文档编号：mov-0001

影院售票系统

****课程报告

**2018年12月**

目录

[1需求分析 6](#_Toc532921998)

[1.1引言 6](#_Toc532921999)

[1.1.1编写目的 6](#_Toc532922000)

[1.1.2预期读者 7](#_Toc532922001)

[1.1.3参考文献 7](#_Toc532922002)

[1.2系统概述 8](#_Toc532922003)

[1.2.1功能需求概述 8](#_Toc532922004)

[1.2.2功能描述表 9](#_Toc532922005)

[1.3功能需求 10](#_Toc532922006)

[1.3.1功能划分 10](#_Toc532922007)

[1.3.2功能描述 11](#_Toc532922008)

[2总体设计 25](#_Toc532922009)

[2.1引言 25](#_Toc532922010)

[2.1.1编写目的 25](#_Toc532922011)

[2.1.2背景 25](#_Toc532922012)

[2.1.3定义 26](#_Toc532922013)

[2.2系统功能结构设计 26](#_Toc532922014)

[2.2.1模块的划分和描述 26](#_Toc532922015)

[2.2.2系统的功能结构图 27](#_Toc532922016)

[2.3系统的体系结构 28](#_Toc532922017)

[3详细设计 29](#_Toc532922018)

[3.1会员登录模块 29](#_Toc532922019)

[3.2会员注册模块 29](#_Toc532922020)

[3.3管理员登录模块 30](#_Toc532922021)

[3.4添加管理员模块 30](#_Toc532922022)

[3.5会员个人中心模块 31](#_Toc532922023)

[3.6添加会员模块 31](#_Toc532922024)

[3.7取票功能 31](#_Toc532922025)

[3.8会员信息更新模块 32](#_Toc532922026)

[3.9会员查询模块 32](#_Toc532922027)

[3.10电影添加模块 33](#_Toc532922028)

[3.11电影查询模块 33](#_Toc532922029)

[3.12电影删除模块 34](#_Toc532922030)

[3.13电影更新模块 34](#_Toc532922031)

[3.14场次增加模块 35](#_Toc532922032)

[3.15场次删除模块 35](#_Toc532922033)

[3.16场次更新模块 36](#_Toc532922034)

[3.17场次查询 36](#_Toc532922035)

[3.18购票功能 37](#_Toc532922036)

[3.19选座功能 37](#_Toc532922037)

[3.20支付功能 38](#_Toc532922038)

[3.21订单查看 38](#_Toc532922039)

[4接口设计 39](#_Toc532922040)

[4.1界面接口设计 39](#_Toc532922041)

[4.1.1用户界面接口设计 39](#_Toc532922042)

[4.1.2管理界面 43](#_Toc532922043)

[4.2数据库接口设计 46](#_Toc532922044)

[4.1.3.1会员信息接口 46](#_Toc532922045)

[4.1.3.2电影场次信息接口 47](#_Toc532922046)

[4.1.3.3电影信息接口 47](#_Toc532922047)

[4.1.3.4管理员信息接口 47](#_Toc532922048)

[4.1.3.5订单信息接口 47](#_Toc532922049)

[4.1.3.6状态码 47](#_Toc532922050)

[4.3外部接口 48](#_Toc532922051)

[4.3.1用户的输入和反馈的结果 48](#_Toc532922052)

[4.3.2 和其他系统之间的接口。 48](#_Toc532922053)

[4.4内部接口 49](#_Toc532922054)

[4.4.1模块的接口：模块的输入和输出。 49](#_Toc532922055)

[5数据库设计 52](#_Toc532922056)

[5.1数据库逻辑结构设计 52](#_Toc532922057)

[5.2数据库物理结构设计 53](#_Toc532922058)

[6个人工作和总结 56](#_Toc532922059)

# 1需求分析

## 1.1引言

### 1.1.1编写目的

本部分用于描述“影院售票系统”项目的系统需求，为该项目的总体设计、详细设计、接口设计和数据库设计提供设计依据。

电子商务发展已经在世界范围内取得了巨大的成就，传统商业模式正在被现代电子商务逐渐取代，影院订票管理系统的出现对于影院的管理工作来说是至关重要的，并且可以通过对系统的有效利用，建立起影院和观众之间的新型管理，从而提高影院的市场竞争力。影院订票管理系统应该能够为用户提供充足的信息、快捷的查询手段、可靠的票务办理业务和轻松的查阅自己的消费信息等，在正常的网络环境下啊，一般网上购票都能在几分钟时间内选择需要的影片、时间、座位等诸多过程。这种快速高效的订票方式最大的作用是能够帮助影院分散巨大的人流，提高影院的工作效率，也节省用户的时间、方便用户更好的进行订票。

### 1.1.2预期读者

本文档面向以下几种对象：

（1）项目经理：项目经理可以根据该文档了解产品的预期功能，并据此进行系统设计、项目管理。

（2）设计员：对需求进行分析，并设计出系统，包括数据库的设计。

（3）程序员：配合《设计报告》，了解系统功能，编写《用户手册》。

（4）测试员：根据本文档编写测试用例，并对软件产品进行功能性测试和非功能性测试。

（5）用户：了解产品的预期功能和性能，并与分析人员一起对整个需求进行讨论和协商。

### 1.1.3参考文献

《软件工程—实践者的研究方法》——（美）Roger S.Pressman著 机械工业出版社

《数据库系统概论（第5版）》——王珊，萨师煊 编著 高等教育出版社

## 1.2系统概述

### 1.2.1功能需求概述

该影院售票系统的功能需求，有3个方面：

**1. 对于客户**

客户能够注册、登陆该系统，能够在该系统上选择电影、场次、座位，并购票、支付。客户还能够通过该系统，进行取票。客户能够在该系统上查看、修改自己的个人信息。

**2. 对于售票员**

售票员经过登陆验证后，登陆系统，可在系统中查询正在上映的电影。查询各种订单信息，并对其进行管理。售票员还能售票给来到现场买票的客户，并让用户通过支付宝、微信等支付方式进行支付。售票员还能帮客户办理会员，能根据客户的描述替客户填写客户信息。

**3. 对于管理员**

管理员通过系统验证、登陆后，能够管理各种电影信息。比如删除、添加、修改、查询正在上映的电影、场次。增加、修改、查询会员信息。查询、删除订单。

### 1.2.2功能描述表

|  |  |
| --- | --- |
| 序号 | 功能描述 |
| 1 | 登陆 |
| 2 | 注册 |
| 3 | 购票 |
| 4 | 选座 |
| 5 | 支付 |
| 6 | 取票 |
| 7 | 查询电影 |
| 8 | 查看场次 |
| 9 | 查看订单 |
| 10 | 查看个人信息 |
| 11 | 修改个人信息 |
| 12 | 管理电影 |
| 13 | 管理场次 |
| 14 | 删除订单 |

## 1.3功能需求

### 1.3.1功能描述

#### 1.3.1.1登陆

**用例名：**登陆

**用例编号：**1

**参与角色：**售票员或客户或管理员

**前置条件：**角色点击登陆按钮

**后置条件：**无

**输入数据：**账号、密码

**输出数据：**登陆结果

**事件流：**

1. 角色输入账号、密码
2. 系统接收账号、密码，将其拿到数据库中检索
3. 数据库中检索到账号，且对应的密码一致
4. 系统提示“登陆成功”，显示登陆成功后的界面

**异常：**

3.a. 密码不一致。系统提示“账号或密码错误，请重新输入”，返回1

3.b. 没有找到账号。系统提示“账号或密码错误，请重新输入”，返回1

#### 1.3.1.2注册

**用例名：**注册

**用例编号：**2

**参与角色：**客户

**前置条件：**角色点击注册按钮

**后置条件：**无

**输入数据：**手机号（充当账号）、昵称、密码、再次输入的密码

**输出数据：**注册结果

**事件流：**

1. 角色输入手机号、昵称、密码、再次输入的密码
2. 系统接收手机号、昵称、密码、再次输入的密码，将手机号拿到数据库中检索
3. 数据库中没有检索到手机号
4. 系统比对两次输入的密码是否一致
5. 系统将手机号、昵称、密码存储到数据库中，向角色提示“注册成功”

**异常：**

3.a. 数据库中存在该手机号。系统提示“该手机号已存在，请登陆或注册其他手机号”，返回1

4.a. 两次输入的密码不一致。系统提示“密码不一致，请重新输入”，返回1

#### 1.3.1.3购票

**用例名：**购票

**用例编号：**3

**参与角色：**售票员或客户

**前置条件：**角色查看场次、角色已经登录

**后置条件：**角色选座

**输入数据：**无

**输出数据：**选座界面或提示无票

**事件流：**

1. 角色点击购票按钮
2. 系统收到到购票请求，系统从数据库中，获取该场次的所有信息
3. 系统检索该场次的余票信息
4. 该场次有余票
5. 系统根据场次的座位信息，显示该场次的选座界面

**异常：**

4.a. 该场次没有余票了。系统提示“该场次没有余票”，退出

#### 1.3.1.4选座

**用例名：**选座

**用例编号：**4

**参与角色：**售票员或客户

**前置条件：**角色购票、角色已经登录

**后置条件：**角色支付

**输入数据：**座位号

**输出数据：**选座结果

**事件流：**

1. 角色选定一个座位
2. 系统接收到选定的座位，将其拿到数据库中检索
3. 数据库中检索到该场次的该座位为被购买
4. 系统提示“选座成功”，并根据该场次信息，显示支付界面

**异常：**

3.a. 座位已经被购买。系统提示“该座位已经被购买，请重新选座”，返回1

#### 1.3.1.5支付

**用例名：**支付

**用例编号：**5

**参与角色：**客户

**前置条件：**角色选座、角色已经登录

**后置条件：**无

**输入数据：**金钱

**输出数据：**支付结果，取票码

**事件流：**

1. 角色在支付界面中选择一个支付方式进行支付
2. 系统支付请求后，核对信息，确认金额到账
3. 系统将该订单、订座等相关记录写入数据库，
4. 系统提示“支付成功”，显示取票码

**异常：**

2.a. 角色支付金额不足。系统提示“余额不足请选择其他方式支付”，返回1

#### 1.3.1.6取票

**用例名：**取票

**用例编号：**6

**参与角色：**售票员或客户

**前置条件：**角色已经登录

**后置条件：**无

**输入数据：**取票码

**输出数据：**取票结果

**事件流：**

1. 角色输入取票码
2. 系统接收取票码，将其拿到数据库中检索
3. 数据库中检索到取票码，返回取票码对应的订票信息
4. 系统提示“取票成功”，将票的信息打印出来

**异常：**

3.a. 没有检索到取票码。系统提示“取票码错误，请重新输入”，返回1

#### 1.3.1.7查询电影

**用例名：**查询电影

**用例编号：**7

**参与角色：**售票员或客户或管理员

**前置条件：**角色已经登录

**后置条件：**无

**输入数据：**无

**输出数据：**查询电影结果

**事件流：**

1. 角色点击查询电影按钮
2. 系统接收查询请求，系统向数据库发起请求，获取正在上映的电影信息
3. 数据库返回电影信息
4. 系统显示正在上映的许多电影的信息

#### 1.3.1.8查看场次

**用例名：**查看场次

**用例编号：**8

**参与角色：**售票员或客户或管理员

**前置条件：**角色查询电影、角色已经登录

**后置条件：**无

**输入数据：**要查看场次的电影

**输出数据：**登陆结果

**事件流：**

1. 角色选择一部电影，点击查看场次
2. 系统接收请求，向数据库发起请求，获取该电影的场次信息
3. 数据库中检索该电影的场次信息，并返回给系统
4. 系统接收到返回的信息，并显示出来

**异常：**

3.a. 电影尚未安排场次。系统提示“该电影尚未安排场次”，返回1

#### 1.3.1.9查看订单

**用例名：**查看订单

**用例编号：**9

**参与角色：**售票员或客户或管理员

**前置条件：**角色已经登录

**后置条件：**无

**输入数据：**无

**输出数据：**订单情况

**事件流：**

1. 角色点击查看订单
2. 系统接收请求，根据角色权限，向数据库发起不同的请求，申请不同的订单信息
3. 数据库根据系统的请求，返回相应的订单信息
4. 系统接收到订单信息，并显示出来

**异常：**

3.a. 没有订单。系统提示“没有订单”，退出

#### 1.3.1.10查看个人信息

**用例名：**查看个人信息

**用例编号：**10

**参与角色：**售票员或客户

**前置条件：**角色已经登录

**后置条件：**无

**输入数据：**无

**输出数据：**个人信息

**事件流：**

1. 角色选择查看个人信息
2. 系统接收请求，根据角色权限，向数据库发起不同的请求，申请不同的个人信息
3. 数据库根据系统的请求，返回相应的个人信息
4. 系统接收到个人信息，并显示出来

#### 1.3.1.11修改个人信息

**用例名：**修改个人信息

**用例编号：**11

**参与角色：**售票员或客户

**前置条件：**角色已经登录

**后置条件：**无

**输入数据：**个人信息

**输出数据：**修改结果

**事件流：**

1. 角色输入需要修改的个人信息
2. 系统接收需要修改的个人信息，将其写入数据库中
3. 系统提示“修改成功”

**异常：**

2.a. 输入的个人信息有误（如不符合要求或逻辑上有问题）。系统提示“信息有误，请重新输入”，返回1

#### 1.3.1.12管理电影

**用例名：**管理电影

**用例编号：**12

**参与角色：**管理员

**前置条件：**角色已经登录

**后置条件：**无

**输入数据：**要对电影信息进行的操作，以及操作可能存在的附加信息

**输出数据：**操作结果

**事件流：**

1. 角色选择对电影进行增加、修改、删除，三个操作中的一个，并输入操作可能存在的附加信息
2. 系统接收角色选择的操作，根据操作的不同，系统向数据库发起不同的操作请求
3. 数据库根据系统发过来的请求，对电影信息进行更改，返回更改结果
4. 系统提示“操作成功”

**异常：**

3.a. 更改失败。数据库回滚操作，系统提示“操作失败，重试”，返回1

#### 1.3.1.13管理场次

**用例名：**管理场次

**用例编号：**13

**参与角色：**管理员

**前置条件：**角色已经登录

**后置条件：**无

**输入数据：**要对场次信息进行的操作，以及操作可能存在的附加信息

**输出数据：**操作结果

**事件流：**

1. 角色选择对场次进行增加、修改、删除，三个操作中的一个，并输入操作可能存在的附加信息
2. 系统接收角色选择的操作，根据操作的不同，系统向数据库发起不同的操作请求
3. 数据库根据系统发过来的请求，对场次信息进行更改，返回更改结果
4. 系统提示“操作成功”

**异常：**

3.a. 更改失败。数据库回滚操作，系统提示“操作失败，重试”，返回1

#### 1.3.1.14删除订单

**用例名：**删除订单

**用例编号：**14

**参与角色：**售票员或管理员

**前置条件：**角色已经登录

**后置条件：**无

**输入数据：**要删除的订单

**输出数据：**删除结果

**事件流：**

1. 角色选择对某个订单进行删除操作
2. 系统接收删除请求，向数据库发起删除请求
3. 数据库根据系统发过来的请求，对订单信息进行删除
4. 系统提示“操作成功”

# 2总体设计

## 2.1系统功能结构设计

### 2.1.1模块的划分和描述

本系统包含用户登录注册模块、售票管理模块、影片管理模块、场次管理模块、会员管理模块、工作人员管理模块、订单管理模块等。

### 2.1.2系统的功能结构图

电影院售票管理系统

会员管理

场次管理

影片管理

售票管理

登录、注册

注册

登录

修改会员

添加会员

会员信息查询

增加场次

影片信息查询

删除修改场次

订票

订单管询

影片信息查询

增加

影片

删除修改影片

## 

## 2.3系统的体系结构

B/S（Browser/Server）结构即浏览器和服务器结构。它是随着Internet技术的兴起，对C/S结构的一种变化或者改进的结构。在这种结构下，用户工作界面是通过WWW浏览器来实现，极少部分事务逻辑在前端（Browser）实现，但是主要事务逻辑在服务器端（Server）实现，形成所谓三层3-tier结构。这样就大大简化了客户端电脑载荷，减轻了系统维护与升级的成本和工作量，降低了用户的整体成本（TCO）。

# 3详细设计

## 3.1会员登录模块

功能：会员输入手机号和密码登陆后进入购票页面

输入项目：手机号，密码。

输出项目：跳转到会员购票页面user\_buy.html。

程序逻辑：

1. 提示输入账号密码；
2. 若数据库匹配到指定的账号密码，则跳转到指定界面；
3. 若无匹配，提示“账号或密码错误”；

测试要点：若功能正常，则会跳转至指定的界面。

## 3.2会员注册模块

功能：会员输入个人基本信息进行注册

输入项目：手机号，昵称，密码

输出项目：提示注册成功，跳转到会员登录页面。

程序逻辑：

P1 用户按提示输入手机号，昵称，密码；

P2 将用户信息插入数据库会员表中

P3 若插入成功则提示“注册成功”，自动跳转到登录页面

1. 若插入失败，则提示“注册失败”

测试要点：若功能正常，则数据库新增记录，且会跳转至指定的界面。

## 3.3管理员登录模块

功能：售票员和管理员输入账号密码，登录到各自的管理页面

输入项目：账号，密码。

输出项目：根据角色跳转到相应的管理页面。

程序逻辑：

1. 提示输入账号密码；
2. 若数据库匹配到指定的账号密码，则跳转到指定界面；
3. 若无匹配，提示“账号或密码错误”；

测试要点：若功能正常，则会跳转至指定的界面。

## 3.4添加管理员模块

功能：管理员可在管理页面添加新的售票员和后天管理员

输入项目：账号，密码，职务

输出项目：管理员列表刷新，新增了管理员信息。

程序逻辑：

P1 输入账号，职务密码

P2 将管理员信息插入数据库管理员表中

P3 若插入成功则页面刷新显示新增的管理员信息

P4若插入失败，则提示“添加失败”

测试要点：若功能正常，则数据库新增记录，页面刷新显示新增的管理员信息。

## 3.5会员个人中心模块

功能：显示会员的个人基本信息。

输入项目：无。

输出项目：会员的个人信息

程序逻辑：

1. 会员登录后进入到个人中心页面
2. 根据当前登录的会员id获取会员信息
3. 将会员信息显示到个人中心页面

测试要点：是否显示对应信息

## 3.6添加会员模块

功能：在售票员管理页面增加新的会员信息。

输入项目：会员的手机号、密码、昵称、积分和享受折扣。

输出项目：显示新的会员信息。

程序逻辑：

1. 输入对应的会员信息；
2. 将会员信息插入到数据库的会员信息表中；
3. 若插入成功，则显示新的会员信息
4. 若未成功，则提示“添加失败”

测试要点：若添加成功、通过查询功能能找对对应的会员信息。

## 3.7取票功能

功能：完成取票功能。

输入项目：无。

输出项目：购票信息。

程序逻辑：

1. 用户完成支付；
2. 将订单信息插入到数据库
3. 调动相关短信接口。发送短信给购票用户

测试要点：用户是否能收到短信

## 3.8会员信息更新模块

功能：修改会员的部分或所有信息。

输入项目：会员信息。

输出项目：更新后的信息。

程序逻辑：

1. 输入需要修改的会员折扣，积分等信息；
2. 根据会员id将新的会员信息更新到相应的数据库记录中；
3. 若更新成功，则页面刷新显示新的会员信息；

测试要点：会员信息是否进行了更新。

## 3.9会员查询模块

功能：查询指定会员的相关信息。

输入项目：指定会员的手机号。

输出项目：返回指定的会员信息。

程序逻辑：

1. 在查询功能中输入会员的手机号；
2. 根据会员手机号在数据库中检索是否有匹配的信息；
3. 若找到则返回并显示；
4. 若未找到则显示未找到相关信息；

测试要点：是否返回指定会员信息。

## 3.10电影添加模块

功能：在系统中增加新的电影信息。

输入项目：指定电影的电影名、主演、类型、时长、简介。

输出项目：显示该电影相关信息。

程序逻辑：

1. 在添加功能中输入电影信息；
2. 将电影信息插入到数据库电影表中
3. 若插入成功则页面刷新，显示新电影信息
4. 若未成功，则提示“添加失败”

测试要点：成功添加则能通过电影查询功能找到该信息。

## 3.11电影查询模块

功能：查询指定电影的相关信息。

输入项目：指定电影的电影名。

输出项目：显示指定的电影信息。

程序逻辑：

1. 在查询功能中输入该电影相关信息；
2. 在数据库中检索是否有匹配的信息；
3. 若找到则返回相应的电影信息；
4. 若未找到则返回并显示“未找到相关信息”；

测试要点：是否返回指定电影信息。

## 3.12电影删除模块

功能：删除指定电影的所有信息。

输入项目：鼠标点击事件。

输出项目：无。

程序逻辑：

1. 在电影列表中选定要删除的电影；
2. 点击“删除”按钮
3. 根据选择的电影id在数据库中删除相应的电影信息；
4. 若删除成功，则页面刷新，删除的电影信息已经消失

测试要点：删除成功后电影信息不再显示且电影查询功能无法找到相关信息。

## 3.13电影更新模块

功能：修改指定电影的部分或所有信息。

输入项目：电影的部分或所有信息。

输出项目：显示修改后的电影信息。

程序逻辑：

1. 选择列表中某个电影，点击“编辑”按钮；
2. 输入新的电影信息，确认保存
3. 根据电影id在数据库中更新相应的电影信息
4. 若更新成功，则页面刷新显示新的电影信息；

测试要点：页面中电影的信息是否更新了。

## 3.14场次增加模块

功能：在系统中添加具体电影场次。

输入项目：日期，电影名称，厅号，开始时间和价格。

输出项目：显示新增场次。

程序逻辑：

1. 输入对应的场次相关信息；
2. 将场次信息插入到数据库中
3. 若添加成功，则刷新页面显示新增的场次信息
4. 若未成功，则提示“添加失败”

测试要点：若添加成功，在场次管理界面能显示对应信息

## 3.15场次删除模块

功能：删除指定场次。

输入项目：鼠标点击事件。

输出项目：无。

程序逻辑：

1. 在场次列表中选定要删除的场次；
2. 点击“删除”按钮
3. 根据选择的场次id在数据库中删除相应的场次信息；
4. 若删除成功，则页面刷新，删除的场次信息已经消失

测试要点：删除成功后场次信息不再显示且场次查询功能无法找到相关信息

## 3.16场次更新模块

功能：更新场次信息。

输入项目：场次的日期，厅号，时间和价格。

输出项目：显示修改后的场次信息。

程序逻辑：

1. 选择列表中某个场次，点击“编辑”按钮；
2. 输入新的场次信息，确认保存
3. 根据电场次id在数据库中更新相应的场次信息
4. 若更新成功，则页面刷新显示新的场次信息；

测试要点：页面中场次的信息是否更新了。

## 3.17场次查询

功能：在系统中查询具体电影场次。

输入项目：电影名称。

输出项目：显示该电影的所有场次。

程序逻辑：

**P1**:输入需要查询的电影名称；

**P2**：根据电影id在数据库中检索是否有匹配的信息；

**P3**：若找到则返回场次信息；

**P4：**若是没找到。则实现无记录

测试要点：场次信息是否返回并显示。

## 3.18购票功能

功能：用户在购票页面选择相应场次的电影购票。

输入项目：鼠标点击事件。

输出项目：跳转至选座页面。

程序逻辑：

1. 将数据库中电影的场次信息显示在购票页面
2. 检测鼠标事件；
3. 根据用户选择的场次信息，跳转到选座页面

测试要点：点击购票后能跳转至选座页面，并显示正确的场次信息。

## 3.19选座功能

功能：用户选择观影座位。

输入项目：鼠标点击事件。

输出项目：座位名称、座位数量、总价。

程序逻辑：

1. 根据用户选择的场次id来查询座位信息
2. 显示已被预订座位，且无法产生鼠标点击事件；
3. 检测鼠标点击座位事件，显示用户选择的座位名称数量和总价
4. 当用户点击：“支付”按时，跳转到支付页面

测试要点：选中座位后右侧能显示指定信息，点击支付后进入支付页面。

## 3.20支付功能

功能：用户完成支付。

输入项目：鼠标点击事件。

输出项目：提示购票成功，进入取票页面。

程序逻辑：

1. 获取之前用户选择的场次和座位信息
2. 显示订单信息和总额
3. 提示用户在规定时间内完成支付；
4. 若用户取消，则返回；
5. 若点击“确认支付”，提示购票成功，进入取票页面；

测试要点：用户点击支付后，进入取票页面，用户会收到购票成功的短信。

## 3.21订单查看

功能：查看订单信息。

输入项目：无。

输出项目：显示订单id、取票码、会员手机、时间、座位数目、折后价格。

程序逻辑：

1. 管理员进入订单管理页面，触动查询事件
2. 返回数据库中已有的订单信息；
3. 显示在订单管理界面；

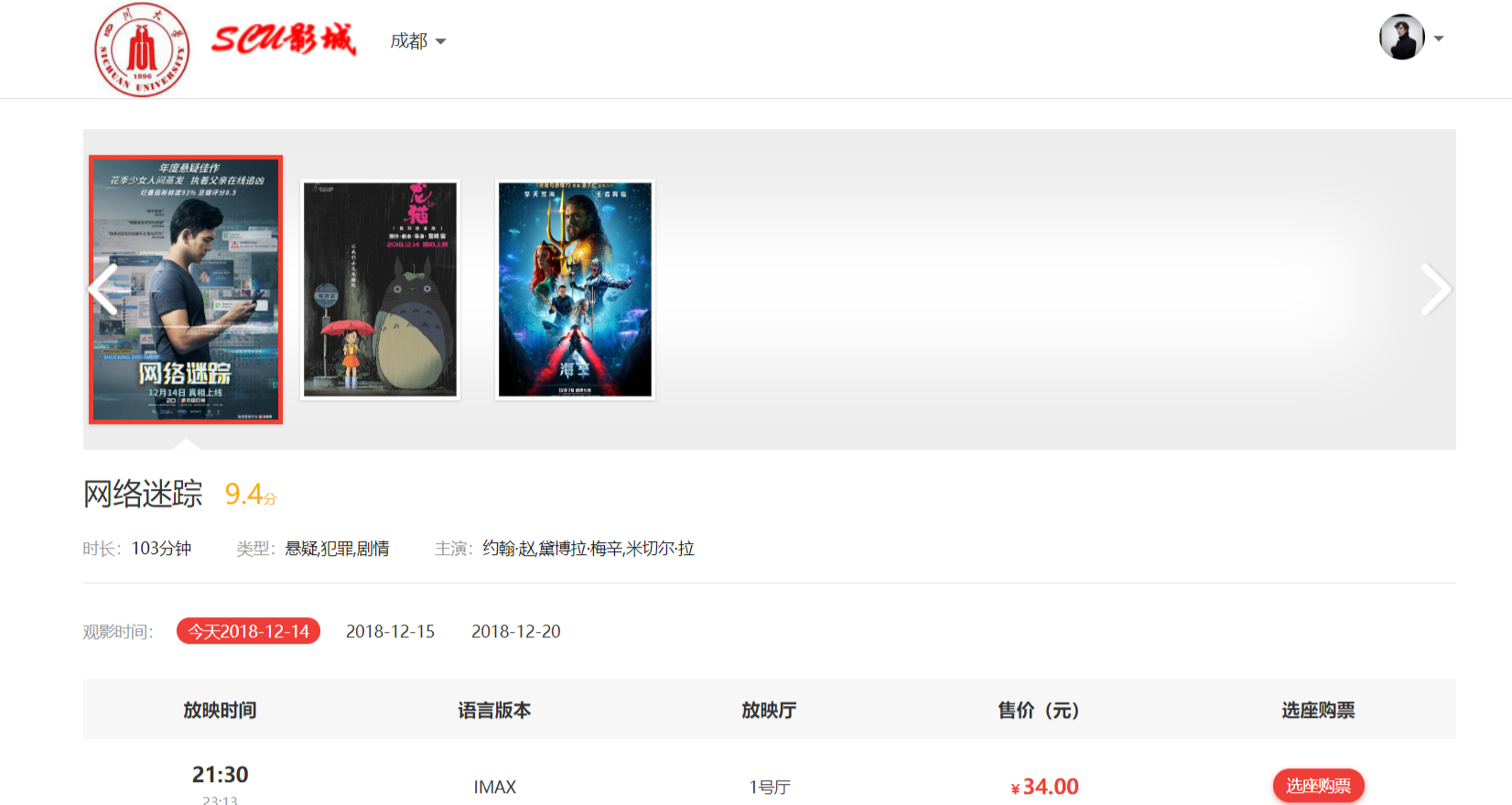
测试要点：是否显示订单信息

# 4接口设计

## 4.1界面接口设计

### 4.1.1用户界面接口设计

#### 4.1.1.1用户购买页面

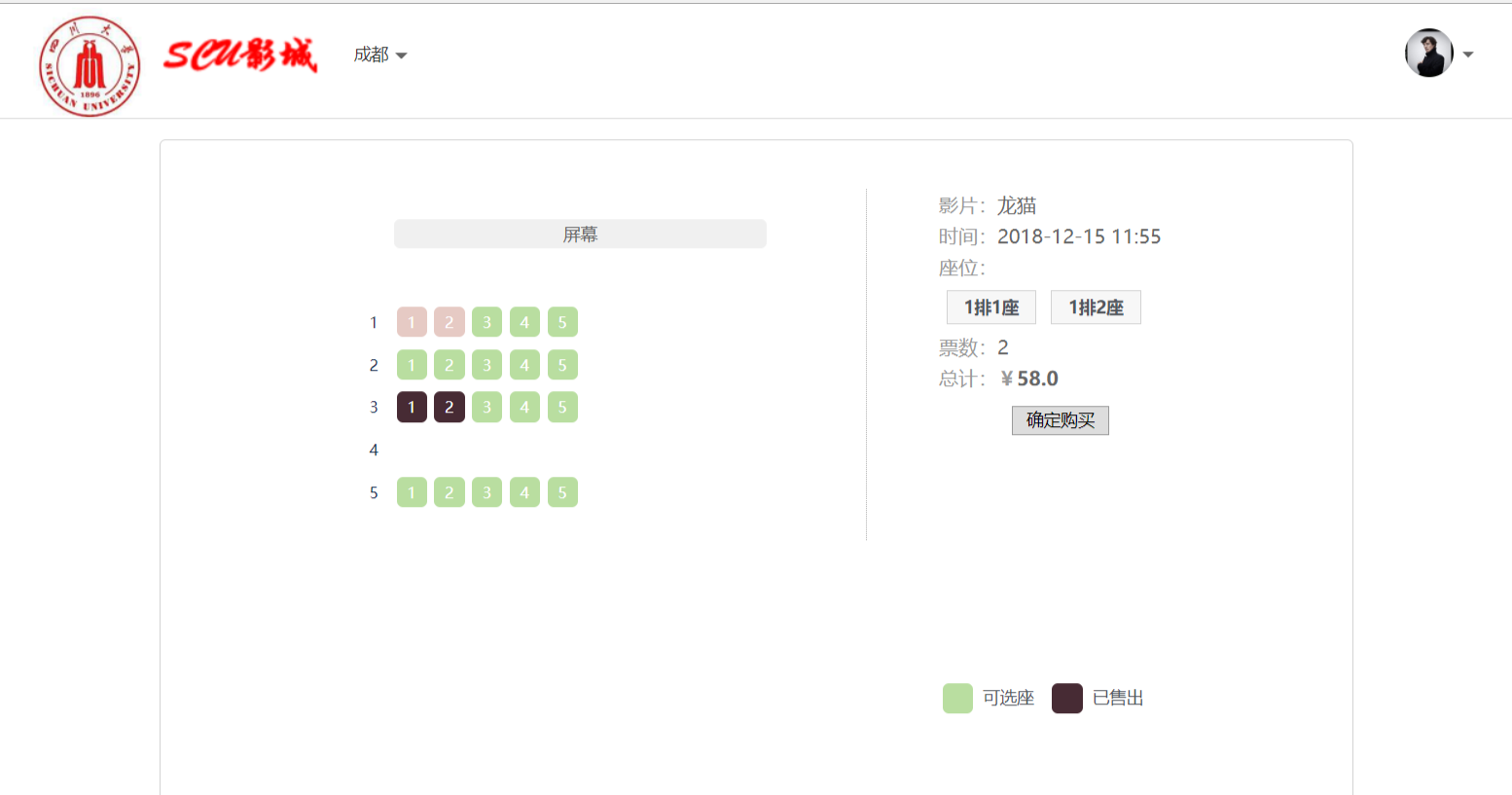


显示信息：后端数据库的票价信息、排片信息、电影信息等等

右上角为用户登录后的头像，可以查看和修改基本信息和退出登录。

#### 4.1.1.2用户选票界面

用户可以选择目前没有被购买的座位点击确定购买



输入信息：鼠标选择座位位置

输出信息：在右边显示选中的座位号

#### 4.1.1.3用户支付界面



输入信息：上个页面传来的座位信息与票价信息

输出信息：购买的总费用

#### 4.1.1.4取票码页面

用户购买成功后，给用户提供取票码

显示信息：用户购买的电影信息、

#### 4.1.1.5用户信息界面

显示信息：包括昵称、会员积分、享受折扣、手机号等。

### 4.1.2管理界面

#### 4.1.2.1管理登录界面

输入信息：管理人员账号

#### 4.1.2.2注册界面

输入信息：会员手机号、会员昵称、账号密码、再次确认密码

勾选同意注册协议

#### 4.1.2.3管理员查看信息界面

管理员可以进行用户管理、会员管理、电影管理、场次管理与订单管理。

**拥有增加、删除、修改、查询、导出、打印的功能。**

其中每个模块的信息与数据库的字段基本吻合，故仅展示用户管理为例。

**用户管理（包括管理员与售票员）**

显示信息有：序号、id、用户账号、密码、职务

#### 4.1.2.4售票员查看信息界面

售票员可以进行会员管理、订单管理的信息查看与售票功能

相较管理员，售票员没有管理用户的功能与电影、场次修改的功能。

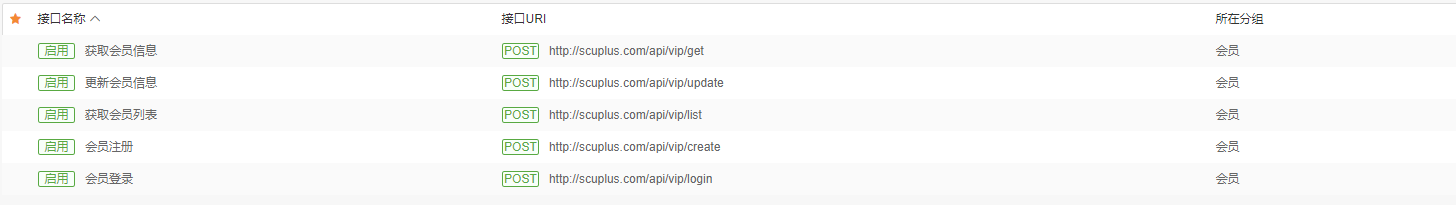
**订单管理（电影票购买记录）**

显示信息有：订单id、取票码、时间、会员id、座位数据、折扣后价格、条目的创建时间、条目的最后更新时间。

## 4.2数据库接口设计

我们将数据库发布到云端，可以通过访问云服务器的API来对数据库信息进行操作

### 4.1.3.1会员信息接口



以更新会员信息为例：

请求参数： id(会员的id号)

Data（数据）

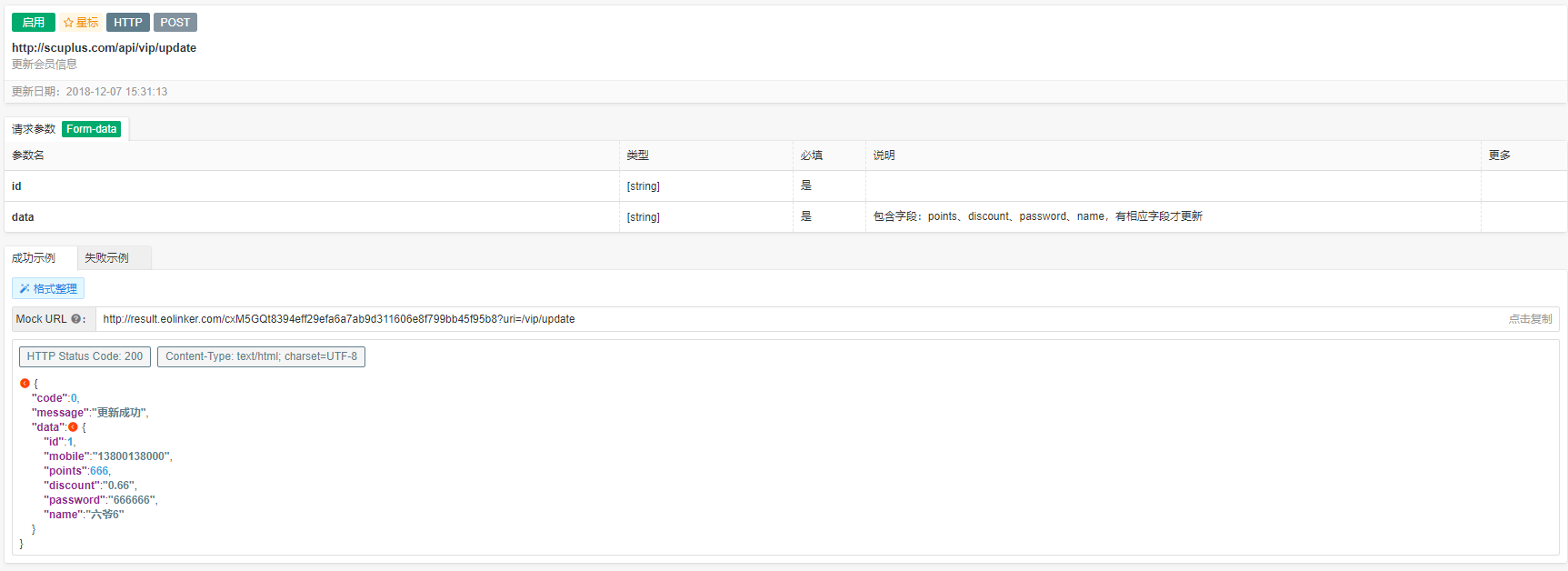
返回值： id（会员id）

Mobile（会员手机号）

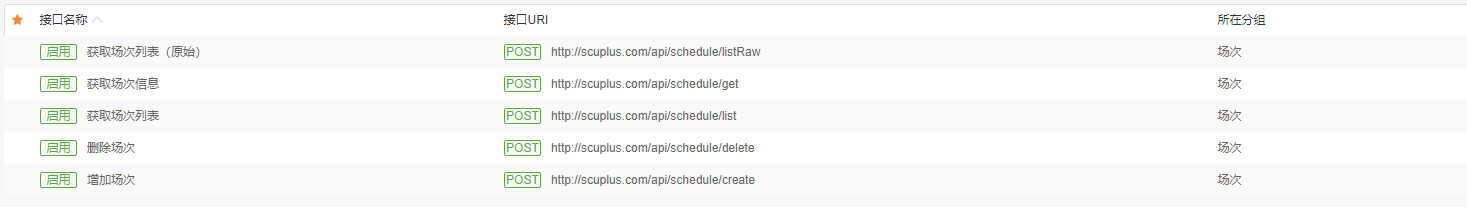
points（会员的积分）

password（会员密码）

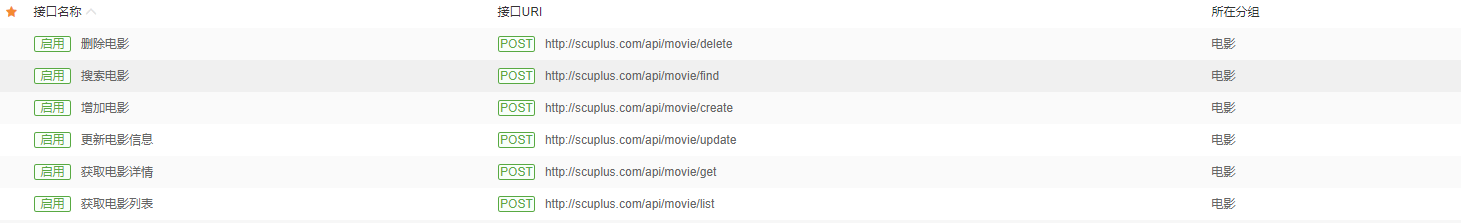
name（会员名称）



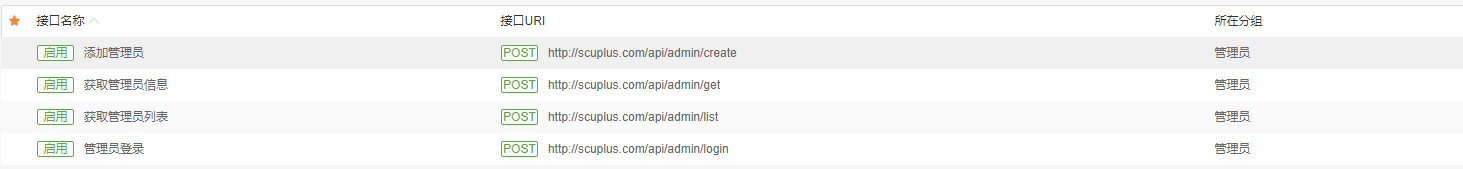
### 4.1.3.2电影场次信息接口



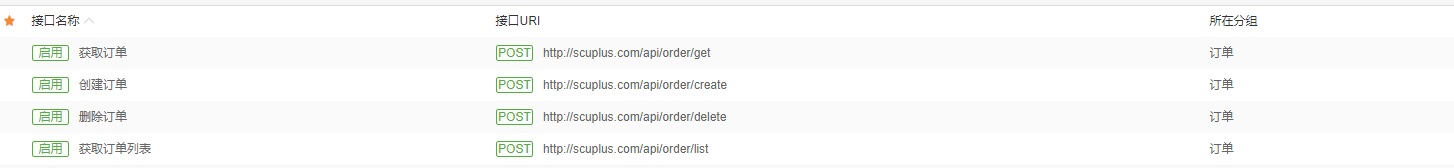
### 4.1.3.3电影信息接口



### 4.1.3.4管理员信息接口



### 4.1.3.5订单信息接口



### 4.1.3.6状态码



## 4.3外部接口

### 4.3.1用户的输入和反馈的结果

在输入方面，通过键盘、鼠标进行输入操作。在输出方面，系统通过网页浏览器显示结果，从而实现输出结果反馈。

### 4.3.2 和其他系统之间的接口。

本系统在网络中是以网站的形式独立运行的，与其他系统之间没有太多的关联，充其量也就是部分友情链接，故接口将以超链接方式实现。

## 4.4内部接口

### 4.4.1模块的接口：模块的输入和输出。

本系统的用户分为管理员和售票员和会员，不同用户所对应的功能模块不同。

（1）数据库连接模块。系统中因为涉及到一系列对数据库查询、写入、修改、删除操作，故在每个功能模块加载之初就建立与数据库的连接，即数据库模块接口，在此基础上实现以下操作。

Page\_load

（2）登录模块

Login

输入：用户名、密码，选择登录类型

输出：根据登陆身份不同反馈相应的功能界面

（3）注册模块

Regist

输入：会员输入用户名、密码、手机号

输出：根据输入信息是否符合要求显示注册成功或信息有误

（4）影片信息查询模块

movieManage

输入：管理员登录的前提下，选择刷新，或者输入查询条件（电影名称、开始日期、结束日期），也可以选择输出某种文件格式或打印

输出：显示全部电影信息或者相应电影信息包括电影id、电影名、主演、简介、片长、类型、操作类型。或者启动打印机或成功输出文件。

（5）影片增加模块

movieManage

输入：管理员登录的前提下，输入新的影片相关信息包括电影名称、主演、简介、语言、片长、类型

输出：成功添加影片

（6）影片修改删除模块

movieManage

输入：管理员登录的前提下，输入影片新的相关信息

输出：刷新显示修改后的结果，也可以删除该影片

（7）场次信息查询模块

scheduleManage

输入：管理员登录的前提下，选择刷新，也可以选择输出某种文件格式或打印

输出：显示全部场次信息包括场次id、电影id、开始时间、厅id、价格、日期、操作类型。或者启动打印机或成功输出文件。

（8）场次增加模块

scheduleManage

输入：管理员登录的前提下，输入新的场次相关信息包括电影编号、开始时间、厅号、价格

输出：成功添加场次

（9）场次修改删除模块

scheduleManage

输入：管理员登录的前提下，输入场次新的相关信息

输出：刷新显示修改后的结果，也可以删除该场次

（10）订票模块

User\_buy->choose\_seat->pay->ticket

输入：售票员或会员登录的前提下，选择相应的电影、场次、座位（可多选）

输出：订票成功

（11）订单管理模块

orderManage

输入：售票员或管理员登录的前提下，选择刷新

输出：显示订票的相关信息订单id、取票码、时间、会员id、座位数目、折后价格、创建时间、最后更新时间、操作类型，也可以新增修改删除某一订单。

（12）会员新增模块

showVip

输入：管理员登录的前提下，输入会员手机、密码、昵称、积分、享受折扣信息

输出：会员新增成功

（13）会员信息查询模块

showVip

输入：管理员登录的前提下，选择刷新，或者输入查询条件（会员手机号），也可以选择输出某种文件格式或打印

输出：显示全部电影信息或者相应会员信息包括会员id、会员手机、昵称、积分、享受折扣、操作类型。或者启动打印机或成功输出文件。

（14）会员修改模块

showVip

输入：管理员登录的前提下，输入会员新的相关信息

输出：刷新显示修改后的结果，也可以删除该影片会员

# 5数据库设计

## 5.1数据库逻辑结构设计

电影（电影id，电影名称，时长，简介，类型，主演）

场次（场次id，电影id，开始时间，厅id，原价，日期）

座位（座位id，厅id，厅内行号，厅内列号，座位名称）

订单（订单id，取票码，下单时间，来源，下单者id，本单座位数，实付金额）

订座记录（记录id，场次id，座位id，订单id）

会员（会员id，手机号，积分，折扣，密码，用户名称）

工作人员（工作人员id，工作人员类型，账号，密码）

厅（厅id，名称，行数，列数）

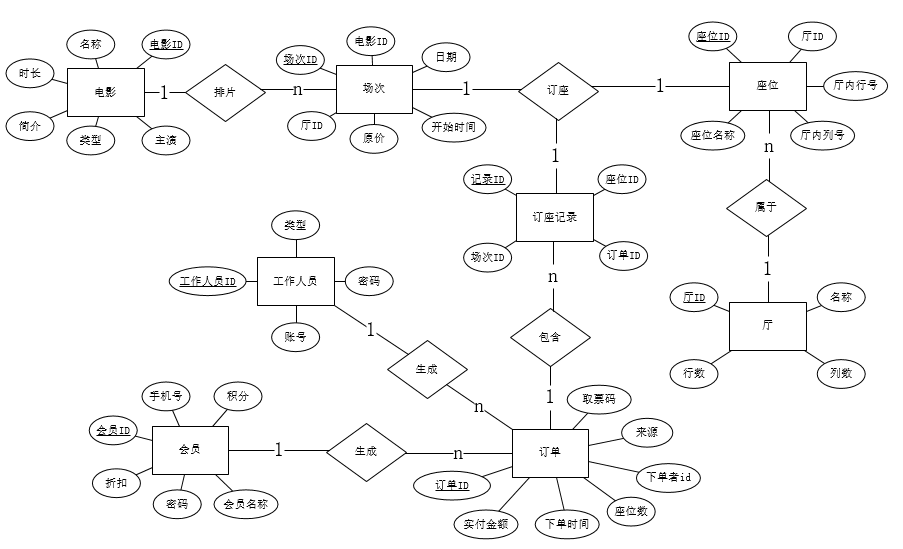


图5.1-1 ER图

## 5.2数据库物理结构设计

电影表movies

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段英文名 | 字段中文名 | 是否必须项 | 字段类型 |
| 1 | id | 电影id（主键） | 是 | int unsigned |
| 2 | name | 名称 | 是 | tinytext |
| 3 | duration | 时长（分钟） | 是 | smallint unsigned |
| 4 | description | 简介 | 否 | text |
| 5 | type | 类型 | 是 | tinytext |
| 6 | star | 主演 | 是 | tinytext |

场次表schedules

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段英文名 | 字段中文名 | 是否必须项 | 字段类型 |
| 1 | id | 场次id（主键） | 是 | int unsigned |
| 2 | movie\_id | 电影id | 是 | int unsigned |
| 3 | start\_time | 开始时间 | 是 | char(5) |
| 4 | room\_id | 厅id | 是 | tinyint |
| 5 | price | 原价 | 是 | decimal(5.2) |
| 6 | date | 日期 | 是 | date |

订座记录表order\_seats

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段英文名 | 字段中文名 | 是否必须项 | 字段类型 |
| 1 | id | 记录id（主键） | 是 | int unsigned |
| 2 | schedule\_id | 场次id | 是 | int unsigned |
| 3 | seat\_id | 座位id | 是 | int unsigned |
| 4 | order\_id | 订单id | 是 | int unsigned |

座位表seats

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段英文名 | 字段中文名 | 是否必须项 | 字段类型 |
| 1 | id | 座位id（主键） | 是 | int unsigned |
| 2 | room\_id | 厅id | 是 | tinyint |
| 3 | row | 厅内行号 | 是 | tinyint |
| 4 | col | 厅内列号 | 是 | tinyint |
| 5 | name | 座位名称（几排几号） | 是 | tinytext |

订单表orders

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段英文名 | 字段中文名 | 是否必须项 | 字段类型 |
| 1 | id | 订单id（主键） | 是 | int unsigned |
| 2 | token | 取票码 | 是 | char(12) |
| 3 | time | 下单时间 | 是 | int unsigned |
| 4 | source | 来源（用户或工作人员） | 是 | tinyint(1) |
| 5 | user\_id | 下单者id（用户或工作人员） | 是 | int unsigned |
| 6 | num | 本单座位数 | 是 | tinyint |
| 7 | price | 实付金额 | 是 | decimal(5.2) |

工作人员表admins

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段英文名 | 字段中文名 | 是否必须项 | 字段类型 |
| 1 | id | 工作人员id（主键） | 是 | int unsigned |
| 2 | role | 工作人员类型 | 是 | tinyint |
| 3 | account | 账号 | 是 | int unsigned |
| 4 | password | 密码 | 是 | text |

会员表vips

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段英文名 | 字段中文名 | 是否必须项 | 字段类型 |
| 1 | id | id（主键） | 是 | int unsigned |
| 2 | mobile | 手机号 | 是 | char(11) |
| 3 | points | 积分 | 是 | int unsigned |
| 4 | discount | 折扣 | 是 | decimal(3.2) |
| 5 | password | 密码 | 是 | text |
| 6 | name | 用户名称 | 是 | tinytext |

厅表rooms

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段英文名 | 字段中文名 | 是否必须项 | 字段类型 |
| 1 | id | 厅id（主键） | 是 | int unsigned |
| 2 | name | 名称 | 是 | tinytext |
| 3 | row | 行数 | 是 | tinyint |
| 4 | col | 列数 | 是 | tinyint |

由于系统启用了软删除功能，因此部分数据表还有三个附加字段（在上述表中未列明）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 字段英文名 | 字段中文名 | 是否必须项 | 字段类型 |
| 1 | created\_at | 创建时间 | 否，默认NULL | int unsigned |
| 2 | updated\_at | 更新时间 | 否，默认NULL | int unsigned |
| 3 | deleted\_at | 删除时间 | 否，默认NULL | int unsigned |

# 6个人工作和总结

**6.1做的具体工作：**

我们小组做的项目是影院售票系统，项目的工作分为web前端，java后端和数据库三个部分。作为小组的组长，我承担了全部java后端的开发、部分js编写、系统的集成测试工作。

**技术描述**：

1. SSM框架

SSM（Spring+SpringMVC+MyBatis）框架集由Spring、SpringMVC、MyBatis三个开

源框架整合而成，常作为数据源较简单的web项目的框架。其中spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。SpringMVC分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。MyBatis是一个支持普通SQL查询，存储过程和高级映射的优秀持久层框架。

本项目中我使用了Spring和SpringMVC框架，数据库配置在云端服务器上

1. Httpclient

越来越多的 Java 应用程序需要直接通过 HTTP 协议来访问网络资源。虽然在 JDK 的 java.net 包中已经提供了访问 HTTP 协议的基本功能，但是对于大部分应用程序来说，JDK 库本身提供的功能还不够丰富和灵活。HttpClient 用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且它支持 HTTP 协议最新的版本和建议。

本项目中主要通过httpclient实现的doPost（）方法来向云端数据库发送和请求json数据。

1. Fastjson

fastjson是阿里巴巴的开源JSON解析库，它可以解析JSON格式的字符串，支持将Java

Bean序列化为JSON字符串，也可以从JSON字符串反序列化到JavaBean。

本项目的数据多为json格式，多处用到fastjson对json的解析处理功能。

1. MD5加密算法

MD5，即"Message-Digest Algorithm 5（信息-摘要算法）"。MD5的最大作用在于，将不

同格式的大容量文件信息在用数字签名软件来签署私人密钥前"压缩"成一种保密的格式，关键之处在于这种"压缩"是不可逆的。

本项目使用MD5算法来对用户的密码进行加密和解密，保证网络传输过程中的数据安全。

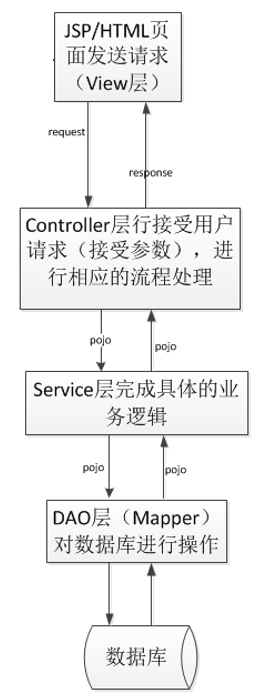
1. jQuery，Ajax

jQuery是一个快速，小巧，功能丰富的JavaScript库。 它通过易于使用的API在大量浏览器中运行，使得HTML文档遍历和操作，事件处理，动画和Ajax更加简单。Ajax用于浏览器与服务器通信而无需刷新整个页面，服务器将不再返回整个页面，而是返回少量数据，通过JavaScript DOM更新一部分节点。

本项目使用ajax() 方法用于执行 AJAX（异步 HTTP）请求，与后端的controller进行数据交互。所有的 jQuery AJAX 方法都使用 ajax() 方法。

**具体实现：**

三层架构运行流程



1. SSM框架中各层级间的作用及关系

* Spring和SpringMVC的配置文件：

├── WEB-IN

│ └── resource

│ ├──Spring

│ └──applicationContext-service.xml

│ └──springmvc.xml

│ ├── web.xml

* 实体层（entity）

定义需要的实体类，与数据库表相对应

├── Entity

│ ├── Mvoie

│ ├── Admin

│ ├── User

│ ├── Schedule

│ ├── Ticket

│ ├── Room

* 表现层（springMVC）：Controller层

• 负责具体的业务模块流程的控制

• Controller层通过要调用Service层的接口来控制业务流程，控制的配置也在Spring配置文件里面。

目录结构

├── Controller

│ ├── AdminController

│ ├── MovieController

│ ├── UserController

│ ├── ScheduleController

│ ├── TicketController

│ ├── client

* 业务层（Spring）：Service层

• Service层：负责业务模块的逻辑应用设计。

• 首先设计其接口，然后再实现他的实现类。

• 通过对Spring配置文件中配置其实现的关联，完成此步工作，我们就可以通过调用Service的接口来进行业务处理。

• 最后通过调用DAO层已定义的接口，去实现Service具体的实现类。

├── Service

│ └── Impl

│ ├──AdminServiceImpl

│ └──UserServiceImpl

│ └──TicketServiceImpl

│ └──ScheduleServiceImpl

│ └──TicketServiceImpl

│ ├── AdminService

│ ├── UserService

│ ├── TicketService

│ ├── ScheduleService

│ ├── TicketService

* 持久层：Mapper层

• Mapper层：负责与数据库进行交互设计，用来处理数据的持久化工作。

• Mapper层的设计首先是设计Mapper的接口，

• 然后在Spring的配置文件中定义此接口的实现类，就可在其他模块中 调用此接口来进行数据业务的处理，而不用关心接口的具体实现类是 哪个类，这里用到的就是反射机制， Mapper层的数据源配置，以及有关数据库连接的参数都在Spring的配置文件中进行配置。

├── Mapper

│ ├── AdminMapper

│ ├── MovieMapper

│ ├── UserMapper

│ ├── ScheduleMapper

│ ├── TicketMapper

* 视图层：View层

• 负责前台html、jsp页面的展示。

• 此层需要与Controller层结合起来开发。

* 各层间的联系：

• 本来Controller层与View层是可以放在.html和.jsp文件里一起开发的，但是为了降低代码的复杂度，提高其可维护性，将其分为了这两层，这也体现了MVC框架的特性，即结构清晰，耦合度低。

• Service层是建立在Mapper层之上的，建立了Mapper层后才可以建立Service层，而Service层又是在Controller层之下的，因而Service层应该既调用Mapper层的接口，又要提供接口给Controller层的类来进行调用，它刚好处于一个中间层的位置。每个模型都有一个Service接口，每个接口分别封装各自的业务处理方法。

* Httpclient



**6.2个人总结：**

1、在这次项目的开发中，我作为小组的组长，首先在团队的合作中获得了一些收获。在项目的最后阶段，小组的人员分工合作。写代码的写代码，写文档的写文档，做最后演示文稿的做演示文稿。效率非常的快，这让我体会到了团队合作的力量。通过这次项目。我熟悉了一个项目从需求分析到最后的编码交互的整个阶段要做的事情，也让我们的组员在整个流程中做了自己应该做的那一份工作，这让我将来在走上工作岗位以后，能够尽快的上手自己的工作。为将来打下了基础。

2、在项目中我作为一名编码人员。我更加深入的理解了开发模式这个概念。学习并且运用了MVC开发模式实现前台显示和后台功能的分离和Mapper实现对数据库的相关操作。通过编写代码我更加深刻的理解了java作为面向对象的编程语言给我们编程带来的方便，以及一些开发模式给后期维护带来的好处。下面我谈谈面向我对面向对象的开发，在面向对象的编程过程中，首先应该考虑什么类会用得很平凡。比如做这个系统，主要会对电影、场次，以及订单信息这几个类有很多的操作。其中MVC模式和Mapper模式的引入，是为了将来在维护阶段，能够可以通过保护以前功能的情况下，修改或新增相应的代码实现功能的更新和添加。实现了高类聚，低耦合的开发原理。

3、这学期，在开发的初期会出现小组成员思路的不统一，团队精神的出现有效的解决了这个问题。在一个项目的开发过程中，每个人都有自己特殊的角色，但是角色的划分是能力的初显，在过程中也有能力的展现，我通过这学期充分了解到在团队精神的支撑下，自己的能力要自己主动展现。