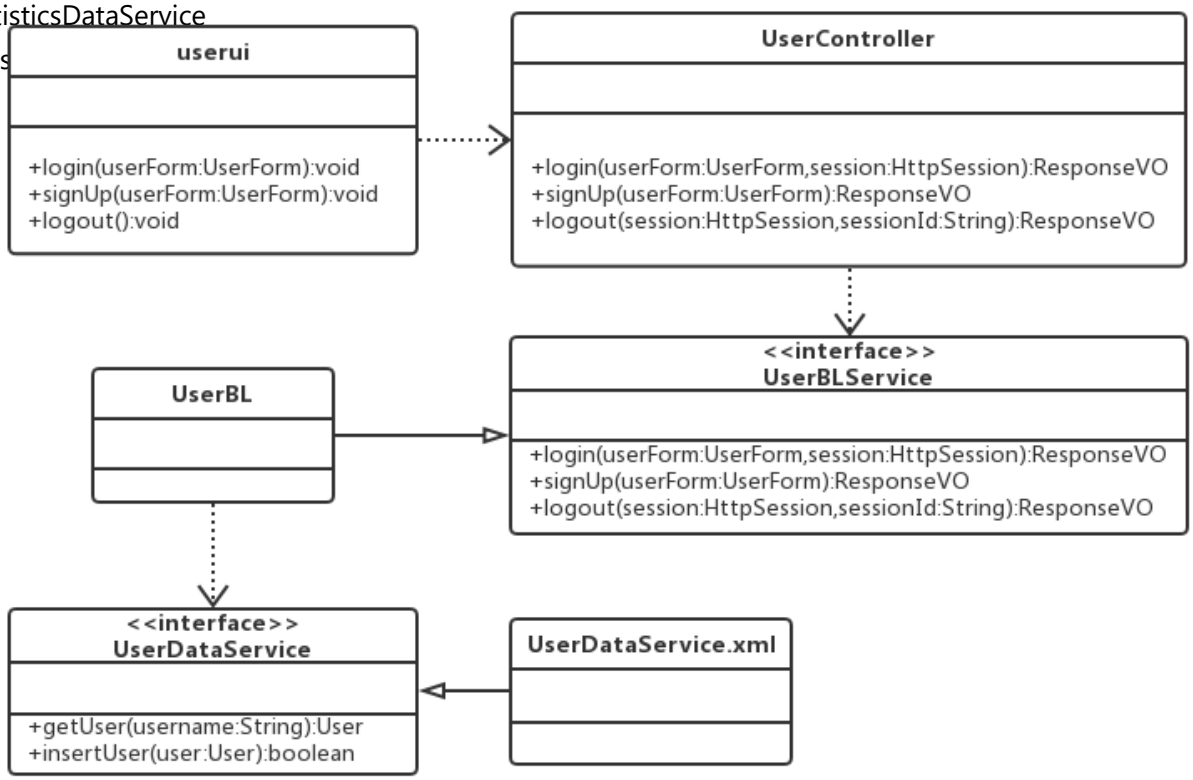

图8 User模块层之间调用的接口

5.2 用户界面层分解

根据需求，系统存在21个用户界面，界面跳转如图9所示。

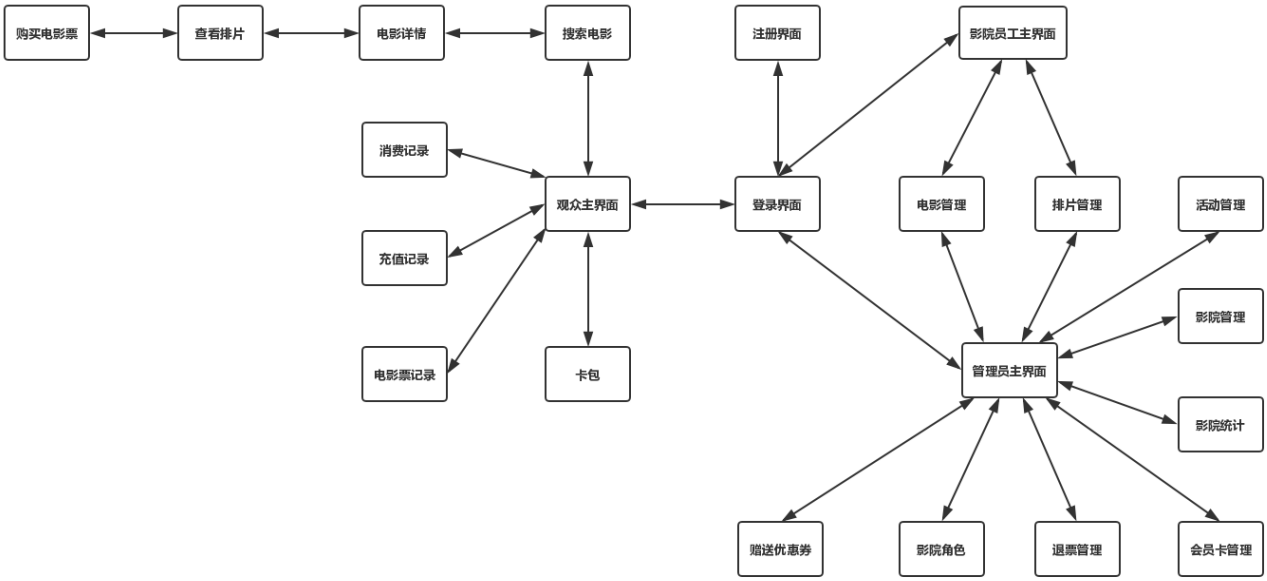

图9 用户界面跳转

服务器端和客户端的用户界面实际上是一样的，通过HTTP协议进行传输。用户界面层的设计如图10所示。

图10 用户界面层设计

5.2.1 职责

用户界面层模块的职责如表4所示。

表4 用户界面层模块的职责

模块	职责
HallUI	负责影厅信息增、删、改、查功能的界面和发送请求
UserUI	负责登录、注册、观众主界面、管理员主界面、影院角色管理的显示和发送请求
SalesUI	负责购票、退票界面，管理退票界面的现实和发送请求
StatisticsUI	负责想看电影标记，电影统计数据界面的显示和发送请求
ConsumeUI	负责充值记录，消费记录，消费详情界面的显示和发送请求
ActivityUI	负责发布优惠活动界面的显示和发送请求
CardUI	负责发布会员卡、购买会员卡、充值会员卡界面的显示和发送请求
CouponUI	负责赠送优惠券的显示和发送请求

5.2.2 接口规范

第三阶段用户界面层的接口规范如表5~12所示。

表5 hallui模块接口规范

	需要的服务（需接口）
服务名	**服务**

HallController.AddHall	添加影厅信息
HallController.UpdateHall	更新影厅信息
HallController.RemoveHall	删除影厅信息
HallController.getAvailableHalls	获取所有空闲的影厅信息列表
HallController.searchAllHall	获取所有的影厅信息列表

表6 userui模块接口规范

需要的服务（需接口）	
服务名	**服务**
UserController.login(UserForm userForm, HttpSession session)	登录系统，建立会话
UserController.registerAccount(UserForm userForm)	注册系统账号
UserController.logOut(HttpSession session)	退出登录，结束回话
UserController.checkPassword(String rawPassword, HttpSession session)	验证账号密码
UserController.editPassword(UserForm userForm)	修改账号密码
UserController.getAdaptableRoles(Integer level)	获取当前身份等级可修改的影院角色
UserController.deleteRole(Integer id)	删除影院角色
UserController.editRole(RoleVO form)	修改影院角色信息
UserController.addRole(RoleVO form)	修改影院角色

表7 salesui模块接口规范

需要的服务（需接口）	
服务名	**服务**
SalesController.buyTicketByVIPCard(TicketAndCouponVO ticketAndCouponVO)	使用会员卡购票
SalesController.lockSeat(TicketForm ticketForm)	未付款时提供锁座服务
SalesController.buyTicket(TicketAndCouponVO ticketAndCouponVO)	购票
SalesController.getTicketByUserId(int userId)	通过用户id获取用户买过的电影票
SalesController.getOccupiedSeats(int scheduleId)	通过排片id获取被锁座位
SalesController.getTicketByUserId(int userId)deleteTicket(int ticketId)	删除电影票
SalesController.cancelTicket(List ticketId)	取消锁座
SalesController.issueTicket(int ticketId)	出票
SalesController.getAllRefund()	获取所有退票策略

SalesController.publishRefund(RefundForm refundForm)	新增退票策略
SalesController.getRefundById(int refundId)	获取退票策略
SalesController.deleteRefund(int refundId)	删除退票策略

表8 statisticsui模块接口规范

需要的服务（需接口）	
服务名	**服务**
StatisticsController.likeMovie(int userId, int movieId)	标记电影为想看
StatisticsController.unlikeMovie(int userId, int movieId)	取消电影为想看
StatisticsController.getCountOfLikes(int movieId)	统计电影想看人数
StatisticsController.getLikeNumsGroupByDate(int movieId)	获得按日期统计的电影想看人数
StatisticsController.getScheduleRateByDate(Date date)	获得指定日期的排片率
StatisticsController.getTotalBoxOffice()	获得全部电影总票房
StatisticsController.getAudiencePriceSevenDays()	获得过去7天内每天客单价
StatisticsController.getMoviePlacingRateByDate(Date date)	获取所有电影某天的上座率
StatisticsController.getPopularMovies(int days, int movieNum)	获取最近days天内，票房最高的的movieNum个电影

表9 consumeui模块接口规范

需要的服务（需接口）	
服务名	**服务**
ConsumeController.getAllTopUpHistory(Integer userId)	获取用户全部的充值记录
ConsumeController.getBriefConsumeHis(Integer userId)	获取用户简略消费记录信息
ConsumeController.getConsumeHisDetail(Integer id)	获取用户消费记录详细信息
ConsumeController.addTopUpHistory(Integer userId, Double money, Double discount, Double balance, Timestamp time)	添加充值记录
ConsumeController.addConsumeHistory(Integer userId, Double money, Double discount, String consumeType,Integer type, Integer contentId)	添加消费记录
ConsumeController.getConsumeQualifiedUsers(Double totalConsume)	获取消费总额满一定值的用户信息

表10 activityui模块接口规范

需要的服务（需接口）

服务名	服务
ActivityController.publishActivity(ActivityForm activityForm)	发布一个优惠活动
ActivityController.getActivities()	获取所有优惠活动信息
ActivityController.deleteActivity(int activityId)	删除优惠活动
ActivityController.getActivityById(int activityId)	获取指定id的优惠活动

表11 cardui模块接口规范

需要的服务（需接口）

服务名	服务
CardTypeController.getCards()	获取所有会员卡的类型
CardTypeController.publishCard(CardTypeForm cardTypeForm)	发布新的会员卡的类型
CardTypeController.deleteCard(int cardId)	删除某种会员卡类型
CardTypeController.updateCard(int cardId,CardTypeForm cardTypeForm)	更新某种会员卡类型的信息
VIPCardController.addVIP(int userId,int cardTypeId)	为指定用户增加会员卡
VIPCardController.getVIP(int userId)	获取某用户的有效会员卡
VIPCardController.charge(VIPCardForm vipCardForm)	为指定会员卡充值制定金额
VIPCardController.changeVIP(int cardId,int cardTypeId)	更换指定用户的会员卡类型

表12 couponui模块接口规范

需要的服务（需接口）

服务名	服务
CouponController.getAllCoupons()	获取所有优惠券
CouponController.getUsersByConsume(double totalConsume)	获取满足一定消费总额的用户
CouponController.presentCoupon2User(PresentForm presentForm)	批量赠送优惠券

5.2.3 用户界面层设计原理

用户界面层利用HTML+CSS+JavaScript来实现。

5.3 控制层分解

控制层包括多个针对用户界面层模块的控制模块。例如，usercon负责处理userui模块下界面的请求；moviecon负责处理movieui模块下界面的请求。控制层的设计如图11所示。

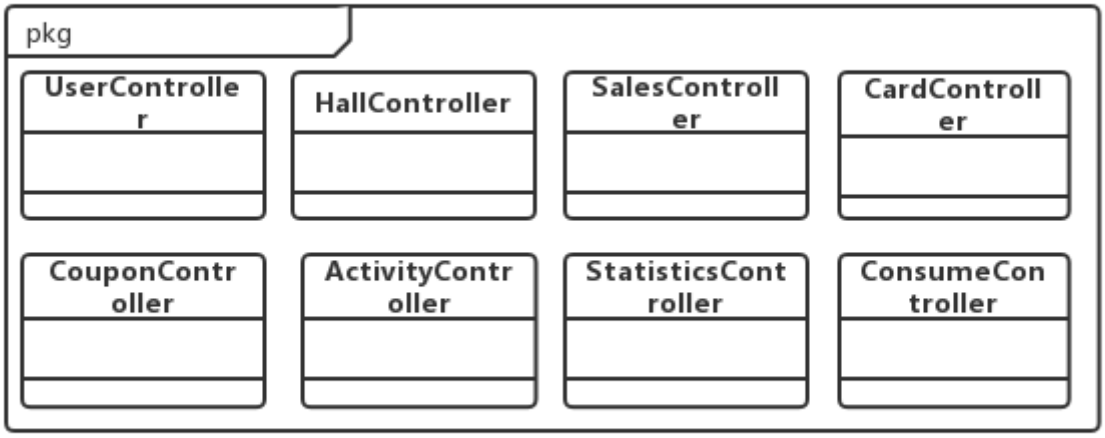

图11 控制层设计

5.3.1 职责

控制层模块的职责如表13所示。

表13 控制层模块的职责

模块	职责
HallController	负责实现与影厅信息增删改查相关的控制逻辑
UserController	负责实现与用户账户相关的控制逻辑
SalesController	负责实现与购票相关的控制逻辑
StatisticsController	负责实现与电影统计数据相关的控制逻辑
ConsumeController	负责实现与充值消费记录相关的控制逻辑
ActivityController	负责实现与优惠活动相关的控制逻辑
CardController	负责实现与会员卡相关的控制逻辑
CouponController	负责实现与优惠券相关的控制逻辑

5.3.2 接口规范

控制层的接口规范如表14~表21所示。

表14 hallcon模块的接口规范

	提供的服务（供接口）	
HallController.searchAllHall	**语法**	public ResponseVO searchAllHall()

	前置条件	GET /hall/all
	后置条件	返回所有影厅信息列表
HallController.addHall	**语法**	public ResponseVO addHall(HallVO hallVO)
	前置条件	POST /hall/add ， hallVO不为空且符合输入规则
	后置条件	根据hallVO查询是否已经存在相同的hallName，如果没有则新增一条影厅信息记录，最后返回增加影厅信息的结果
HallController.updateHall	**语法**	public ResponseVO updateHall(HallVO hallVO)
	前置条件	POST /hall/update ， hallVO不为空且符合输入规则
	后置条件	根据hallVO查询是否已经存在相同的hallName，如果没有则将原来对应的影厅信息更新，最后返回更新影厅信息的结果
HallController.removeHall	**语法**	public ResponseVO removeHall(int hallId)
	前置条件	POST /hall/remove 或者 GET /hall/remove
	后置条件	根据hallId查询是否存在对应的影厅信息记录，如果有则将对应的影厅信息删除，最后返回删除影厅信息的结果
HallController.getAvailableHalls	**语法**	public ResponseVO getAvailableHalls()

	前置条件	POST /hall/available 或者 GET /hall/available
	后置条件	返回所有空闲的影厅信息列表

需要的服务（需接口）

服务名	服务
HallBLService.searchAllHall	获取所有影厅信息列表
HallBLService.addHall	添加影厅信息
HallBLService.updateHall	更新影厅信息
HallBLService.removeHall	删除影厅信息
HallBLService.getAvailableHalls	获取所有空闲（没有排片）的影厅信息列表

表15 usercon模块的接口规范

	提供的服务（供接口）	
UserController.login	**语法**	public User login(UserForm userForm, HttpSession session)
	前置条件	@PostMapping("/login")
	后置条件	根据username查找是否存在相应的User，如果username和password相匹配，则建立session会话，最后返回登录验证的结果
UserController.registerAccount	**语法**	public ResponseVO registerAccount(UserForm userForm)
	前置条件	@PostMapping("/register")
	后置条件	根据UserForm查找是否已经存在对应的username，如果没有，则增加一条新的用户账号记录，最后返回注册的结果
UserController.logout	**语**	public ResponseVO logOut(HttpSession session)

	法	
	前置条件	@PostMapping("/logout")
	后置条件	如果sessionId是有效的，则在session中删除sessionId对应的session记录，最后返回退出登录结果
UserController.checkPassword	**语法**	public ResponseVO checkPassword(String rawPassword, HttpSession session)
	前置条件	@GetMapping("/user/checkPassword")
	后置条件	根据用户输入的密码返回验证结果
UserController.editPassword	**语法**	public ResponseVO editPassword(UserForm userForm)
	前置条件	@PostMapping("/user/edit/password")
	后置条件	根据userForm查找用户的账号并修改账号密码，返回修改的结果
UserController.getAdaptableRoles	**语法**	public ResponseVO getAdaptableRoles(Integer level)
	前置条件	@GetMapping("/role/get")
	后置条件	如果等级是有效的，根据等级查找并返回该等级可修改的影院角色
UserController.deleteRole	**语**	public ResponseVO deleteRole(Integer id)

	法	
	前置条件	@GetMapping("/role/delete")
	后置条件	如果该id是有效的，根据id查找并删除该影院角色，返回删除结果
UserController.editRole	**语法**	public ResponseVO editRole(RoleVO form)
	前置条件	@PostMapping("/role/edit")
	后置条件	修改影院角色并返回修改结果
UserController.addRole	**语法**	public ResponseVO addRole(RoleVO form)
	前置条件	@PostMapping("/role/add")
	后置条件	根据from添加影院角色，并返回添加结果

需要的服务（需接口）

服务名	**服务**
UserBLService.registerAccount(UserForm userForm)	注册账号
UserBLService.login(UserForm userForm)	用户登录，登录成功会将用户信息保存在session中
UserBLService.checkPassword(User user, String rawPassword)	检测密码是否正确
UserBLService.editPassword(UserForm userForm)	修改密码
UserBLService.getAdaptableRoles(Integer level)	获取当前身份等级可修改的影院角色(同级或更高级只可见用户名)

UserBLService.deleteRole(Integer id)	删除影院角色
UserBLService.editRole(RoleVO form)	修改角色信息
UserBLService.addRole(RoleVO form)	添加影院角色

表16 salescon模块的接口规范

	提供的服务（供接口）	
SalesController.buyTicketByVIPCard	**语法**	public ResponseVO buyTicketByVIPCard(TicketAndCouponVO ticketAndCouponVO)
	前置条件	@PostMapping("/ticket/vip/buy")
	后置条件	根据ticket查找是否存在相应的会员卡，根据couponId查找是否存在相应的优惠券，存在则完成购买电影票流程，返回购买结果
SalesController.lockSeat	**语法**	public ResponseVO lockSeat(TicketForm ticketForm)
	前置条件	@PostMapping("/ticket/lockSeat")
	后置条件	根据ticketFrom锁座，返回锁座结果
SalesController.buyTicket	**语法**	public ResponseVO buyTicket(TicketAndCouponVO ticketAndCouponVO)
	前置条件	@PostMapping("/ticket/buy")
	后置条件	根据ticketAndCouponVO查找并购买电影票，并返回购票结果
SalesController.getTicketByUserId	**语法**	public ResponseVO getTicketByUserId(int userId)
	前置	@GetMapping("/ticket/get/{userId}")

	条件	
	后置条件	根据userId查找并返回用户买过的电影票
SalesController.getOccupiedSeats	**语法**	public ResponseVO getOccupiedSeats(int scheduleId)
	前置条件	@GetMapping("/ticket/get/occupiedSeats")
	后置条件	根据scheduleId查找并返回该排片的锁座情况
SalesController.cancelTicket	**语法**	public ResponseVO cancelTicket(List ticketId)
	前置条件	@PostMapping("/ticket/cancel")
	后置条件	根据ticketId查找并删除电影票，返回删除结果
SalesController.deleteTicket	**语法**	public ResponseVO deleteTicket(int ticketId)
	前置条件	@GetMapping("/ticket/delete")
	后置条件	根据ticketId查找并退还电影票，返回退票结果
SalesController.issueTicket	**语法**	public ResponseVO issueTicket(int ticketId)
	前置	@GetMapping("/ticket/issue")

	条件	
	后置条件	根据ticketId查找并修改电影票状态为出票，返回出票结果
SalesController.getAllRefund	**语法**	public ResponseVO getAllRefund()
	前置条件	@GetMapping("/refund/all")
	后置条件	获取并返回所有退票策略
SalesController.publishRefund	**语法**	public ResponseVO publishRefund(RefundForm refundForm)
	前置条件	@PostMapping("/refund/add")
	后置条件	根据refundFrom新增一条退票策略，返回新增结果
SalesController.deleteRefund	**语法**	public ResponseVO deleteRefund(int refundId)
	前置条件	@GetMapping("/refund/delete")
	后置条件	根据refundId查找并删除退票策略，返回删除结果
SalesController.getRefundById	**语法**	public ResponseVO getRefundById(int refundId)
	前置	@GetMapping("/refund/get")

条件	
后置条件	根据refundId查找并返回退票策略

需要的服务（需接口）

服务名	服务
SalesBLService.addTicket(TicketForm ticketForm)	锁座，添加票并将状态设为未付款
SalesBLService.completeTicket(List id, int couponId)	完成购票
SalesBLService.getBySchedule(int scheduleId)	获得该场次的被锁座位和场次信息
SalesBLService.getTicketByUser(int userId)	获得用户买过的票
SalesBLService.completeByVIPCard(List id, int couponId)	使用会员卡完成购票
SalesBLService.cancelTicket(List id)	取消锁座
SalesBLService.issueTicket(int id)	出票，将票的状态改为已出票
SalesBLService.getAllRefund()	获取所有退票策略
SalesBLService.publishRefund(RefundForm refundForm)	添加一个退票策略
SalesBLService.deleteRefund(int refundId)	删除退票策略
SalesBLService.getRefundById(int refundId)	根据id获得退票策略

表17 statisticscon模块的接口规范

	提供的服务（供接口）	
StatisticsController.likeMovie	**语法**	public ResponseVO likeMovie(int movieId,int userId)
	前置条件	@PostMapping("/movie/{movieId}/like")
	后置条件	标记电影为想看
StatisticsController.unlikeMovie	**语法**	public ResponseVO unlikeMovie(int movieId,int userId)
	前置条件	@PostMapping("/movie/{movieId}/unlike")
	后置条件	取消电影为想看
StatisticsController.getMovieLikeCounts	**语法**	public ResponseVO getMovieLikeCounts(int movieId)

	前置条件	@GetMapping("/movie/{movieId}/like/count")
	后置条件	获取电影的想看人数(int)
StatisticsController.getMovieLikeCountByDate	**语法**	public ResponseVO getMovieLikeCountByDate(int movieId)
	前置条件	@GetMapping("/movie/{movieId}/like/date")
	后置条件	获取电影按日期统计的想看人数(DateLike)
StatisticsController.getScheduleRateByDate	**语法**	public ResponseVO getScheduleRateByDate(Date date)
	前置条件	@GetMapping("/statistics/scheduleRate")
	后置条件	获取当日的排片率(MovieScheduleTimeVO)
StatisticsController.getTotalBoxOffice	**语法**	public ResponseVO getTotalBoxOffice()
	前置条件	@GetMapping("/statistics/boxOffice/total")
	后置条件	获取全部电影总票房
StatisticsController.getAudiencePrice	**语法**	public ResponseVO getAudiencePrice()
	前置条件	@GetMapping("/statistics/audience/price")
	后置条件	获取客单价
StatisticsController.getMoviePlacingRateByDate	**语法**	public ResponseVO getMoviePlacingRateByDate(Date date)
	前置条件	@GetMapping("/statistics/PlacingRate")
	后置条件	获取指定日期全部电影的上座率
StatisticsController.getPopularMovies	**语法**	public ResponseVO getPopularMovies(int days,int movieNum)
	前置条件	@GetMapping("/statistics/popular/movie")
	后置	获取指定天数内票房最高的movieNum部电影

条件	
需要的服务（需接口）	
服务名	**服务**
StatisticsBLService.likeMovie(int userId, int movieId)	标记电影为想看
StatisticsBLService.unlikeMovie(int userId, int movieId)	取消电影为想看
StatisticsBLService.getCountOfLikes(int movieId)	统计电影想看人数
StatisticsBLService.getLikeNumsGroupByDate(int movieId)	获得按日期统计的电影想看人数
StatisticsBLService.getScheduleRateByDate(Date date)	获得指定日期的排片率
StatisticsBLService.getTotalBoxOffice()	获得全部电影总票房
StatisticsBLService.getAudiencePriceSevenDays()	获得过去7天内每天客单价
StatisticsBLService.getMoviePlacingRateByDate(Date date)	获取所有电影某天的上座率
StatisticsBLService.getPopularMovies(int days, int movieNum)	获取最近days天内，票房最高的的movieNum个电影

表18 consumecon模块的接口规范

提供的服务（供接口）		
ConsumeController.getAllTopupHistory	**语法**	ResponseVO getAllTopupHistory(Integer userId)
	前置条件	@GetMapping("/history/topup/get")
	后置条件	获取指定用户的充值记录
ConsumeController.getAllConsumeHistory	**语法**	ResponseVO getAllConsumeHistory(Integer userId)
	前置条件	@GetMapping("/history/consume/get")
	后置条件	获取指定用户的消费记录
ConsumeController.getConsumeHisDetail	**语法**	ResponseVO getConsumeHisDetail(Integer id)
	前置条件	@GetMapping("/history/consume/get/detail")
	后置条件	获取消费记录详细信息

需要的服务（需接口）	
服务名	**服务**

ConsumeBLService.getAllTopUpHistory(Integer userId)	获取用户全部的充值记录
ConsumeBLService.getBriefConsumeHis(Integer userId)	获取用户简略消费记录信息
ConsumeBLSerivce.getConsumeHisDetail(Integer id)	获取用户消费记录详细信息
ConsumeBLSerivce.addTopUpHistory(Integer userId, Double money, Double discount, Double balance, Timestamp time)	添加充值记录
ConsumeBLSerivce.addConsumeHistory(Integer userId, Double money, Double discount, String consumeType,Integer type, Integer contentId)	添加消费记录
ConsumeBLSerivce.getConsumeQualifiedUsers(Double totalConsume)	获取消费总额满一定值的用户信息

表19 activitycon模块的接口规范

提供的服务（供接口）		
ActivityController.publishActivity	**语法**	public ResponseVO publishActivity(ActivityForm activityForm)
	前置条件	@PostMapping("/publish")
	后置条件	发布一个优惠活动
ActivityController.getActivities	**语法**	public ResponseVO getActivities()
	前置条件	@GetMapping("/getAll")
	后置条件	获取所有优惠活动信息
ActivityController.deleteActivity	**语法**	public ResponseVO deleteActivity(int activityId)
	前置条件	@GetMapping("/delete")
	后置条件	删除优惠活动
ActivityController.getActivityById	**语法**	public ResponseVO getActivityById(int activityId)
	前置条件	@GetMapping("/get")
	后置条件	获取指定id的优惠活动
需要的服务（需接口）		

服务名	服务
ActivityBLService.publishActivity(ActivityForm activityForm)	发布一个优惠活动
ActivityBLService.getActivities()	获取所有优惠活动信息
ActivityBLService.deleteActivity(int activityId)	删除优惠活动
ActivityBLService.getActivityById(int activityId)	获取指定id的优惠活动

表20 cardcon模块的接口规范

	提供的服务（供接口）	
CardTypeController.getCards	**语法**	public ResponseVO getCards()
	前置条件	@GetMapping("/get")
	后置条件	获取所有会员卡的类型
CardTypeController.publishCard	**语法**	public ResponseVO publishCard(CardTypeForm cardTypeForm)
	前置条件	@PostMapping("/publish")
	后置条件	发布新的会员卡的类型
CardTypeController.deleteCard	**语法**	public ResponseVO deleteCard(int cardId)
	前置条件	@GetMapping("/delete/{cardId}")
	后置条件	删除某种会员卡类型
CardTypeController.updateCard	**语法**	public ResponseVO updateCard(int cardId,CardTypeForm cardTypeForm)
	前置条件	@PostMapping("/update/{cardId}")
	后置条件	更新某种会员卡类型的信息
VIPCardController.addVIP	**语法**	public ResponseVO addVIP(int userId,int cardTypeId)
	前置条件	@PostMapping("/add/{userId}/{cardTypeId}")
	后置条件	为指定用户增加会员卡
VIPCardController.getVIP	**语法**	public ResponseVO getVIP(int userId)

	前置条件	@GetMapping("/get/{userId}")
	后置条件	获取某用户的有效会员卡
VIPCardController.charge	**语法**	public ResponseVO charge(VIPCardForm vipCardForm)
	前置条件	@PostMapping("/charge")
	后置条件	为指定会员卡充值制定金额
VIPCardController.changeVIP	**语法**	public ResponseVO changeVIP(int cardId,int cardTypeId)
	前置条件	@PostMapping("/change/{cardId}/{cardTypeId}")
	后置条件	更换指定用户的会员卡类型

需要的服务（需接口）

服务名	**服务**
CardTypeBLService.getCards()	获取所有会员卡类型
CardTypeBLService.publishCard(CardTypeForm cardTypeForm)	发布会员卡类型
CardTypeBLService.deleteCard(int cardId)	删除会员卡类型
CardTypeBLService.updateCard(int cardId, CardTypeForm cardTypeForm)	更新会员卡类型信息
VIPCardBLService.addVIPCard(int userId, int cardTypeId)	为指定用户增加会员卡
VIPCardBLService.getCardByUserId(int userId)	获取某用户的有效会员卡
VIPCardBLService.charge(VIPCardForm vipCardForm)	为指定会员卡充值制定金额
VIPCardBLService.changeVIPCard(int cardId, int cardTypeId)	更换指定用户的会员卡类型

表21 couponcon模块的接口规范

	提供的服务（供接口）	
CouponController.getAllCoupons	**语法**	public ResponseVO getAllCoupons()
	前置条件	@GetMapping("get/coupons")
	后置条件	获取所有优惠券
CouponController.getUsersByConsume	**语法**	public ResponseVO getUsersByConsume(double totalConsume)
	前置条件	@GetMapping("get/users")

	后置条件	获取满足一定消费总额的用户
CouponController.presentCoupon2User	**语法**	public ResponseVO presentCoupon2User(PresentForm presentForm)
	前置条件	@PostMapping("present")
	后置条件	批量赠送优惠券

需要的服务（需接口）

服务名	**服务**
CouponBLService.getAllCoupons()	获取所有优惠券
CouponBLService.getUsersByConsume(double totalConsume)	获取满足一定消费总额的用户
CouponBLService.presentCoupon2User(PresentForm presentForm)	批量赠送优惠券

5.4 业务逻辑层分解

业务逻辑层包括多个针对控制层模块的业务逻辑模块。例如，usercon会根据用户请求调用userbl模块的服务。业务逻辑层的设计如图12所示。

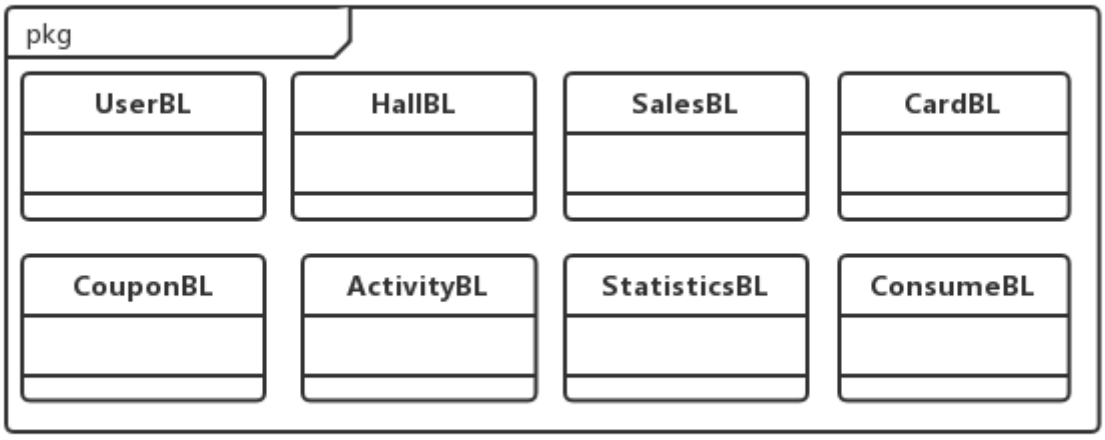

图12 业务逻辑层设计

5.4.1 职责

业务逻辑层的模块职责如表22所示。

表22 业务逻辑层模块的职责

模块	**职责**

HallBL	负责实现对应的控制层需要的与影厅信息增删改查功能相关的业务逻辑服务
ScheduleBL	负责实现对应的控制层需要的与排片信息相关的业务逻辑服务
UserBL	负责实现对应的控制层需要的与用户相关的业务逻辑服务
SalesBL	负责实现对应的控制层需要的与购票相关的业务逻辑服务
StatisticsBL	负责实现对应的控制层需要的与电影统计数据相关的业务逻辑服务
ConsumeBL	负责实现对应的控制层需要的与充值消费记录相关的业务逻辑服务
ActivityBL	负责实现对应的控制层需要的与优惠活动相关的业务逻辑服务
CardBL	负责实现对应的控制层需要的与会员卡相关的业务逻辑服务
CouponBL	负责实现对应的控制层需要的与优惠券相关的业务逻辑服务

5.4.2 接口规范

业务逻辑层的接口规范如表23~表31所示。

表23 hallbl模块的接口规范

	提供的服务（供接口）	
HallBLService.searchAllHall	**语法**	public ResponseVO searchAllHall()
	前置条件	无
	后置条件	返回所有影厅信息列表
HallBLService.addHall	**语法**	public ResponseVO addHall(HallVO hallVO)
	前置条件	hallVO不为空且符合输入规则
	后置条件	根据hallId查询是否存在对应的影厅信息记录，如果有则将对应的影厅信息删除，最后返回删除影厅信息的结果
HallBLService.updateHall	**语法**	public ResponseVO updateHall(HallVO hallVO)

	前置条件	hallVO不为空且符合输入规则
	后置条件	根据hallVO查询是否已经存在相同的hallName，如果没有则将原来对应的影厅信息更新，最后返回更新影厅信息的结果
HallBLService.removeHall	**语法**	public ResponseVO removeHall(int hallId)
	前置条件	无
	后置条件	根据hallId查询是否存在对应的影厅信息记录，如果有则将对应的影厅信息删除，最后返回删除影厅信息的结果
HallBLService.getAvailableHalls	**语法**	public ResponseVO getAvailableHalls()
	前置条件	无
	后置条件	获取所有空闲（没有排片）的影厅信息列表

需要的服务（需接口）

服务名	**服务**
HallDataService.selectAllHall	获取所有影厅信息列表
HallDataService.checkHallName	查询是否存在相同名字的影厅记录
HallDataService.addHall	添加影厅信息
HallDataService.updateHall	更新影厅信息
HallDataService.removeHallById	根据影厅id删除影厅信息
HallDataService.getHallsExcept	根据影厅id列表，获取影厅id不属于该id列表的影厅信息列表
HallDataService.selectHallById	根据影厅id获取影厅信息
ScheduleService.getScheduledHalls	获取当前已经有排片的影厅id列表

表24 schedulebl模块的接口规范

ScheduleServiceForBl.getScheduledHalls	**提供的服务（供接口）**	
	语法	public List<Integer> getScheduledHalls()
	前置条件	无
	后置条件	返回所有已排片的影厅信息列表
	需要的服务（需接口）	
服务名	**服务**	
ScheduleDataService.getHallsInSchedules	获取数据库中当前没有排片（当前时间晚于电影结束时间）的所有影厅id列表	

表25 userbl模块的接口规范

	提供的服务（供接口）	
UserBL.login	**语法**	public User login(UserForm userForm)
	前置条件	userName符合输入条件
	后置条件	根据username查找是否存在相应的User，如果username和password相匹配，则建立session会话，最后返回登录验证的结果
UserBL.registerAccount	**语法**	public ResponseVO registerAccount(UserForm userForm)
	前置条件	userFrom符合输入条件
	后置条件	根据UserForm查找是否已经存在对应的username，如果没有，则增加一条新的用户账号记录，最后返回注册的结果
UserBL.checkPassword	**语法**	public ResponseVO checkPassword(User user, String rawPassword)
	前置条件	rawPassword符合输入条件
	后	根据用户输入的密码返回验证结果

	置条件	
UserBL.editPassword	**语法**	public ResponseVO editPassword(UserForm userForm)
	前置条件	用户账号存在
	后置条件	根据userForm查找用户的账号并修改账号密码，返回修改的结果
UserBL.getAdaptableRoles	**语法**	public ResponseVO getAdaptableRoles(Integer level)
	前置条件	输入的等级是有效等级
	后置条件	如果等级是有效的，根据等级查找并返回该等级可修改的影院角色
UserBL.deleteRole	**语法**	public ResponseVO deleteRole(Integer id)
	前置条件	id符合输入条件
	后置条件	如果该id是有效的，根据id查找并删除该影院角色，返回删除结果
UserBL.editRole	**语法**	public ResponseVO editRole(RoleVO form)
	前置条件	from符合输入条件
	后	修改影院角色并返回修改结果

	置条件	
UserBL.addRole	**语法**	public ResponseVO addRole(RoleVO form)
	前置条件	from符合输入条件
	后置条件	根据from添加影院角色，并返回添加结果

需要的服务（需接口）

服务名	**服务**
UserDataService.createNewAccount(String username, String password, Integer auth)	注册账号
UserDataService.getAccountByName(String username)	根据用户名查找账号
UserDataService.updatePassword(UserForm userForm)	修改密码
UserDataService.getCinemaRoles()	获取全部影院角色
UserDataService.deleteRoleById(Integer id)	删除影院角色
UserDataService.updateRoleById(Integer id, String username, String password, Integer auth)	更新角色信息

表26 salesbl模块的接口规范

	提供的服务（供接口）	
SalesBL.completeByVIPCard	**语法**	public ResponseVO completeByVIPCard(List id, int couponId)
	前置条件	id不为空
	后置条件	根据ticket查找是否存在相应的会员卡，根据couponId查找是否存在相应的优惠券，存在则完成购买电影票流程，返回购买结果
SalesBL.addTicket	**语法**	public ResponseVO addTicket(TicketForm ticketForm)
	前	ticketFrom符合条件

	置条件	
	后置条件	根据ticketFrom锁座，返回锁座结果
SalesBL.completeTicket	**语法**	public ResponseVO completeTicket(List id, int couponId)
	前置条件	id不为空
	后置条件	根据ticketAndCouponVO查找并购买电影票，并返回购票结果
SalesBL.getTicketByUser	**语法**	public ResponseVO getTicketByUser(int userId)
	前置条件	userId有效
	后置条件	根据userId查找并返回用户买过的电影票
SalesBL.getBySchedule	**语法**	public ResponseVO getBySchedule(int scheduleId)
	前置条件	scheduleId是有效id
	后置条件	根据scheduleId查找并返回该排片的锁座情况
SalesBL.cancelTicket	**语法**	public ResponseVO cancelTicket(List id)
	前	id不为空

	置条件	
	后置条件	根据ticketId查找并删除电影票，返回删除结果
SalesBL.issueTicket	**语法**	public ResponseVO issueTicket(int id)
	前置条件	id为有效电影票id
	后置条件	根据ticketId查找并修改电影票状态为出票，返回出票结果
SalesBL.getAllRefund	**语法**	public ResponseVO getAllRefund()
	前置条件	无
	后置条件	获取并返回所有退票策略
SalesBL.publishRefund	**语法**	public ResponseVO publishRefund(RefundForm refundForm)
	前置条件	refundFrom符合输入条件
	后置条件	根据refundFrom新增一条退票策略，返回新增结果
SalesBL.deleteRefund	**语法**	public ResponseVO deleteRefund(int refundId)
	前	refundId存在

	置条件	
	后置条件	根据refundId查找并删除退票策略，返回删除结果
SalesBL.getRefundById	**语法**	public ResponseVO getRefundById(int refundId)
	前置条件	refundId存在
	后置条件	根据refundId查找并返回退票策略

需要的服务（需接口）

服务名	**服务**
SalesDataService.insertTicket(Ticket ticket)	添加单张电影票
SalesDataService.insertTickets(List tickets)	添加多张电影票
SalesDataService.deleteTicket(int ticketId)	删除电影票
SalesDataService.selectTicketByUser(int userId)	获得用户买过的票
SalesDataService.deleteLockedTicket(int userId, int scheduleId)	删除被锁座位的票
SalesDataService.updateTicketState(int ticketId, int state)	更新电影票状态
SalesDataService.updateTicketActualPay(int ticketId, double actualPay)	更新票的实际支付金额
SalesDataService.selectTicketsBySchedule(int scheduleId)	根据排片获取电影票
SalesDataService.selectTicketByScheduleIdAndSeatint scheduleId, int columnIndex, int rowIndex)	根据排片和座位获取电影票
SalesDataService.selectTicketById(int id)	根据id获取电影票
SalesDataService.selectRefund()	获得所有退票策略
SalesDataService.insertRefund(Refund refund)	新增退票策略
SalesDataService.deleteRefundById(int refundId)	根据id删除一条退票策略
SalesDataService.selectRefundByMovieId(int movieId)	根据电影获取退票策略
SalesDataService.selectRefundById(int refundId)	根据id获取退票策略

表27 statisticsbl模块的接口规范

	提供的服务（供接口）	
StatisticsBL.likeMovie	**语法**	ResponseVO likeMovie(int userId, int movieId)
	前置条件	无
	后置条件	标记电影为想看
StatisticsBL.unLikeMovie	**语法**	ResponseVO unlikeMovie(int userId, int movieId)
	前置条件	无
	后置条件	取消电影为想看
StatisticsBL.getCountOfLikes	**语法**	ResponseVO getCountOfLikes(int movieId)
	前置条件	无
	后置条件	统计想看电影的人数
StatisticsBL.getLikeNumsGroupByDate	**语法**	ResponseVO getLikeNumsGroupByDate(int movieId)
	前置条件	无
	后置条件	获得电影每日的想看人数
StatisticsBL.getScheduleRateByDate	**语法**	ResponseVO getScheduleRateByDate(Date date)
	前置条件	无
	后置条件	获取某日各影片排片率统计数据
StatisticsBL.getTotalBoxOffice	**语法**	ResponseVO getTotalBoxOffice()
	前置条件	无
	后置条件	获取所有电影的累计票房(降序排序，且包含已下架的电影)
StatisticsBL.getAudiencePriceSevenDays	**语法**	ResponseVO getAudiencePriceSevenDays()
	前置条件	无

	后置条件	获得过去7天内每天客单价
StatisticsBL.getMoviePlacingRateByDate	**语法**	ResponseVO getMoviePlacingRateByDate(Date date)
	前置条件	无
	后置条件	获取所有电影某天的上座率
StatisticsBL.getPopularMovies	**语法**	ResponseVO getPopularMovies(int days, int movieNum)
	前置条件	无
	后置条件	获取最近days天内，票房最高的的movieNum个电影

需要的服务（需接口）

服务名	**服务**
StatisticsDataService.insertOneLike(int movieId,int userId)	插入一条想看记录
StatisticsDataService.deleteOneLike(int movieId,int userId)	删除一条想看记录
StatisticsDataService.selectLikeNums(int movieId)	根据movieId查看电影的想看人数
StatisticsDataService.selectLikeMovie(int movieId,int userId)	根据movieId和userId查找特定想看记录
StatisticsDataService.selectMovieScheduleTimes(Date date, Date nextDate)	查询date当天每部电影的排片次数
StatisticsDataService.getDateLikeNum(int movieId)	根据movieId获取按日期统计的想看人数
StatisticsDataService.selectMovieTotalBoxOffice()	查询所有电影的总票房（包括已经下架的，降序排列）
StatisticsDataService.selectAudiencePrice(Date date,Date nextDate)	查询date当天每个客户的购票金额
StatisticsDataService.selectPlacingRate(Date date,Date nextDate)	查询date当天各电影上座率
StatisticsDataService.selectRecentTotalBoxOffice(Date startDate,Date today,int num)	查询startDate到today日期内票房最高的num部电影

表28 consumebl模块的接口规范

ConSumeBL.getAllTopUpHistory	**提供的服务（供接口）**	
	语法	ResponseVO getAllTopUpHistory(Integer userId)
	前置	无

	条件	
	后置条件	获取用户全部的充值记录
ConSumeBL.getBriefConsumeHis	**语法**	ResponseVO getBriefConsumeHis(Integer userId)
	前置条件	无
	后置条件	获取用户简略消费记录信息
ConSumeBL.getConsumeHisDetail	**语法**	ResponseVO getConsumeHisDetail(Integer id)
	前置条件	无
	后置条件	获取用户消费记录详细信息
ConSumeBL.addTopUpHistory	**语法**	ResponseVO addTopUpHistory(Integer userId, Double money, Double discount, Double balance, Timestamp time)
	前置条件	无
	后置条件	添加充值记录
ConSumeBL.addConsumeHistory	**语法**	ResponseVO addConsumeHistory(Integer userId, Double money, Double discount, String consumeType,Integer type, Integer contentId)

	前置条件	无
	后置条件	添加消费记录
ConSumeBL.getConsumeQualifiedUsers	**语法**	ResponseVO getConsumeQualifiedUsers(Double totalConsume)
	前置条件	无
	后置条件	获取消费总额满一定值的用户信息

需要的服务（需接口）

服务名	**服务**
ConsumeDateService.getTopUpHistoryByUserId(Integer userId)	根据userId获取此用户的充值记录列表，若无记录则返回空
ConsumeDateService.getConsumeHistoryByUserId(Integer userId)	根据userId获取此用户的消费记录列表，若无记录则返回空
ConsumeDateService.getConsumeHistoryById(Integer id)	根据id获取消费记录
ConsumeDateService.insertTopUpHistory(Integer userId, Double money,Double discount,Double balance,Timestamp time)	插入新的充值记录
ConsumeDateService.insertConsumeHistory(Integer userId, Double money,Double discount,String consumeType,Integer type,Integer contentId)	插入新的消费记录
ConsumeDateService.selectConsumeQulifiedUsers(Double totalConsume)	获得消费总额大于totalConsume的用户
HallServiceForBl.getHallById(int id)	获得指定影厅信息
MovieServiceForBl.getMovieById(int id)	获得指定电影信息
ScheduleServiceForBl.getScheduleItemById(int id)	获得指定排片信息
VipService.getCardById(int id)	获得指定会员卡信息
TicketServiceForBl.getTicketById(int id)	获得指定电影票信息

表29 activitybl模块的接口规范

	提供的服务（供接口）	
ActivityBL.publishActivity	**语法**	public ResponseVO publishActivity(ActivityForm activityForm)
	前置条件	activityForm格式正确
	后置条件	发布一个优惠活动
ActivityBL.getActivities	**语法**	public ResponseVO getActivities()
	前置条件	无
	后置条件	获取所有优惠活动信息
ActivityBL.deleteActivity	**语法**	public ResponseVO deleteActivity(int activityId)
	前置条件	activityId格式正确
	后置条件	删除优惠活动
ActivityBL.getActivityById	**语法**	public ResponseVO getActivityById(int activityId)
	前置条件	activityId格式正确
	后置条件	获取指定id的优惠活动

需要的服务（需接口）

服务名	服务
ActivityDataService.insertActivity(Activity activity)	在数据库中插入一个优惠活动
ActivityDataService.insertActivityAndMovie(int activityId,List<Integer> movieId)	在数据库中插入一条活动和电影的关系
ActivityDataService.selectActivities()	返回所有优惠活动信息
ActivityDataService.selectActivitiesByMovie(int movieId)	返回特定电影的优惠活动信息
ActivityDataService.selectById(int id)	获取指定id的优惠活动
ActivityDataService.deleteActivityById(int id)	删除指定的优惠活动
ActivityDataService.deleteActivityAndMovie(int id)	根据活动id删除活动和电影关系

表30 cardbl模块的接口规范

	提供的服务（供接口）	
CardTypeBL.getCards	**语法**	public ResponseVO getCards()
	前置条件	无
	后置条件	获取所有会员卡类型
CardTypeBL.publishCard	**语法**	public ResponseVO publishCard(CardTypeForm cardTypeForm)
	前置条件	cardTypeForm信息格式正确

	后置条件	发布会员卡类型
CardTypeBL.deleteCard	**语法**	public ResponseVO deleteCard(int cardId)
	前置条件	cardId格式正确
	后置条件	删除会员卡类型
CardTypeBL.updateCard	**语法**	public ResponseVO updateCard(int cardId, CardTypeForm cardTypeForm)
	前置条件	cardId和cardTypeForm格式正确
	后置条件	更新会员卡类型信息
VIPCardBL.addVIPCard	**语法**	public ResponseVO addVIPCard(int userId, int cardTypeId)
	前置条件	userId和cardTypeId格式正确
	后置条件	为指定用户增加会员卡
VIPCardBL.getCardByUserId	**语法**	public ResponseVO getCardByUserId(int userId)
	前置条件	userId格式正确
	后置条件	获取某用户的有效会员卡
VIPCardBL.charge	**语法**	public ResponseVO charge(VIPCardForm vipCardForm)
	前置条件	vipCardForm格式正确
	后置条件	为指定会员卡充值制定金额
VIPCardBL.changeVIPCard	**语法**	public ResponseVO changeVIPCard(int cardId, int cardTypeId)
	前置条件	cardId和carTypeId格式真确
	后置条件	更换指定用户的会员卡类型

需要的服务（需接口）

服务名	**服务**
CardTypeDataService.selectAllCards()	获取所有会员卡类型

CardTypeDataService.insertOneCard(CardType cardType)	在数据库中插入会员卡类型
CardTypeDataService.deleteCardById(int cardId)	在数据库中删除会员卡类型，将会员卡类型状态设为0
CardTypeDataService.updateCardById(int cardId, CardType cardType)	更新会员卡类型信息
VIPCardDataService.insertOneCard(VIPCard vipCard)	为指定用户增加会员卡
VIPCardDataService.selectCardByUserId(int userId)	获取某用户的有效会员卡
VIPCardDataService.updateCardBalance(int id, double balance)	为指定会员卡更新余额
VIPCardDataService.selectCardById(int id)	获取指定id的会员卡
VIPCardDataService.deleteCardById(int cardId)	从数据库中删除指定id的会员卡，将状态设置为0

表31 couponbl模块的接口规范

	提供的服务（供接口）	
CouponBL.getAllCoupons	**语法**	public ResponseVO getAllCoupons()
	前置条件	管理员准备赠送优惠券
	后置条件	获取所有优惠券
CouponBL.getUsersByConsume	**语法**	public ResponseVO getUsersByConsume(double totalConsume)
	前置条件	totalConsume符合输入规范
	后置条件	获取满足一定消费总额的用户
CouponBL.presentCoupon2User	**语法**	public ResponseVO presentCoupon2User(PresentForm presentForm)
	前置条件	presentForm符合输入规范
	后置条件	批量赠送优惠券
CouponBL.getCouponsByUser	**语法**	public ResponseVO getCouponsByUser(int userId)
	前置条件	userId存在
	后置条件	返回用户拥有的有效优惠券

CouponBL.addCoupon	**语法**	public ResponseVO addCoupon(CouponForm couponForm)
	前置条件	couponForm符合输入规范
	后置条件	发布新优惠券

需要的服务（需接口）

服务名	服务
ConsumeBL.getConsumeQualifiedUsers(double totalConsume)	获取满足一定消费总额的用户
CouponDataService.selectAllCoupons()	获取所有有效优惠券
CouponDataService.insertCouponUser(int couponId,int userId)	赠送某种优惠券给某个用户
CouponDataService.insertCoupon(Coupon coupon)	在数据库中插入优惠券
CouponDataService.selectCouponByUser(int userId)	获取用户拥有的所有有效优惠券

5.5 数据层分解

数据层主要给业务逻辑层提供数据访问服务，包括对于持久化数据的增、删、改、查。例如，业务逻辑模块userbl需要的服务由userdataservice接口提供。userdataservice接口通过mybatis绑定同名xml文件实现。持久化数据的保存通过mysql数据库的形式。数据层模块的描述具体设计如图13~19所示。

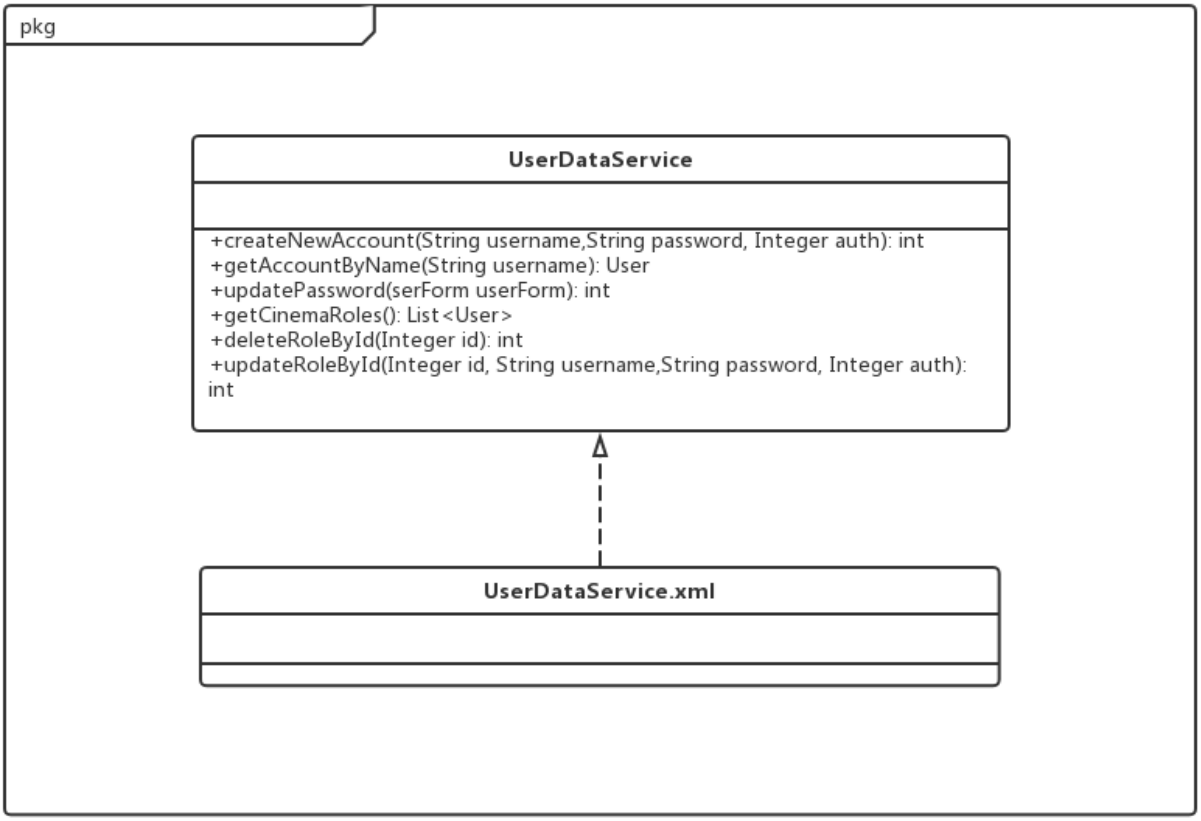

图13 userdata模块设计图

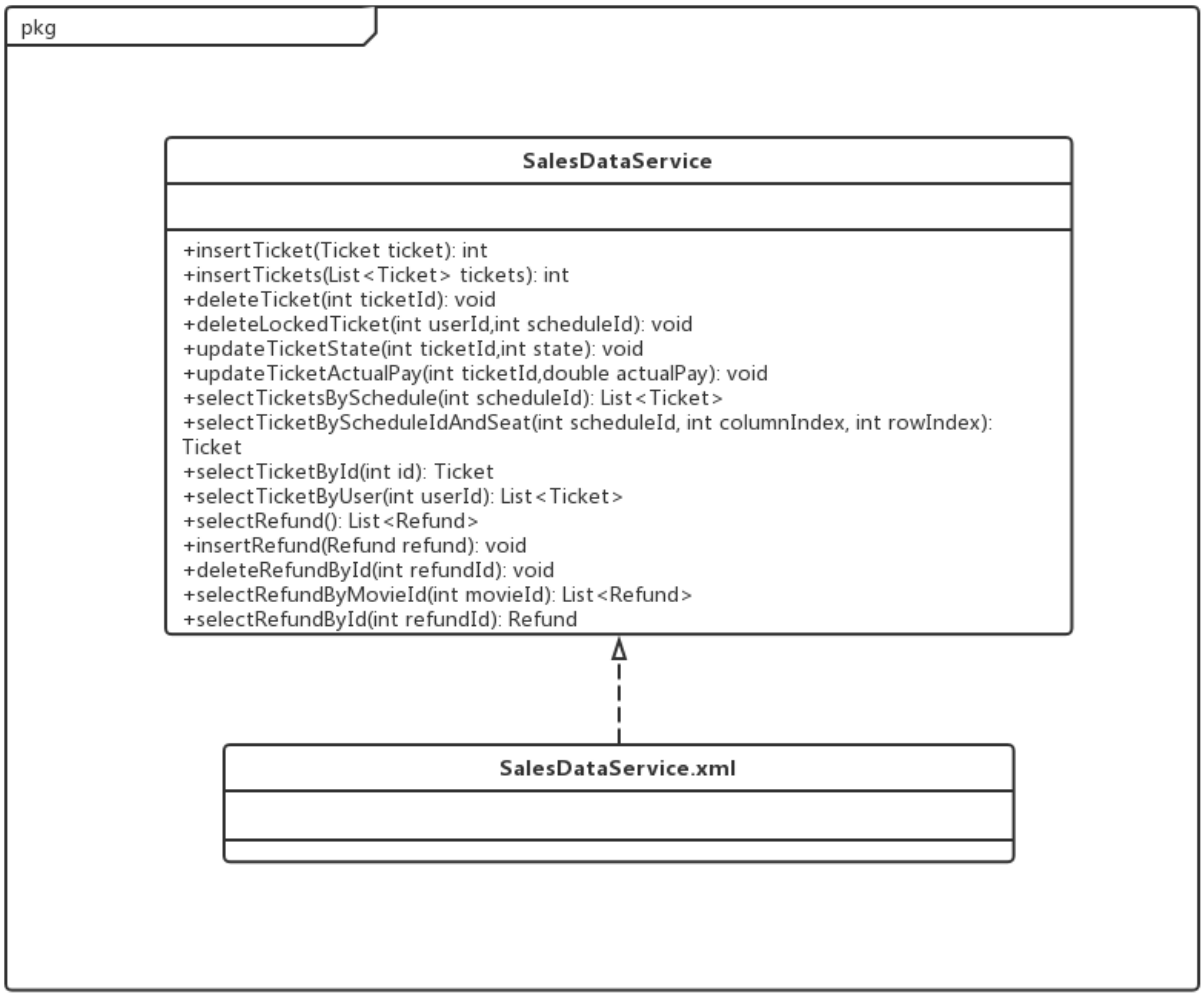

图14 salesdata模块设计图

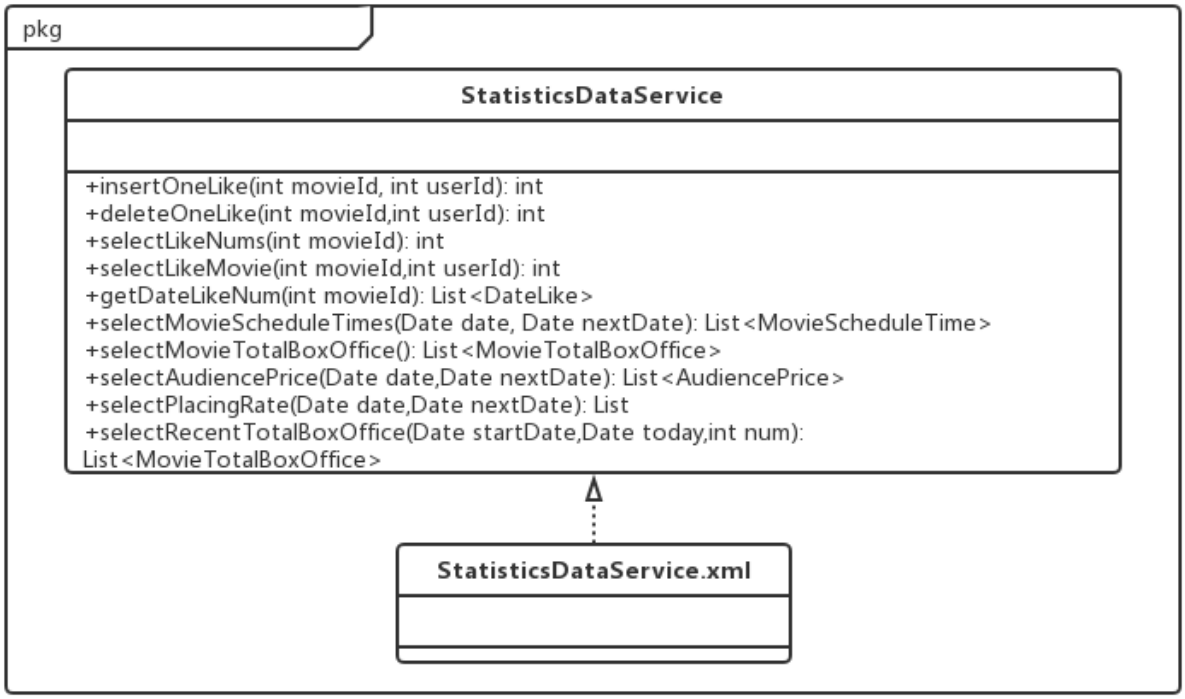

图15 statisticsdata模块设计图

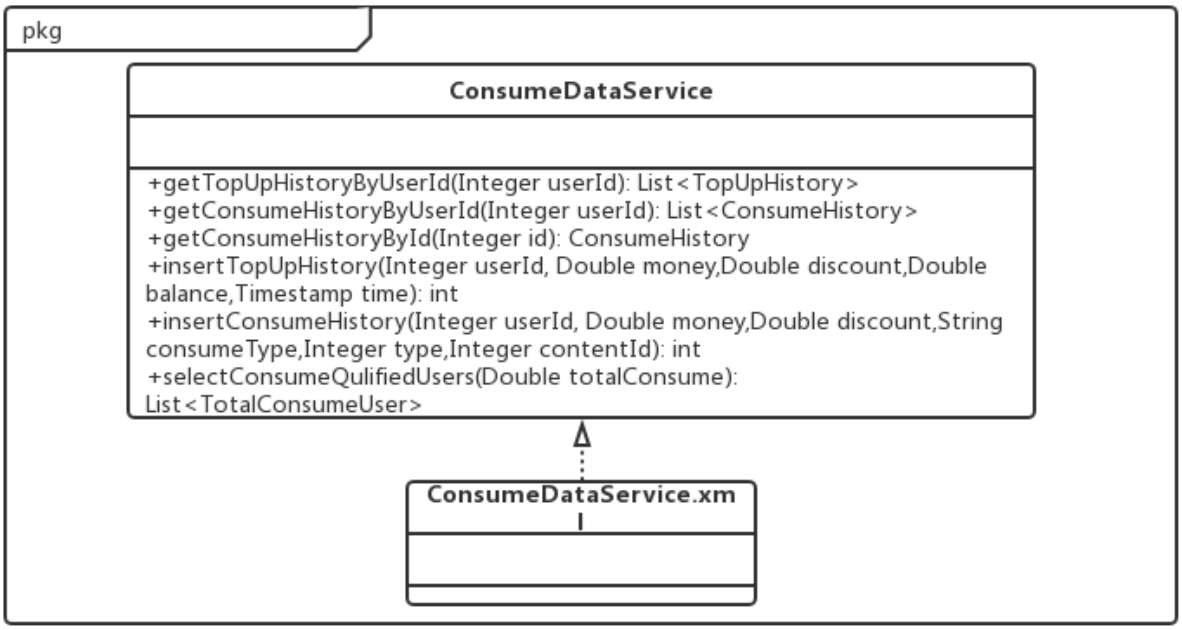

图16 consumedata模块设计图

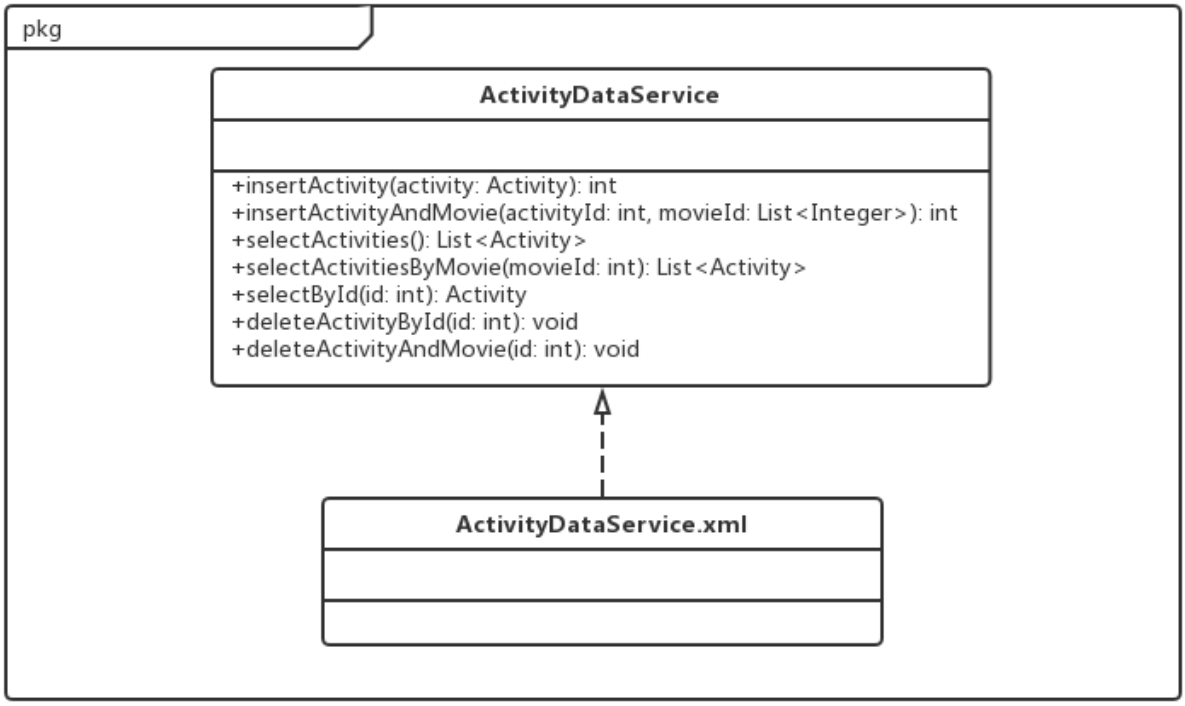

图17 activitydata模块设计图

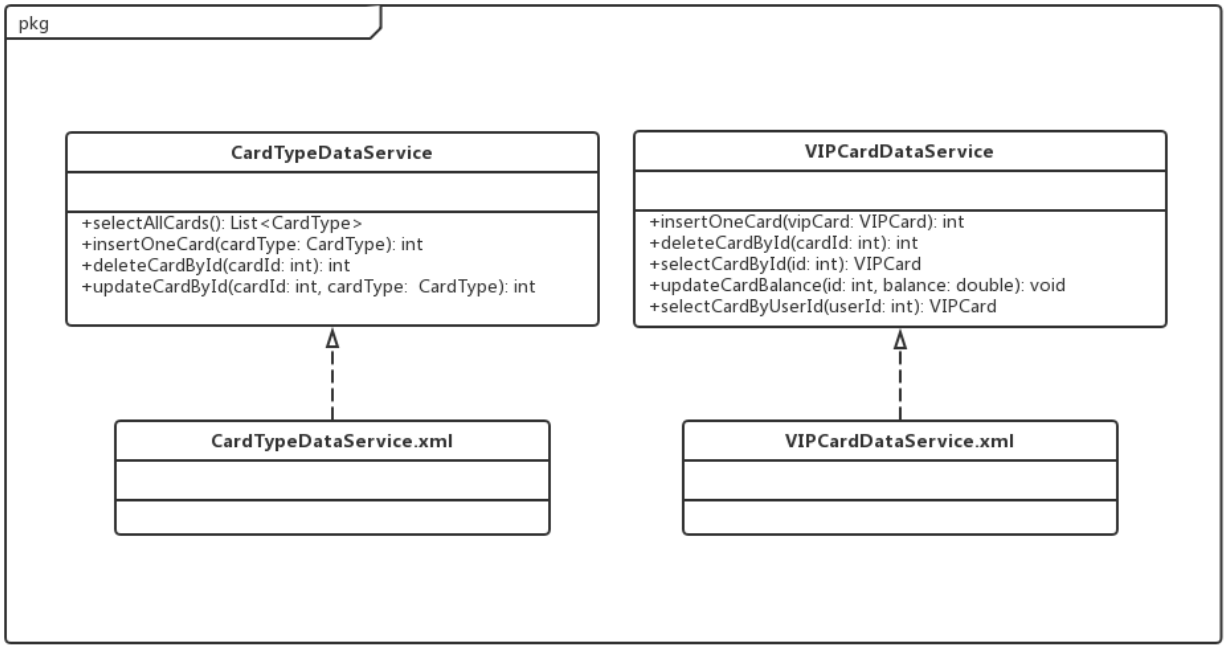

图18 carddata模块设计图

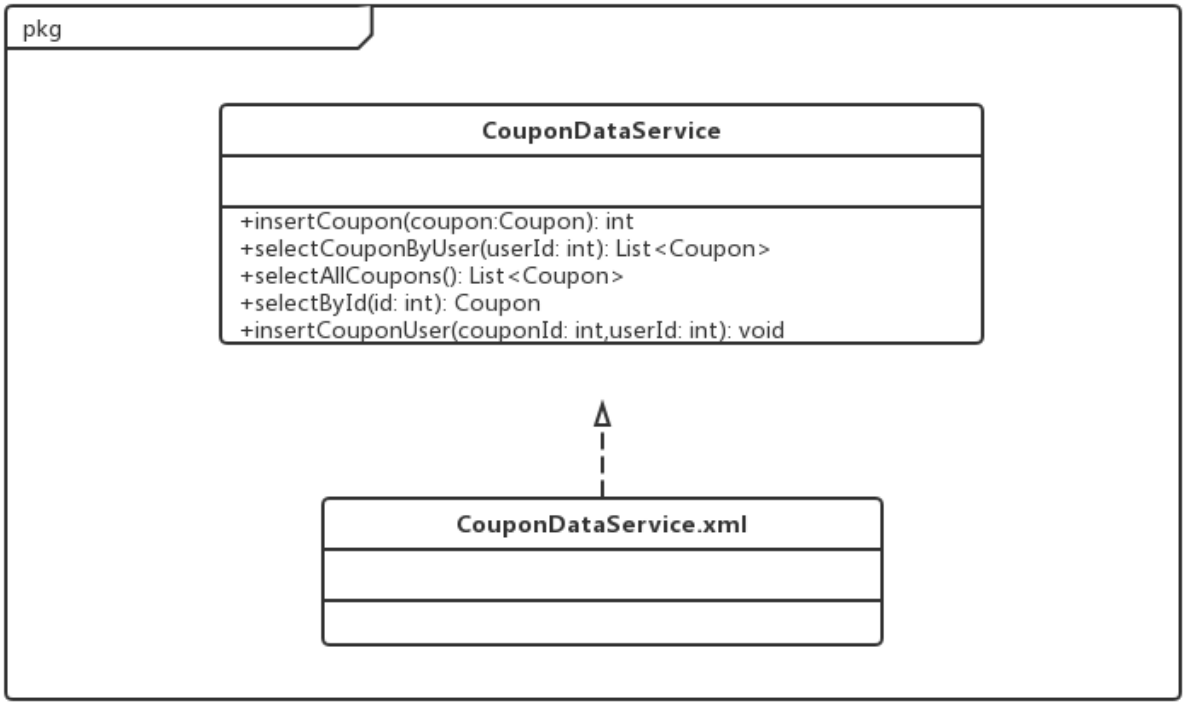

图19 coupondata模块设计图

5.5.1 职责

数据层模块的职责如表32所示。

表32 数据层模块的职责

模块	职责
HallData	提供与影厅信息数据相关的增、删、改、查服务
ScheduleData	提供与排片信息数据相关的增、删、改、查服务
UserData	提供用户数据的插入、查询服务
SalesData	提供电影票数据及退票策略数据的插入、查询服务
StatisticsData	提供电影统计数据获取，添加，删除的服务
ConsumeDate	提供消费充值记录的插入，查询的服务
ActivityData	提供与优惠活动相关的增、删、改、查服务
CardData	提供与会员卡相关的插入、修改、查询服务
CouponData	提供与优惠券相关的增、删、改、查服务

5.5.2 接口规范

数据层模块的接口规范如表33~41所示。

表33 scheduledata模块的接口规范

ScheduleDataService.getHallsInSchedules	**提供的服务（供接口）**	
	语法	ScheduleDataService.getHallsInSchedules()
	前置条件	无
	后置条件	获取数据库中当前没有排片（当前时间晚于电影结束时间）的所有影厅id列表并返回

表34 halldata模块的接口规范

HallDataService.selectAllHall	**提供的服务（供接口）**	
	语法	public List<Hall> selectAllHall()
	前置条件	无
	后置条件	获取数据库中所有影厅记录列表并返回
HallDataService.selectHallById	**语法**	public Hall selectHallById(int hallId)
	前置条件	无
	后置条件	根据hallId获取数据库中对应的影厅信息记录并返回
HallDataService.addHall	**语法**	public int addHall(Hall hall)
	前置条件	hall不为null且符合输入格式
	后置条件	根据hall往数据库中添加一条影厅信息记录，添加成功则返回1，否则返回0
HallDataService.removeHallById	**语法**	public int removeHallById(int hallId)
	前置条件	无
	后置条件	根据hallId从数据库中将对应的影厅信息记录删除，删除成功则返回1，否则返回0
HallDataService.updateHall	**语法**	public int updateHall(Hall hall)
	前置条件	无
	后置条件	根据hall在数据库中更新对应的影厅信息记录，更新成功则返回1，否则返回0
HallDataService.getHallsExcept	**语法**	public List<Hall> getHallsExcept(List hallIds)

	前置条件	hallIds不为空
	后置条件	在数据库中查找id不在hallIds列表中的对应的影厅记录列表并返回
HallDataService.checkHallName	**语法**	public int checkHallName(String hallName, int hallId)
	前置条件	hallName不为空
	后置条件	在数据库中查找id为hallId、影厅名称为hallName的影厅信息记录条数并返回

表35 userdata模块的接口规范

	提供的服务（供接口）	
UserDataService.createNewAccount	**语法**	int createNewAccount(String username, String password, Integer auth)
	前置条件	输入参数不为空
	后置条件	根据username，password和auth新建一个用户账号
UserDataService.getAccountByName	**语法**	User getAccountByName(String username)
	前置条件	无
	后置条件	根据用户名查找并返回一个账号
UserDataService.updatePassword	**语法**	int updatePassword(UserForm userForm)
	前置条件	无
	后置条件	更新用户密码，并返回更新结果
UserDataService.getCinemaRoles	**语法**	List getCinemaRoles()
	前置条件	无
	后置条件	获取并返回全部影院角色
UserDataService.deleteRoleById	**语法**	int deleteRoleById(Integer id)
	前置条件	无
	后置	根据id删除一个角色，并返回删除结果

	条件	
UserDataService.getAccountByName	**语法**	int updateRoleById(Integer id, String username, String password, Integer auth)
	前置条件	无
	后置条件	更新角色信息并返回更新结果

表36 salesdata模块的接口规范

提供的服务（供接口）		
SalesDataService.insertTicket	**语法**	int insertTicket(Ticket ticket)
	前置条件	无
	后置条件	新增一张电影票
SalesDataService.insertTickets	**语法**	int insertTickets(List tickets)
	前置条件	无
	后置条件	新增多张电影票并返回新增结果
SalesDataService.deleteTicket	**语法**	void deleteTicket(int ticketId)
	前置条件	无
	后置条件	删除一张电影票

SalesDataService.deleteLockedTicket	**语法**	void deleteLockedTicket(int userId, int scheduleId)
	前置条件	无
	后置条件	删除被锁的票
SalesDataService.updateTicketState	**语法**	void updateTicketState(int ticketId, int state)
	前置条件	无
	后置条件	更新电影票状态
SalesDataService.updateTicketActualPay	**语法**	void updateTicketActualPay(int ticketId, double actualPay)
	前置条件	无
	后置条件	更新电影票实际支付金额
SalesDataService.selectTicketsBySchedule	**语法**	List selectTicketsBySchedule(int scheduleId)
	前置条件	无
	后置条件	根据排片查找并返回电影票

SalesDataService.selectTicketByScheduleIdAndSeat	**语法**	Ticket selectTicketByScheduleIdAndSeat(int scheduleId, int columnIndex, int rowIndex)
	前置条件	无
	后置条件	根据排片和座位查找并返回电影票
SalesDataService.selectTicketById	**语法**	Ticket selectTicketById(int id)
	前置条件	无
	后置条件	根据id查找并返回电影票
SalesDataService.selectRefund	**语法**	List selectRefund()
	前置条件	无
	后置条件	查找并返回所有退票策略
SalesDataService.insertRefund	**语法**	void insertRefund(Refund refund)
	前置条件	无
	后置条件	添加一条退票策略

SalesDataService.deleteRefundById	**语法**	void deleteRefundById(int refundId)
	前置条件	无
	后置条件	根据id删除一条退票策略
SalesDataService.selectRefundByMovieId	**语法**	List selectRefundByMovieId(int movieId)
	前置条件	无
	后置条件	根据电影id查找并返回退票策略
SalesDataService.selectRefundById	**语法**	Refund selectRefundById(int refundId)
	前置条件	无
	后置条件	根据id查找并返回退票策略

表37 statisticsdata模块的接口规范

提供的服务（供接口）		
StatisticsDataService.insertOneLike	**语法**	int insertOneLike(int movieId,int userId)
	前置条件	无
	后置	插入一条想看记录

	条件	
StatisticsDataService.deleteOneLike	**语法**	int deleteOneLike(int movieId,int userId)
	前置条件	无
	后置条件	删除一条想看记录
StatisticsDataService.selectLikeNums	**语法**	int selectLikeNums(int movieId)
	前置条件	无
	后置条件	根据movieId查看电影的想看人数
StatisticsDataService.selectLikeMovie	**语法**	int selectLikeMovie(int movieId,int userId)
	前置条件	无
	后置条件	根据movieId和userId查找特定想看记录
StatisticsDataService.getDateLikeNum	**语法**	List<DateLike> getDateLikeNum(int movieId)
	前置条件	无
	后置	根据movieId获取按日期统计的想看人数

	条件	
StatisticsDataService.selectMovieScheduleTimes	**语法**	List<MovieScheduleTime> selectMovieScheduleTimes(Date date, Date nextDate)
	前置条件	无
	后置条件	查询date当天每部电影的排片次数
StatisticsDataService.selectMovieTotalBoxOffice	**语法**	List<MovieTotalBoxOffice> selectMovieTotalBoxOffice()
	前置条件	无
	后置条件	查询所有电影的总票房（包括已经下架的，降序排列）
StatisticsDataService.selectAudiencePrice	**语法**	List<AudiencePrice> selectAudiencePrice(Date date,Date nextDate)
	前置条件	无
	后置条件	查询date当天每个客户的购票金额
StatisticsDataService.selectPlacingRate	**语法**	List<PlacingRateVO> selectPlacingRate(Date date,Date nextDate)
	前置条件	无
	后	查询date当天各电影上座率

	置条件	
StatisticsDataService.selectRecentTotalBoxOffice	**语法**	List<MovieTotalBoxOffice> selectRecentTotalBoxOffice(Date startDate,Date today,int num)
	前置条件	无
	后置条件	查询startDate到today日期内票房最高的num部电影

表38 consumedata模块的接口规范

提供的服务（供接口）		
ConsumeDateService.getTopUpHistoryByUserId	**语法**	List<TopUpHistory> getTopUpHistoryByUserId(Integer userId)
	前置条件	无
	后置条件	根据userId获取此用户的充值记录列表，若无记录则返回空
ConsumeDateService.getConsumeHistoryByUserId	**语法**	List<ConsumeHistory> getConsumeHistoryByUserId(Integer userId)
	前置条件	无
	后置条件	根据userId获取此用户的消费记录列表，若无记录则返回空
ConsumeDateService.getConsumeHistoryById	**语法**	ConsumeHistory getConsumeHistoryById(Integer id)
	前置	无

	条件	
	后置条件	根据id获取消费记录
ConsumeDateService.insertTopUpHistory	**语法**	int insertTopUpHistory(Integer userId, Double money,Double discount,Double balance,Timestamp time)
	前置条件	无
	后置条件	插入新的充值记录
ConsumeDateService.insertConsumeHistory	**语法**	int insertConsumeHistory(Integer userId, Double money,Double discount,String consumeType,Integer type,Integer contentId)
	前置条件	无
	后置条件	插入新的消费记录
ConsumeDateService.selectConsumeQulifiedUsers	**语法**	List<TotalConsumeUser> selectConsumeQulifiedUsers(Double totalConsume)
	前置条件	无
	后置条件	获得消费总额大于totalConsume的用户

表39 activitydata模块的接口规范

提供的服务（供接口）		
ActivityDataService.insertActivity	**语法**	int insertActivity(Activity activity)
	前置条件	activity格式正确
	后置条件	在数据库中插入一个优惠活动
ActivityDataService.insertActivityAndMovie	**语法**	int insertActivityAndMovie(int activityId,List<Integer> movieId)
	前置条件	activityId和movieId格式正确
	后置条件	在数据库中插入一条活动和电影的关系
ActivityDataService.selectActivities	**语法**	List<Activity> selectActivities()
	前置条件	无
	后置条件	返回所有优惠活动信息
ActivityDataService.selectActivitiesByMovie	**语法**	List<Activity> selectActivitiesByMovie(int movieId)
	前置条件	movieId格式正确
	后置条件	返回特定电影的优惠活动信息
ActivityDataService.selectById	**语法**	Activity selectById(int id)
	前置条件	id格式正确
	后置条件	获取指定id的优惠活动
ActivityDataService.deleteActivityById	**语法**	void deleteActivityById(int id)
	前置条件	id格式正确
	后置条件	删除指定的优惠活动
ActivityDataService.deleteActivityAndMovie	**语法**	void deleteActivityAndMovie(int id)
	前置条件	id格式正确
	后置	根据活动id删除活动和电影关系

条件

表40 carddata模块的接口规范

	提供的服务（供接口）	
CardTypeDataService.selectAllCards	**语法**	List<CardType> selectAllCards()
	前置条件	无
	后置条件	获取所有会员卡类型
CardTypeDataService.insertOneCard	**语法**	int insertOneCard(CardType cardType)
	前置条件	cardTypeForm信息格式正确
	后置条件	在数据库中插入会员卡类型
CardTypeDataService.deleteCardById	**语法**	int deleteCardById(int cardId)
	前置条件	cardId格式正确
	后置条件	在数据库中删除会员卡类型，将会员卡类型状态设为0
CardTypeDataService.updateCardById	**语法**	int updateCardById(int cardId, CardType cardType)
	前置条件	cardId和cardTypeForm格式正确
	后置条件	更新会员卡类型信息
VIPCardDataService.insertOneCard	**语法**	int insertOneCard(VIPCard vipCard)
	前置条件	vipCard格式正确
	后置条件	为指定用户增加会员卡
VIPCardDataService.selectCardByUserId	**语法**	VIPCard selectCardByUserId(int userId)
	前置条件	userId格式正确
	后置条件	获取某用户的有效会员卡
VIPCardDataService.updateCardBalance	**语法**	void updateCardBalance(int id, double balance)
	前置条	id和balance格式正确

	件	
	后置条件	为指定会员卡更新余额
VIPCardDataService.selectCardById	**语法**	VIPCard selectCardById(int id)
	前置条件	id正确
	后置条件	获取指定id的会员卡
VIPCardDataService.deleteCardById	**语法**	int deleteCardById(int cardId)
	前置条件	cardId正确
	后置条件	从数据库中删除指定id的会员卡，将状态设置为0

表41 coupondata模块的接口规范

	提供的服务（供接口）	
CouponDataService.selectAllCoupons	**语法**	List<Coupon> selectAllCoupons()
	前置条件	无
	后置条件	返回所有有效优惠券
CouponDataService.insertCouponUser	**语法**	void insertCouponUser(int couponId,int userId)
	前置条件	用户id和优惠券id存在
	后置条件	在数据库中插入优惠券与用户关系
CouponDataService.selectById	**语法**	Coupon selectById(int id);
	前置条件	优惠券id存在
	后置条件	返回特定id的优惠券
CouponDataService.selectAllCoupons	**语法**	List<Coupon> selectCouponByUser(int userId)
	前置条件	userId存在
	后置条件	返回某个用户所拥有的有效优惠券
CouponDataService.insertCoupon	**语法**	int insertCoupon(Coupon coupon)
	前置条件	coupon格式正确
	后置条件	在数据库中插入优惠券

5.6 信息视角

5.6.1 数据持久化对象

系统的PO类就是对应的相关实体类，其定义见下，getter/setter及一些辅助方法略去。

- Hall类属性：影厅id、名称，座位排布，规模

```
public class Hall {
    private Integer id;
    private String name;
    private String seats;
    private Integer scale;
}
```

- User类属性：身份，账户id，用户名，用户密码

```
public class User {
    private Integer id;
    private String username;
    private String password;
    private Integer auth;
}
```

- Ticket类属性：电影票id，用户id，排片id，列号，排号，订单状态，实付款，付款时间

```
public class Ticket {
    private int id;
    private int userId;
    private int scheduleId;
    private int columnIndex;
    private int rowIndex;
    private int state;
    private double actualPay;
    private Timestamp time;
}
```

- Refund类属性：id，电影id，适用时间，折算策略

```
public class Refund {
    private int id;
    private int movieId;
    private int time;
    private int price;
}
```

- Coupon类属性：优惠券id，优惠券描述，优惠券名称，使用门槛，优惠金额，可用时间、失效时间

```
public class Coupon {
    private int id;
    private String description;
    private String name;
    private double targetAmount;
    private double discountAmount;
    private Timestamp startTime;
    private Timestamp endTime;
}
```

- CardType类属性：会员卡类型id，会员卡名称，会员卡描述，购买价格，充值需满金额，充值优惠金额，购票需满金额，购票优惠金额

```
public class CardType {
    private int id;
    private String name;
    private String description;
    private double price;
    private double topUpTarget;
    private double topUpDiscount;
    private double ticketTarget;
    private double ticketDiscount;
}
```

- VIPCard类属性：会员卡id，用户id，会员卡类型id，会员卡余额，办卡日期，会员卡类型

```
public class VIPCard {
    private int userId;
    private int id;
    private int cardTypeId;
    private double balance;
    private Timestamp joinDate;
    private CardType cardType;
}
```

- Activity类属性：活动名称，活动描述，活动开始时间，活动截止时间，优惠电影列表，优惠券规格，活动需满金额

```
public class Activity {
    private int id;
    private String name;
    private String description;
    private Timestamp startTime;
    private Timestamp endTime;
    private List<Movie> movieList;
    private Coupon coupon;
```

```
    private double targetAmount;
}
```

- TopupHistory类属性：id，账户id，充值金额，优惠金额，充值后余额，充值时间

```
public class TopUpHistory {
    private Integer id;
    private Integer userId;
    private Double money;
    private Double discount;
    private Double balance;
    private Timestamp time;
}
```

- ConsumeHistory类属性：id，账户id，消费金额，优惠金额，消费方式，消费类型，消费内容id

```
public class ConsumeHistory {
    private Integer id;
    private Integer userId;
    private Double money;
    private Double discount;
    private String consumeType;
    private Integer type;
    private Integer contentId;
}
```

5.6.2 数据库表

第三阶段主要使用的数据库表有：hall表、coupon表、coupon_user、card_type表、vip_card表、activity表、activity_movie表、topup_history表、consume_history表