中山大學
SUN YAT-SEN UNIVERSITY

互联网订票系统

分析与设计

[V1.0(版本号)]

项目组成员：张玄镐 15331403
吴志滨 15331318
艾力亚尔 15331002
任洪坤 15331264
张睿 15331399

指导老师： 衣杨

目录

一、需求分析

1.1 问题描述

1.1.1 项目目的

本项目为采用基于 UML 的面相对象分析与设计的方法，以 B/S 架构建立一个互联网电影购票系统，建立该系统的主要目的在于：

1. 让用户能够更容易找到自己喜欢的电影
2. 让用户得到最新的电影动态并以最方便的方式买到自己喜欢的电影的票

1.1.2 项目背景

随着电影行业经济的快速增长，各大在线购票电商平台也百家争鸣，互相竞争电影市场。电影购票代理网站的核心是购票， 但是当下电影产业发展使得购票绝不会成为一个与电影相关的网站的主要竞争力。因而关于构建新一代电影购票网站的理念也就应运而生

1.1.3 系统角色划分

系统将用户划分为用户与系统管理员。

1.1.4 系统业务功能说明

本系统主要由以下 6 个板块构成，分别为：

1.在线选座，购票
2.上映电影简介，评分，评价
3.按电影院位置，名称查找电影院
4.按电影名称，类型，演员筛选电影
5.根据特定场景进行影片推荐
6.观影后进行类似影片推荐

1.2 用例图

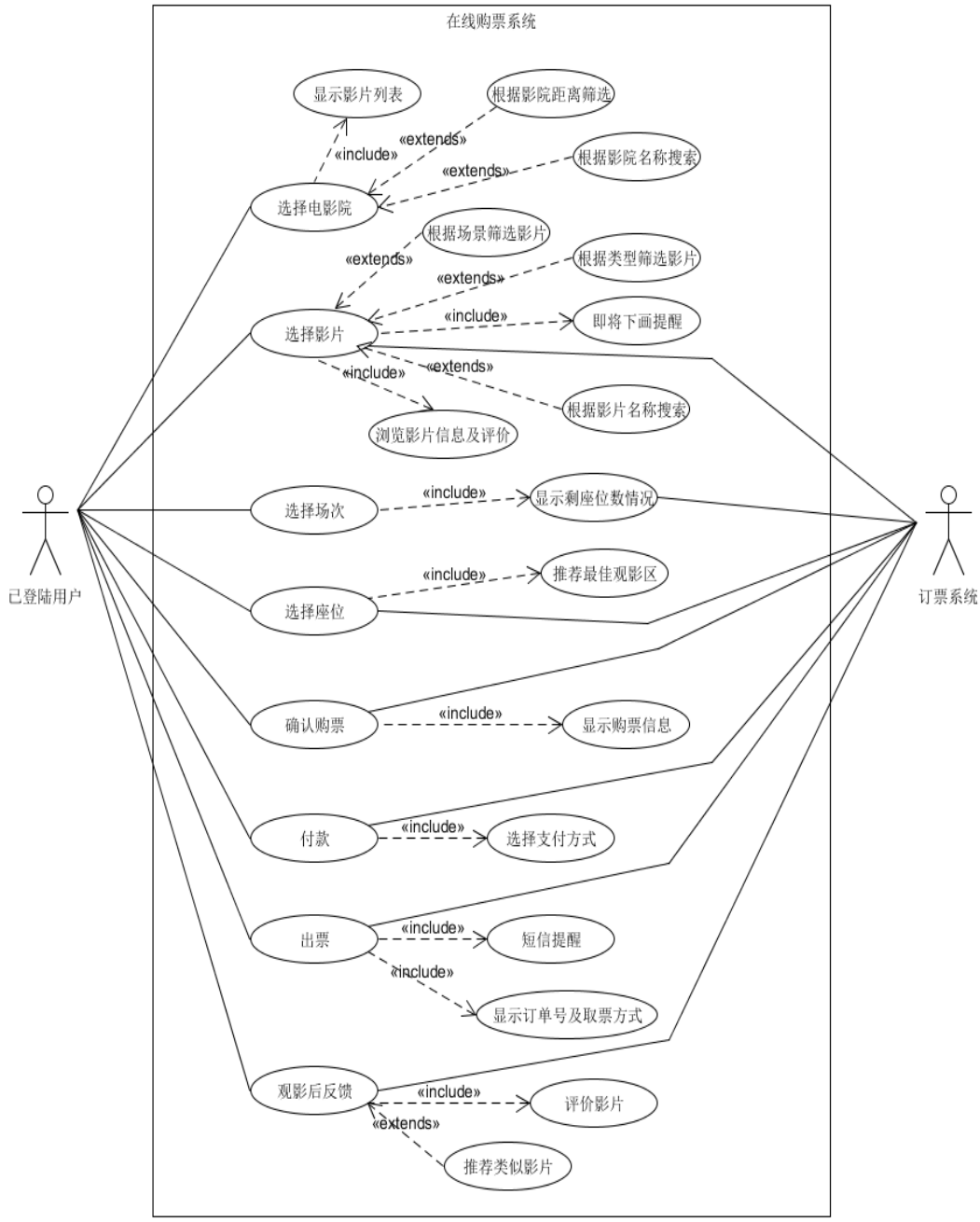

图 1-1 用例图

1.3 用例规约

1.3.1 选择影院

1. 简要描述

选择影院即是用户在登陆后通过距离，或影院名称获取影院信息的过程。

2. 参与者

用户

3. 场景描述

用户在后台管理系统登录之后可以进行影院的选择，在没有信息的时候，系统调用百度地图 API 按照距离列出影院列表，在搜索框内有输入时，在影院信息库中搜索对应的影院。

4. 前置条件

用户已经在后台管理系统成功登录。

5. 后置条件

用户在进行查询信息的操作时，系统均会有确认反馈。

1.3.2 选择影片

（1）简要描述

已登录用户通过系统推荐、搜索、或者影院场次信息选择影片。

（2）参与者

注册用户

（3）场景描述

用户成功注册后即可以选择影片，注册用户在系统页面上点击对应的影片图片后，系统就会向用户传回影片信息。

（4）前置条件

注册用户已经成功登录。

（5）后置条件

用户会收到影片信息。

1.3.3 选择场次

（1）简要描述

已登录用户通过系统推荐、搜索、或者影院场次信息选择影片后选取对应的场次。

（2）参与者

注册用户

（3）场景描述

用户在成功选择影片与影院后，系统发回场次信息，用户点选场次，完成场次选择。

（4）前置条件

注册用户已经成功登录并完成选择影片与影院操作。

（5）后置条件

用户会收到场次信息。

1.3.4 选择座位

（1）简要描述

已登录用户选取场次后，选取当前场次的座位。

（2）参与者

注册用户

（3）场景描述

用户在成功选择场次后，系统发回座位信息，用户点选座位，完成座位选择。

（4）前置条件

注册用户已经成功登录并完成选择场次操作。

（5）后置条件

用户会收到座位确认信息。

1.3.5 付款

（1）简要描述

已登录用户确认座位后，付款的操作。

（2）参与者

注册用户

（3）场景描述

用户在成功选择座位后，系统发回订单信息，系统调用支付系统的接口，完成付款操作。

（4）前置条件

注册用户已经成功登录并完成选择座位操作。

（5）后置条件

用户会收到付款确认信息。

1.4 术语表

名词术语	定义	英文名
数据库	数据库指的是以一定方式储存在一起、能为多个用户共享、具有尽可能小的冗余度的特点、是与应用程序彼此独立的数据集合。在本文中特指数据库管理系统所管理的数据库。	Database
用户名	包括超级管理员、管理员、实验室用户以及注册用户登录时候填写的名称	User
密码	包括超级管理员、管理员、实验室用户以及注册用户登录时候填写的口令	Password
匿名用户	仅能查看项目公有信息等用户	Anonymous User
注册用户	通过注册的用户，允许申请加入到项目中	Registered user
观影后反馈	已登录用户看完电影以后评价影片也能通过订票系统得到类似影片推荐	Comment
选择座位	已登录用户或匿名用户选择最佳观影区	Seat
选择影片	已登录用户或匿名用户选择最佳影片，浏览影片信息	Film
选择影院	已登录用户或匿名用户根据影院距离，要看的影片信息选择影院	Theater
出票	已登录用户确定影片信息以后确认出票，得到短信通知	Ticket

表 1-2 术语表

二、架构设计

2.1 架构描述

整个网站的架构分为 MVC 三层，分别为模型层、视图层和控制器层。MVC 架构设计将业务逻辑、数据和界面显示分离开来，使得开发人员在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。这样的架构不仅使整个网站的开发清晰明了，开发模块化，而且大大提高了开发的效率。

2.1.1 模型层

本网站的数据模型分为两部分。

1. 第一部分为网站用户数据模型，其中包括了管理员和用户的数据。

2. 第二部分为网站项目的主要数据，包括了影院信息，影片信息，场次信息和票务信息。

2.1.2 视图层

本网站的的视图层由以下四个板块构成：

1.首页

2.影院

3.电影

4. 我的

2.1.3 控制器层

本网站的控制器层主要包括三个部分。

1. 第一部分为用户注册登录的控制逻辑。用户注册和登录需要填写的信息均包括用户名、密码和验证码。控制器从视图层的表单中获取这些数据，并作相应的验证处理，最终将这些数据存储到数据库中或者改变某些数据状态的值。

2. 第二部分为用户根据影院选场次、根据电影选影院和场次和购买电影票的控制逻辑。用户在相应的视图层页面中选择电影，影院，选座，付款的相关信息，控制器从视图层的表单中获取这些数据，然后将这些数据连同发布用户的信息存储在数据库中，最终在相应的视图展示这些信息。

3. 第三部分为点击链接页面跳转的控制逻辑。用户点击网站的链接，经由控制逻辑请求响应对应的 URL 后，网站会跳转到相应的页面。

2.2 架构图

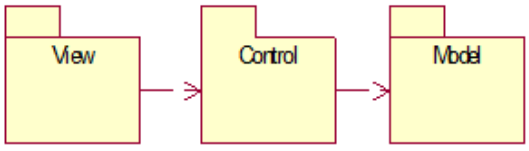

图 2-1 架构图

2.3 关键抽象

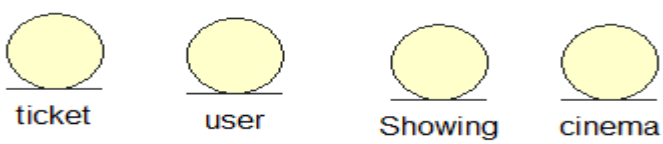

图 2-2 关键类抽象

三、用例分析

3.1 选择影院用例类的析取

所有用户都能够执行选择影院以及其后的所有用例。管理员用户能够通过管理页面（cinema_service）对影院信息进行管理：删除、添加、更新。

最后 service 类注入信息控制数据，最后通过 DAO 与 DB 交互修改数据。

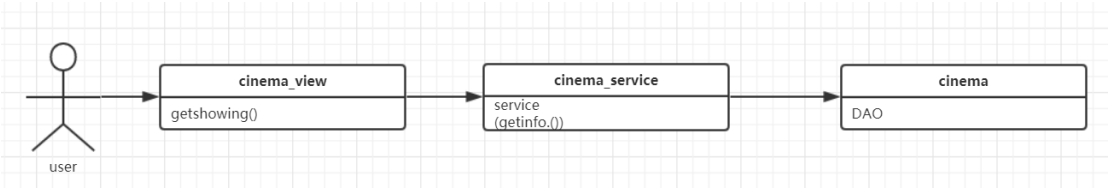

图 3-1 选择影院用例类的析取

3.2 选择影片用例类的析取

只有 registerUser 能够进行选择影片操作，其通过不定向异步提交表单的方式确定 movie。

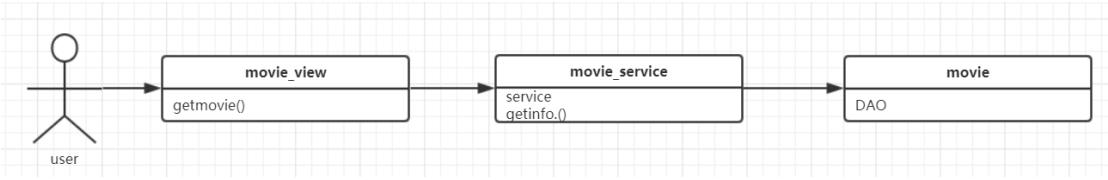

图 3-2 选择影片用例类的析取

3.3 选择场次用例类的析取

通过 cinema 与 movie 提交的表单，定向到某场 show，通过 showing service，调取 dao 类与 db 交互。

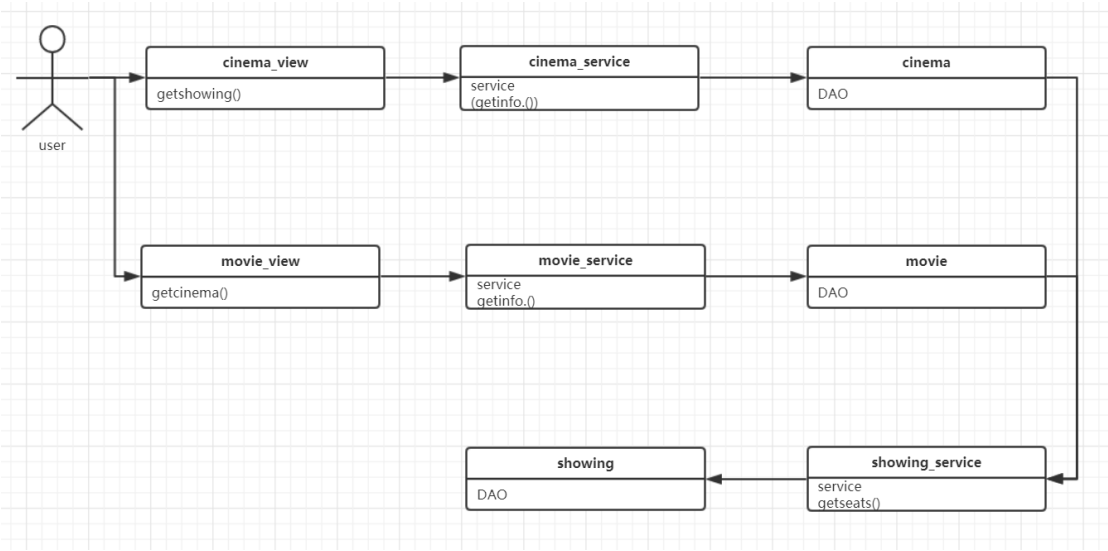

图 3-3 选择场次用例类的析取

3.4 确认购票用例类的析取

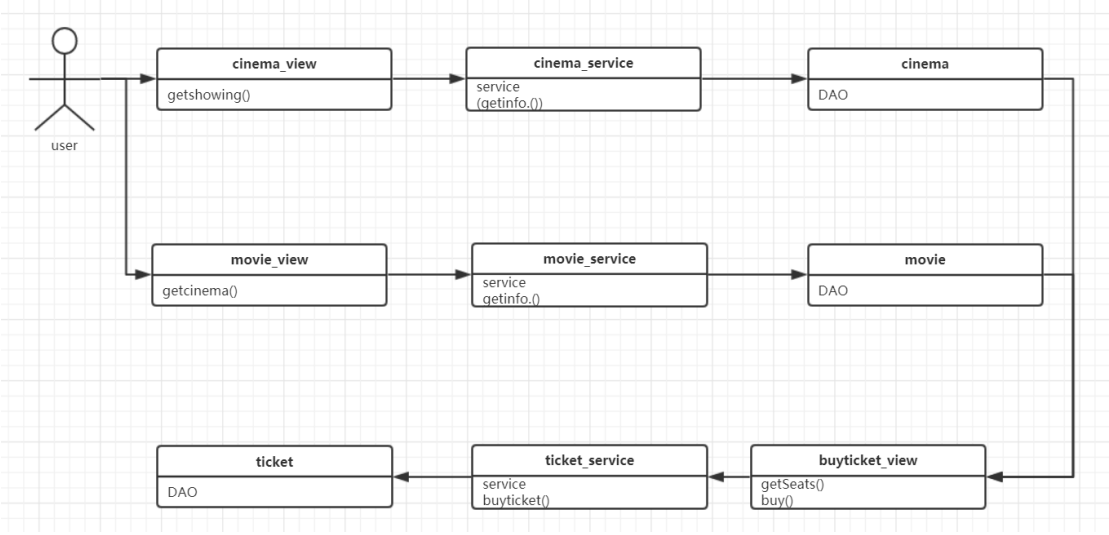

图 3-4 确认购票用例类的析取

3.5 付款、出票用例类的析取

通过系统间消息传递实现，属于系统边界。由 ticket 类提供服务。

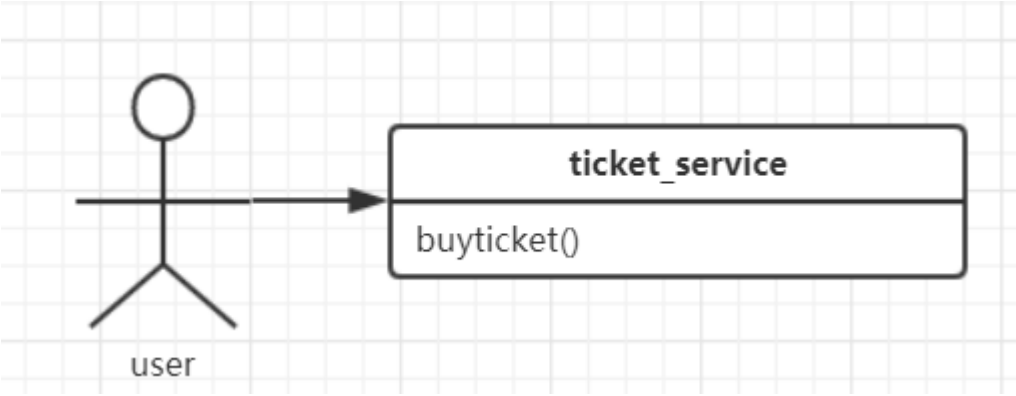

图 3-5 付款，出票用例类的析取

3.6 观影反馈用例类的析取

由 movie_info_service 类与 movie_info 类提供服务。User 完成付款与观影后，通过 movie_view 提供的入口调用 movie_info_service，进行评论，再交由 movie_info 完成数据持久化。

图 3-6 观影后反馈用例类的析取

3.7 总体类的析取

所有用户均为 User 的子类，不同用户具有不同的权限。注册用户能够在系统中得到影院、电影、场次、座位等信息；而对于影院、电影、场次、座位的基本信息（如影院名称，电影场次等）的修改只有管理员拥有该权限。

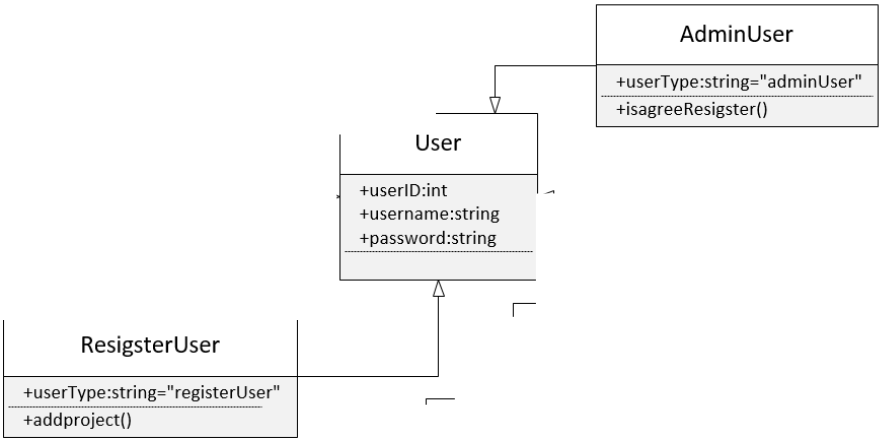

图 3-7 选择影院用例类的析取

3.8 分析机制

安全：将用户的相关信息进行加密保存到数据库中。

持久化：将系统的数据保存到数据库中，实现数据的持久化。

接口：系统的设计为了能够保障系统的鲁棒性，需要实现高内聚，低耦合的设计原则，通过接口，将不同板块、子系统的逻辑进行解耦。对于一个子系统的访问需要通过访问该系统的外部类（接口的实现）。

四、类与子系统设计

4.1 确定类的设计

以下是系统的总体类设计图：

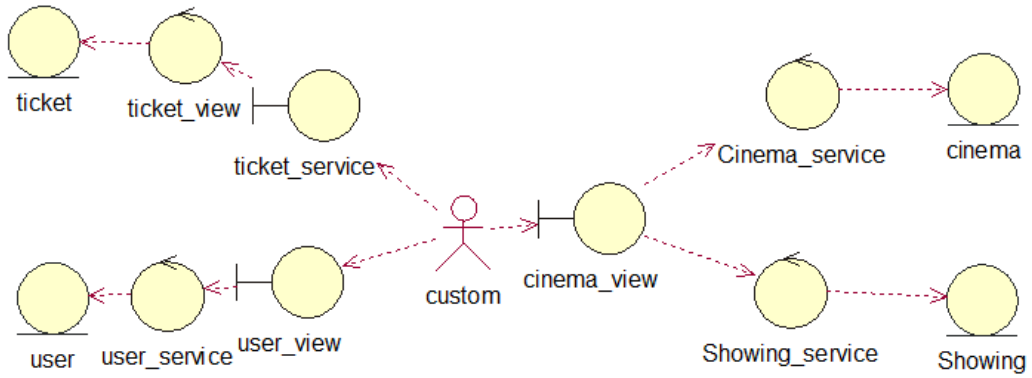

图 4-1 总体类设计

4.2 子系统划分

本研究平台网站包括三个子系统：购票选择子系统、支订单付子系统、观影反馈子系统。三个子系统向主程序提供接口服务，主程序调用子系统提供的接口完成主要的流程。

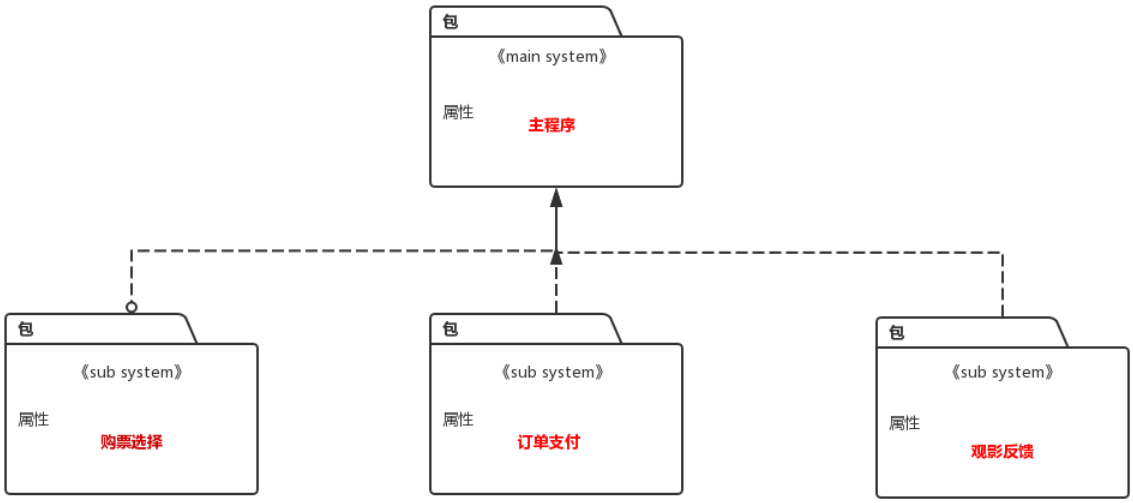

图 4-2 购票平台系统设计图

4.3.1 购票选择系统：包括选择电影院，选择电影，选择场次，选择座位的功能。

根据系统总架构设计图，购票选择子系统也采用三层架构模式：表示层，业务逻辑层，数据库访问层。

表示层以窗口界面的形式与消费者进行交互，展示电影院，电影名称，场次与座位的相关信息，同时负责对信息进行逻辑处理，按照项目来对电影信息进行管理，管理操作包括显示附近电影院，上映电影名称，可选场次与剩余座位。

在交互过程中，表示层获取相关信息后，以消息的形式传到业务逻辑层的控制器，控制器根据消息的类型，转发到相应的业务逻辑层处理类进行处理。处理完毕后，业务逻辑层处理类根据需要将处理后的消息进一步转发到数据访问层的相应类进行处理。最后，通过数据访问层对信息的持久化处理。

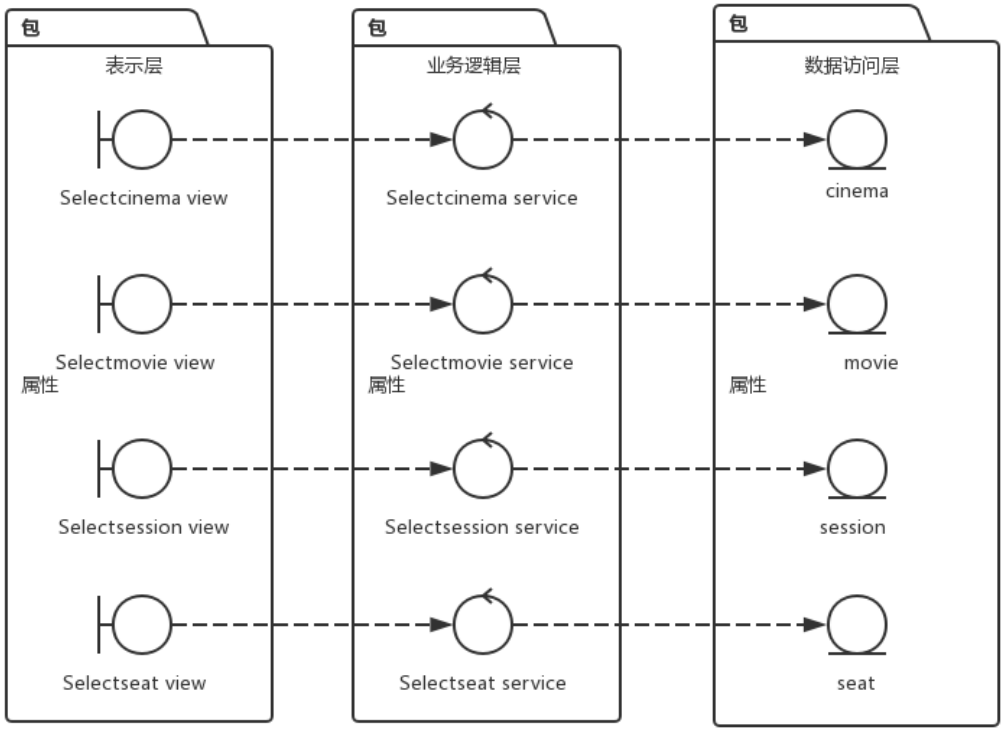

图 4-3 选择与购票子系统设计

4.3.2 支付子系统：包括了订单确定，付款，出票订单号的功能。

支付子系统采用三层架构模式：表示层、业务逻辑层、数据访问层。

表示层以窗口界面展示网站信息。消费者可以浏览购票平台网站上面的信息。在交互过程中，表示层获取获取网站信息的相关信息后，以消息的形式传到业务逻辑层的控制器，控制器根据消息的类型，转发到相应的业务逻辑层处

理类进行处理。处理完毕后，业务逻辑层处理类根据需要将处理后的消息进一步转发到数据访问层的相应类进行处理。最后，通过数据访问层对信息的更新。

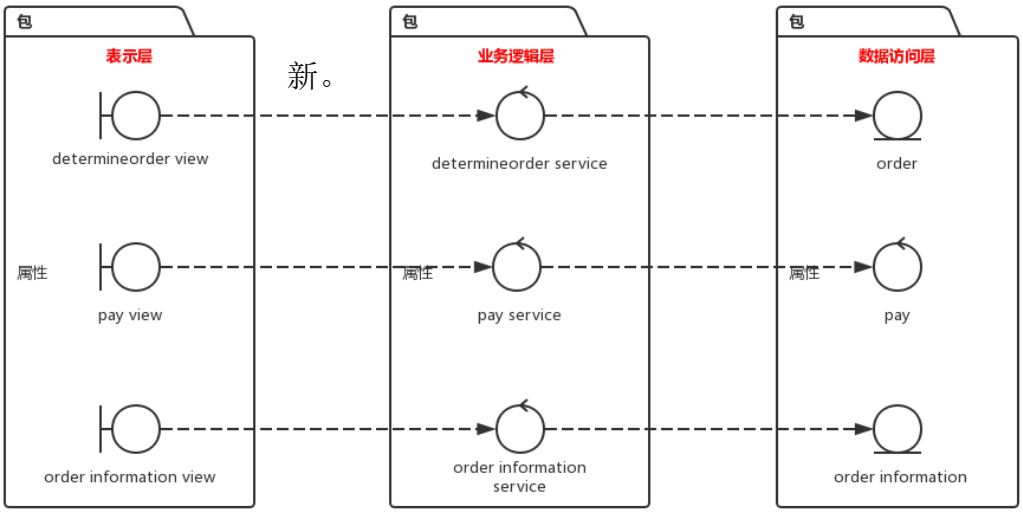

图 4-4 支付子系统设计

4.3.3 观影反馈子系统：包括评价影院，评价电影的功能

观影反馈子系统遵循总体架构的分层策略，分别分为表示层、业务逻辑层与数据访问层。

表示层以窗口的方式与用户进行交互，展示用户需要输入的反馈，包括对电影院环境与服务的反馈，对电影的反馈等等。

业务逻辑层负责提供完整的数据流控制服务，对每一个数据实体提供一个管理器类，管理器类负责处理数据的更改和转移，是整个系统的中枢。管理器向上层提供调用接口，并调用下层的数据访问服务来完成持久化的操作。

数据访问层封装了数据库的连接、读取、写入等操作，以提供数据库访问的服务。

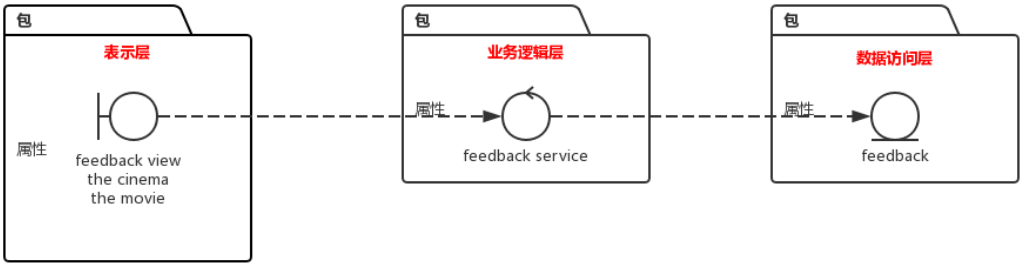

图 4-5 观影反馈子系统设计

五、运行时架构设计

5.1 分析本系统的并发需求

本系统作为一个网页系统，供给不同的用户登录使用，因此该系统需要满足多用户登陆、并发请求加载网页的内容等。

假设一个用户相当于一个请求，该系统的并发需求很大程度由在线的用户数量所决定，由于该系统仅针对重要热带病传播入侵媒介及病原体生物学特性的研究平台，参与的用户数量级相对较小，因此该系统的并发需求相对较小。总共的用户的数量大概控制在 100 人以下，而并发量会比 100 小，因此该系统将并发需求控制在：一个时刻最多同时只有 75 个人同时进行请求。

5.2 识别出相应的进程和线程

由于该系统需要考虑并发需求，因此该系统包含以下进程和线程：

5.2.1 进程

主进程：整一个系统仅由一个进程构成，系统的入口则为该进程的入口，该进程生命周期与系统的生命周期一致，当系统结束运行时候，进程将结束；如果进程被迫结束（例如异常），则该系统也会结束。而主进程中包括多个线程。

5.2.2 线程

1. 请求线程（RequestHandler）

   由于可能存在多个用户请求，因此需要存在一个请求线程，该线程仅用来接收所有用户的请求，并将请求放到对应的请求队列中去；请求线程不对用户请求做具体的响应（请求线程相当于一个生产者）。

2. 响应线程（ResponseHandler）

   请求线程将用户请求存放到一个请求队列中去，需要一个响应线程对队列进行轮询操作，如果发现队列存在用户的请求，则从资源池请求分配资源进行响应（响应线程相当于一个消费者）。

3. 处理线程（HandlerThread）

   处理线程由资源池进行分配，当响应线程请求一个线程处理请求的时候，如果资源池存在空闲的线程，则将该空闲线程用于处理该请求。

5.3 描述相应的进程和线程

下图表示了该系统的请求处理过程，该过程主要包含 3 个部分：请求过程、响应过程、处理过程。

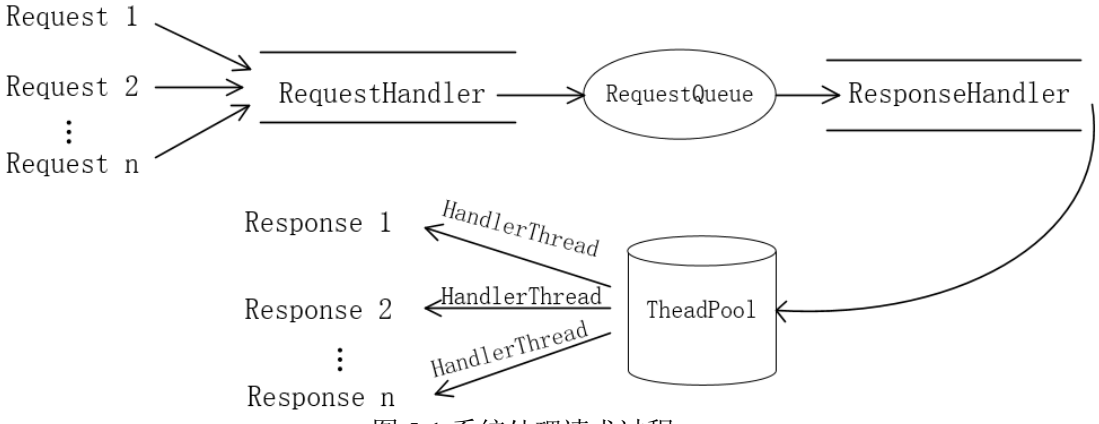

图 5-1 系统处理请求过程

上图中，RequestHandler 为请求线程，该线程将所有的请求 Request 放到请求队列 RequestQueue 中；ResponseHandler 为响应队列，将 RequestQueue 中的请求取出进行相应。而 RequestQueue 队列的请求处理方式对应于生产者和消费者模式：

RequestHandler 充当生产者，将请求放入到请求队列中，如果请求队列 RequestQueue 已满，则会阻塞 RequestHandler，直至 RequestQueue 有空余的位置；

ResponseHandler 充当消费者，从 RequestQueue 中取出请求，如果 RequestQueue 中为空，则会阻塞 ResponseHandler。如果 ResponseHandler 取出了请求，则从线程池中取出处理线程 HandlerThread 进行请求的处理。

线程池 ThreadPool 是一个固定大小的线程池，不能无限地从中取出 HandlerThread 进行请求的处理，因为当 ThreadPool 的线程太多时，容易导致空闲资源的浪费、以及线程切换所带来的 CPU 资源损耗。

HandlerThread 线程用于处理每一个用户的请求，每个用户都与一个 HandlerThread 线程进行对应，HandlerThread 处理后将用户请求的结果返回给用户。

组员贡献比例表

姓名	学号	贡献
张玄镐	15331403	24%
艾力亚尔	15331002	24%
吴志滨	15331318	24%
任洪坤	15331264	18%
张睿	15331399	10%