实验十二 编写程序

实验目的：

1. 掌握编程的标准和过程

2. 编程指导原则：讨论程序编写方式与规则

3. 文档：内部文档、外部文档

实验内容：

1. 结合自己的项目，指出其中的数据管理、异常处理、UI方面的设计。

* **数据管理：**

我们的项目的程序设计中根据需求给出了存储和恢复持久性对象的方法。数据管理考虑的就是性能与空间相关的系统需求。根据对数据需求和相关约束的理解，须列出对象及其操作的设计，完成这个任务共分4个步骤。

(1)确认数据、数据结构及它们之间的关系。

(2)设计管理数据结构及其互相关系的服务。

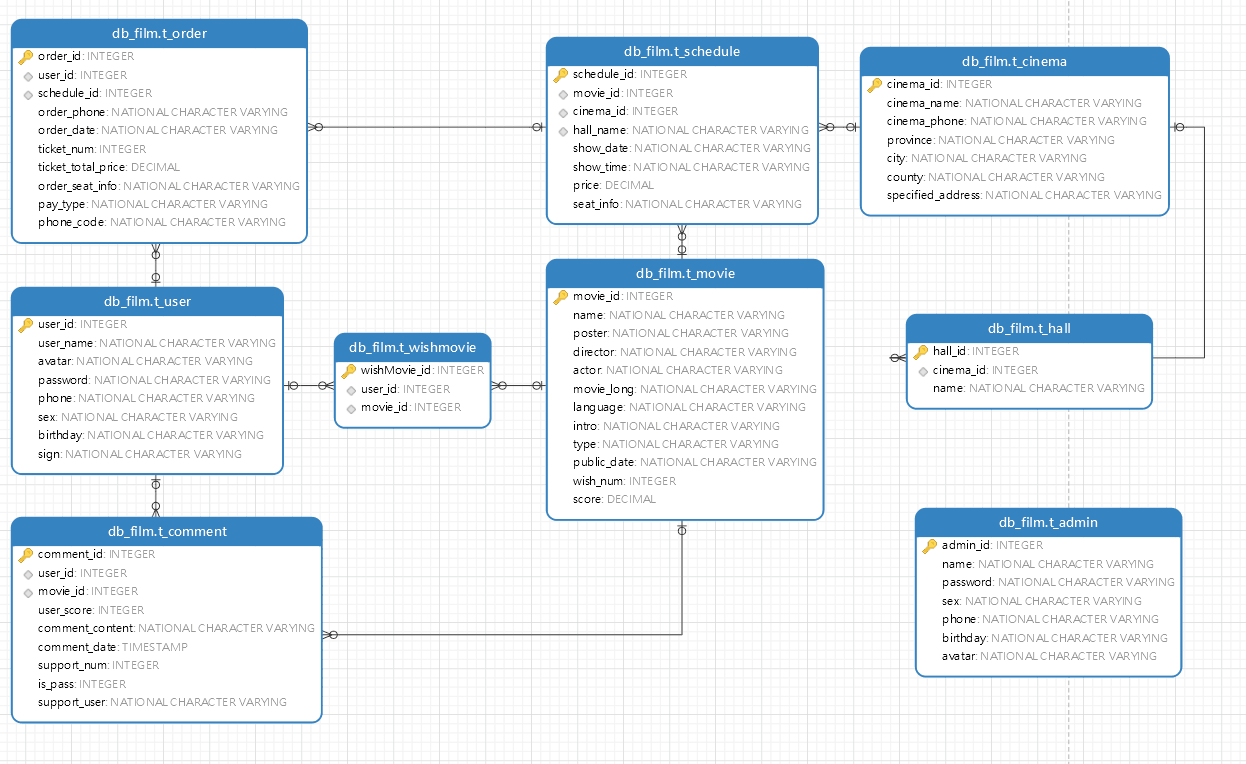
(3)找到工具，例如数据库管理系统，来实现某些数据管理任务。

(4)设计类和类层次，来检查数据管理功能。

主要以下计中数据的类（下述数据都是以类组的形式存在）：

* 管理员数据类
* 电影院信息数据类
* 评论数据类
* 电影厅数据类
* 电影数据类
* 订单数据类
* 排片数据类
* 用户数据类

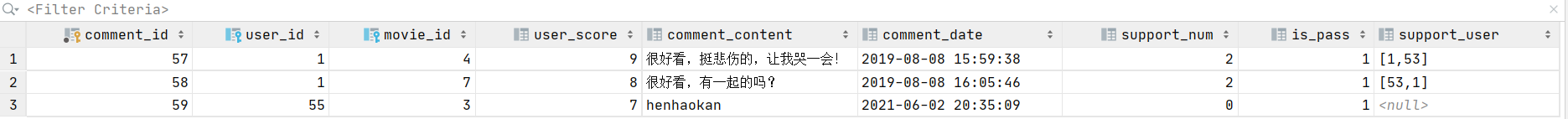
数据库-数据结构相互关系-总览图：



核心数据类别：

|  |  |
| --- | --- |
| 类别 | 主键 |
| 评论 | comment\_id |
| 电影院 | cinema\_id |
| 影厅 | hall\_id |
| 电影 | movie\_id |
| 用户 | user\_id |
| 管理员 | admin\_id |
| 订单 | order\_id |
| 排片 | schedule\_id |

* 评论数据：

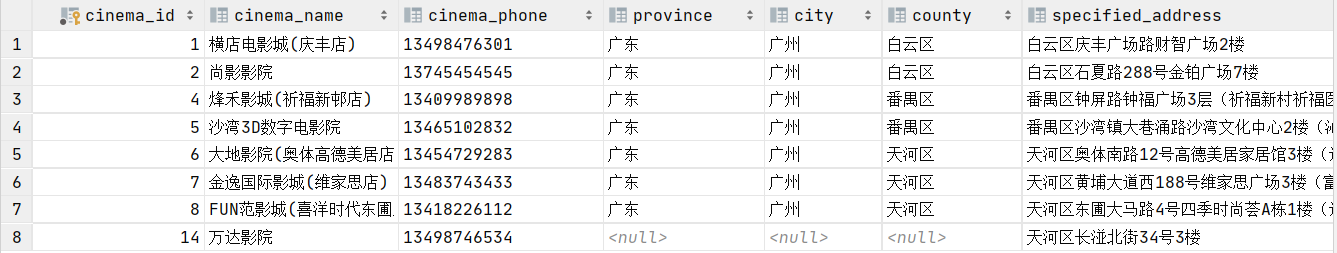


主键：comment\_id

外键：user\_id、movie\_id

|  |  |
| --- | --- |
| 属性 | 功能 |
| user\_score | 用户评分 |
| comment\_content | 用户评价内容 |
| comment\_data | 用户评价时间 |
| support\_num | 该条评论点赞数目 |
| is\_pass | 该条评论是否通过审核 |
| support\_user | 点赞用户数组(其中每个元素是外键) |

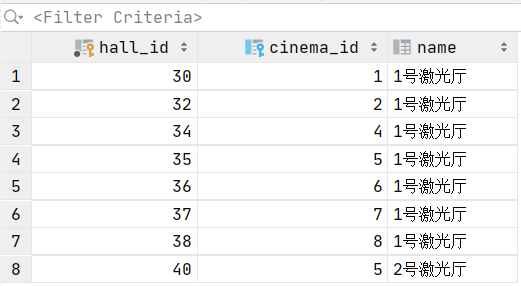
* 电影院数据：



主键：cinema\_id

|  |  |
| --- | --- |
| 属性 | 功能 |
| cinema\_name | 电影名称 |
| cinema\_phone | 海报图片存储位置 |
| province | 省 |
| city | 城市 |
| county | 区 |
| specified\_address | 具体位置 |

* 影厅数据：

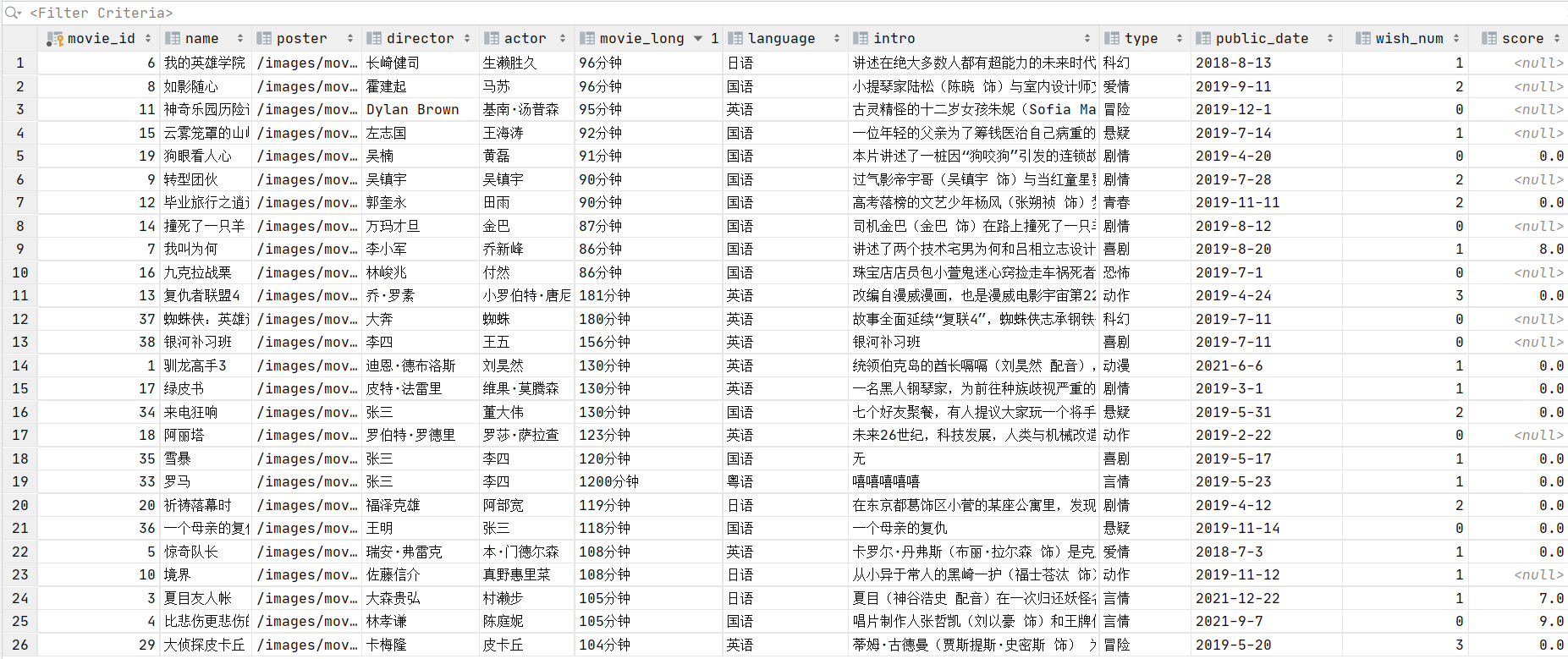


主键：hall\_id

外键：cinema\_id

|  |  |
| --- | --- |
| 属性 | 功能 |
| name | 影厅名称 |

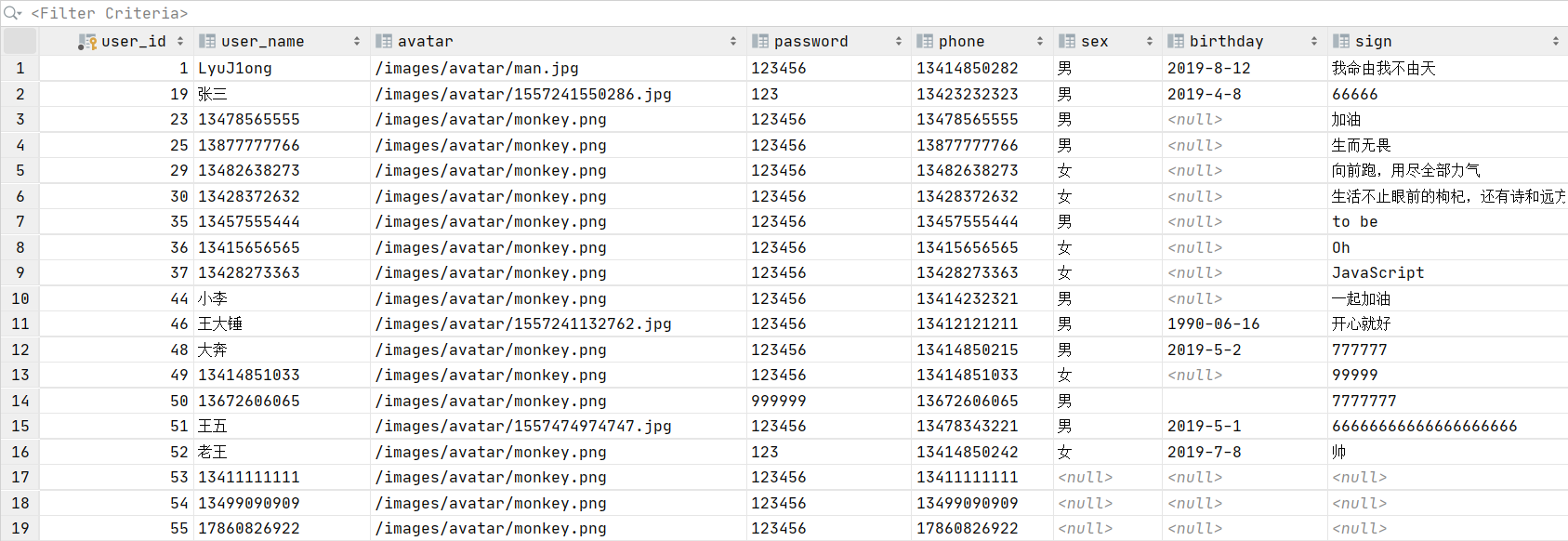
* 电影数据：



主键：movie\_id

|  |  |
| --- | --- |
| 属性 | 功能 |
| name | 电影名称 |
| poster | 海报图片存储位置 |
| director | 导演 |
| actor | 参演演员表 |
| move\_long | 电影时长 |
| language | 语言 |
| intro | 电影简介 |
| type | 电影类型 |
| public\_date | 公演日期 |
| wish\_num | 期待数 |
| score | 评价分数 |

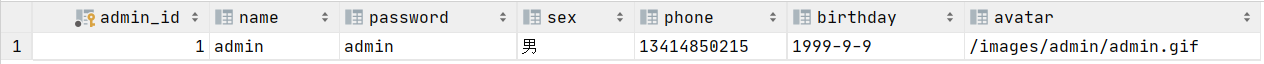
* 用户数据：



主键：user\_id

|  |  |
| --- | --- |
| 属性 | 功能 |
| user\_name | 用户名 |
| avatar | 头像图片存储位置 |
| password | 登陆密码 |
| phone | 手机号码 |
| sex | 性别 |
| birthday | 生日 |
| sign | 签名 |

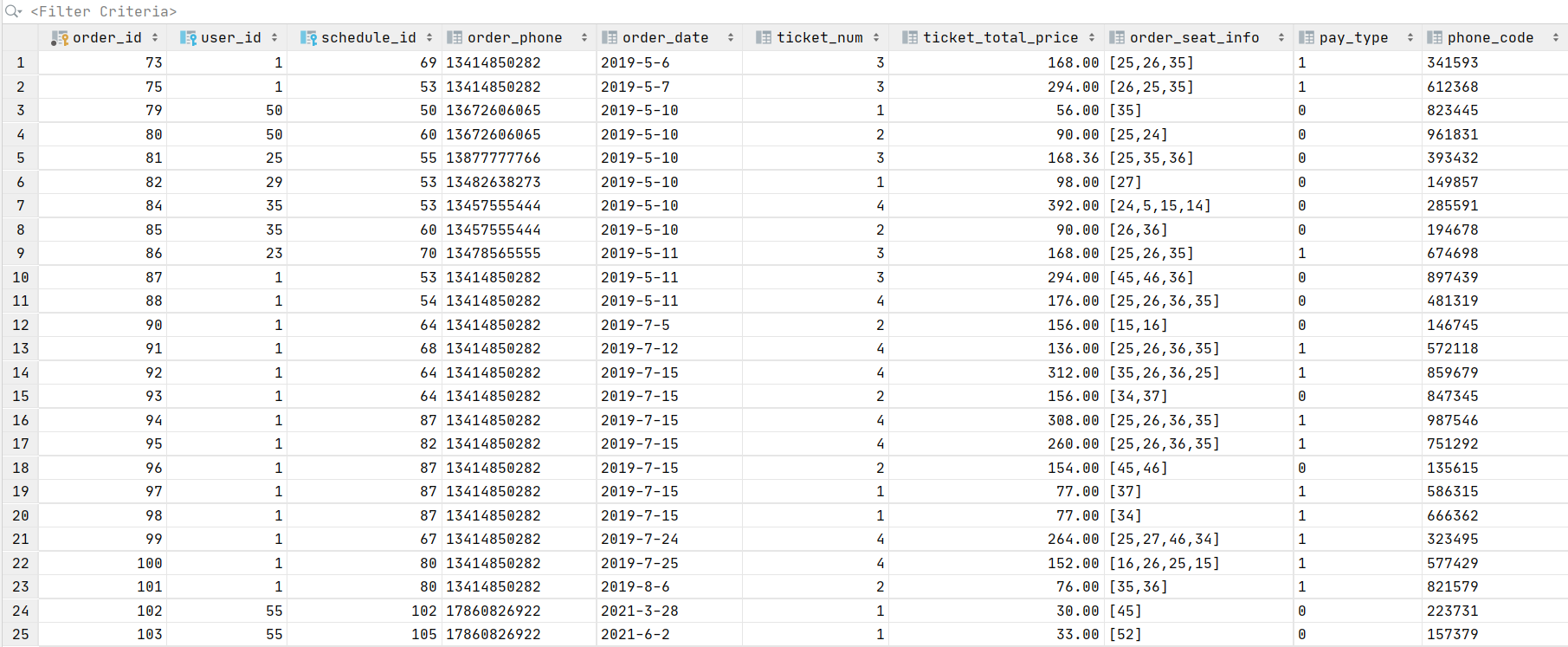
* 管理员数据：



主键：admin\_id

|  |  |
| --- | --- |
| 属性 | 功能 |
| name | 管理员名 |
| avatar | 头像图片存储位置 |
| password | 登陆密码 |
| phone | 手机号码 |
| sex | 性别 |
| birthday | 生日 |

* 订单数据：

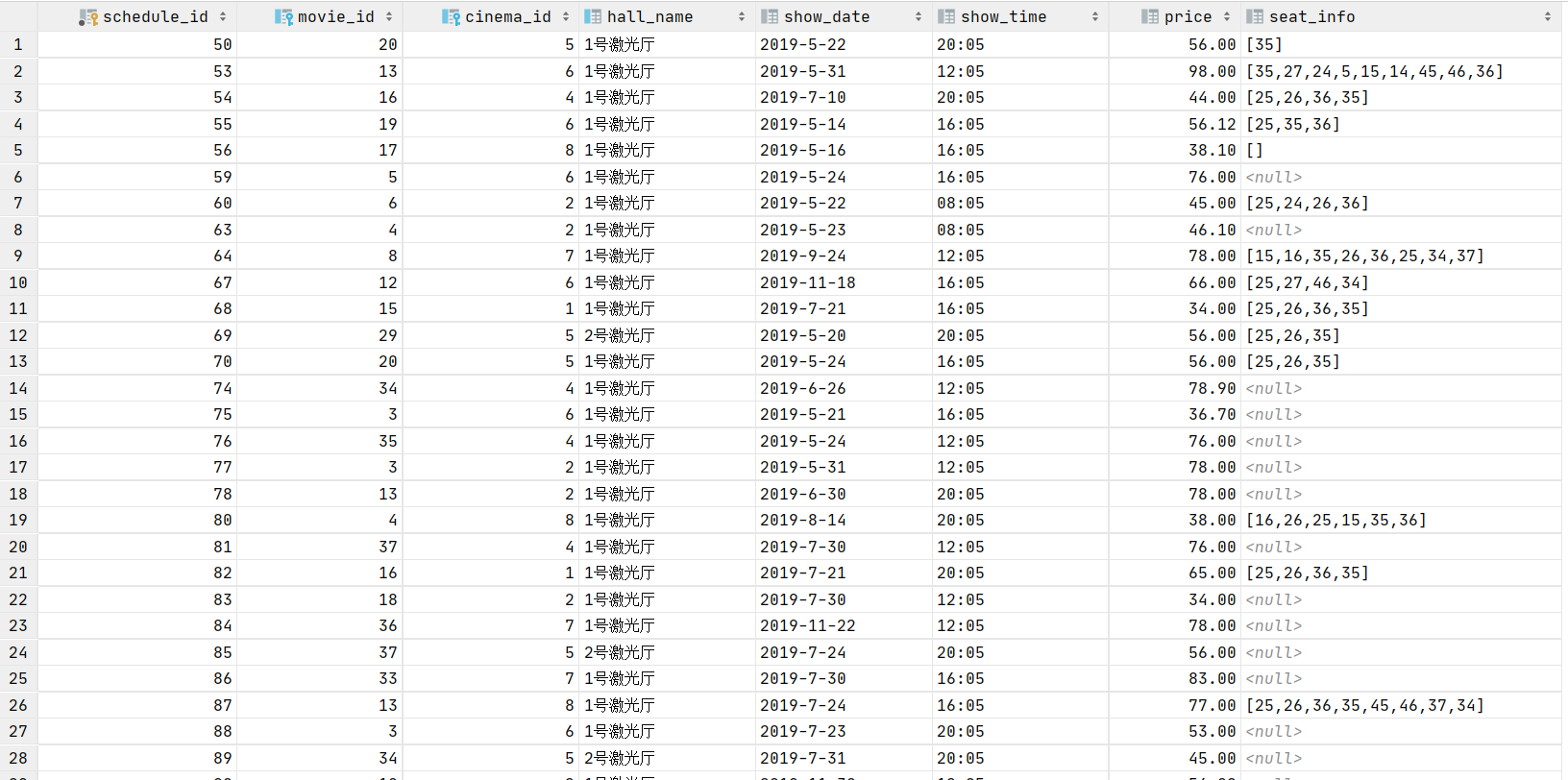


主键：order\_id

外键：user\_id、schedule\_id

|  |  |
| --- | --- |
| 属性 | 功能 |
| order\_phone | 联系电话 |
| order\_date | 订单生成日期 |
| ticket\_num | 订票数 |
| ticket\_total\_price | 订单总价 |
| order\_seat\_info | 订单座位信息(数组) |
| pay\_type | 是否支付 |
| phone\_code |  |

* 排片数据：



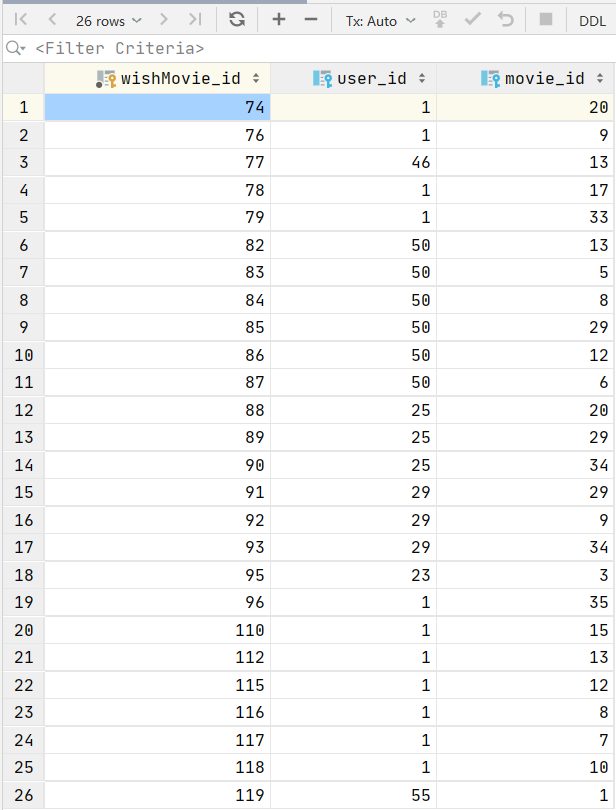
主键：schedule\_id

外键：movie\_id、cinema\_id、hall\_id

|  |  |
| --- | --- |
| 属性 | 功能 |
| show\_date | 放映日期 |
| show\_time | 放映时间 |
| price | 所获票价 |
| seat\_info | 座位信息(数组) |

联系关系表：

|  |  |  |
| --- | --- | --- |
| 联系表 | 唯一标识(主键) | 外键 |
| 想看关系表 | wishMovie\_id | user\_id、movie\_id |



通过mysql对数据进行管理：插入、删除、修改。而数据库铀与我们的代码和界面进行关联，这样就能实现对于我们创建数据的管理。关于相关操作代码在/routes/index.js下，部分信息截图如下：





…

* 异常处理：

我们关于防御性的程序设计为通过预先确认异常（可能导致软件偏离其正常行为的情况）以及包含将系统返回到可接受状态的异常处理。

在我们的项目程序设计中，关于用户交互方面的异常处理主要体现在输入验证(登录)方面，具体解释如下：

验证码库与随机数：



并根据用户登录应有的逻辑编写代码，当用户登录时首先检测验证码是否正确，随后在数据库中调用SELECT函数根据用户名查询用户信息(在用户信息表中存放了用户的登录密码，用户信息表见上的数据管理部分)，当且仅当用户存在且用户输入的密码正确时，用户才可以获取权限登录系统。

核心函数如下：

//密码登录

router.post('/api/pwdLogin',function(req,res){

  let name = req.body.userName;

  let pwd = req.body.password;

  let captcha = req.body.captcha;

  //判断验证码是否正确

  if (captcha.toLowerCase()!==req.session.captcha){

    res.json({error\_code:1,message:'验证码不正确'})

  } else{

    delete req.session.captcha;

    let sqlStr = 'SELECT \* from t\_user WHERE user\_name =? LIMIT 1 ;'

    conn.query(sqlStr,[name],(error,result,field)=>{

      if (error){

        res.json({error\_code:1,message:'查询用户失败'});

      } else{

        result = JSON.parse(JSON.stringify(result));

        if (result[0]){

          if (result[0].password===pwd){

            //保存用户id

            req.session.userId = result[0].user\_id;

            res.cookie('user\_id',result[0].user\_id);

            res.json({success\_code:200})

          } else{

            res.json({error\_code:1,message:'密码错误'});

          }

        } else{

          res.json({error:1,message:'用户不存在'});

        }

      }

    })

  }

});

在我们的项目程序设计中，关于代码层面的异常处理如下：

在javascript中多添加了一个XMLHttpRequest对象。所有的异步交互都是使用XMLHttpRequest对象完成的。

下述代码为封装ajax对象的过程：

1. 判断请求方式(会有相应的不同处理)
2. 返回请求结果，并调用promise的catch函数捕捉error，并进行相应的异常处理。

核心函数如下：

import axios from 'axios'

//封装ajax

export default function ajax(url='',params={},type='GET'){

  let promise;

  return new Promise(((resolve, reject) => {

    //1.判断请求方式

    if ('GET'===type){

      //1.1拼接字符串

      let str = '';

      Object.keys(params).forEach((value,index) => {

        if (index+1===Object.keys(params).length){

          str+=value+'='+params[value];

        }else{

          str+=value+'='+params[value]+'&';

        }

      });

      //1.2完整路径

      url+='?'+str;

      //1.3发送get请求

      promise = axios.get(url);

    }else if('POST'===type){

      //1.3发送post请求

      promise = axios.post(url,params);

    }

    //2.返回请求结果

    promise.then((response)=>{

      resolve(response.data);

    }).catch((error)=>{

      reject(error);

    });

  }))

}

上述显式的异常处理结构有助于将错误检测和恢复与程序的主要功能分离开来。

将安全的代码和不安全的代码分离开，只保护捕捉错误结构中的不安全代码。这种策略不仅可以看请车catch集中哪个部分有潜在的不安全性，而且缩小了预期失效的分为。这样的结构允许再对应的rescue子句中提供更多的恢复方案。此外恢复机制也可能将一个新的一场带入到调用它的代码中，异常有一个很有效的用途，那就是通过弱化模块的前置条件从而增强程序的健壮性。

大体上来讲，异常处理是用来消除易检查的前置条件的最有效的方法。

* UI方面的设计：

进一步研究用户界面的设计时候必须考虑几个问题。

1. 确定谁将与系统进行交互。
2. 开发系统可能执行任务的每种方式的场景。
3. 设计用户命令的层次。
4. 细化用户与系统交互的时序。
5. 设计层次中相关的类，以实现用户界面设计决策。
6. 把用户界面类集成到整个系统的类层次中。

用户界面设计的第一步是以书面的形式展现交互。要做到这一点，可能要确定纸制文档在现有系统中如何流动，这些纸制文档可能会提示在自动化的系统中使用舒适的用户界面。

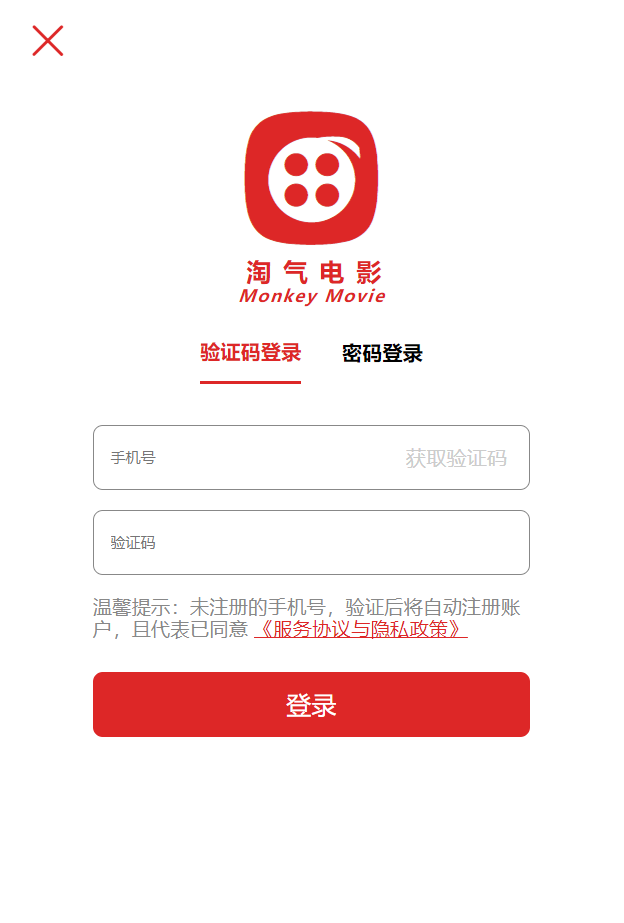
下面给出我们项目的初始用户界面书面形式的设计情况(左部蓝色图)

右侧为代码实现的真实用户界面图

用户界面

书面形式展示交互：

1. 用户登录界面



验证码类型

手机号类型

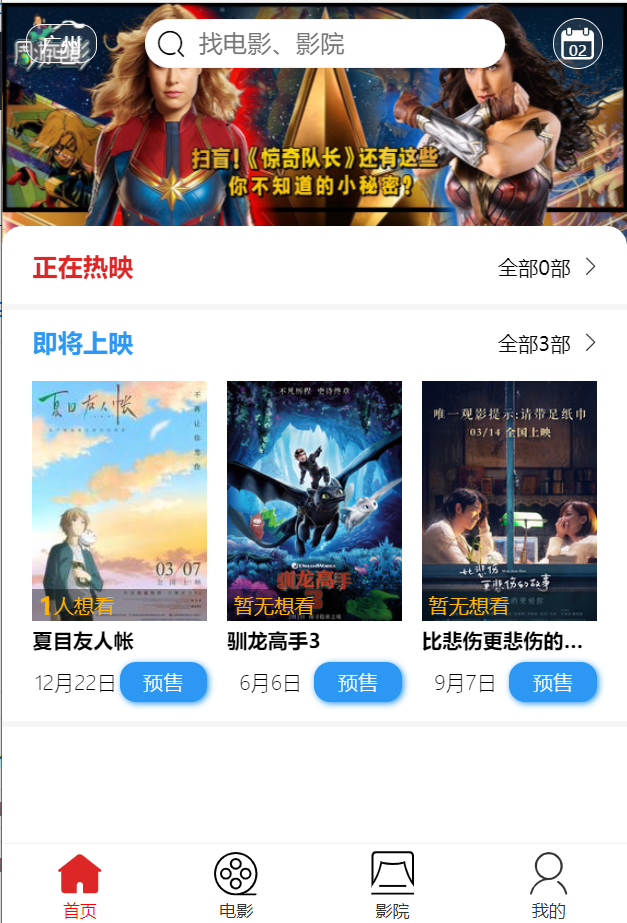
登录

服务协议与隐私政策

验证码登录 密码登录

logo

1. 用户首页界面



电影6

电影5

电影4

电影3

电影2

电影1

即将上映

正在热映

电影推荐页面

日期

地区

搜索框

1. 电影介绍界面



评论

购票

简介

口碑

看过

想看

上映时间类型

电影片长类型

电影主演类型

电影类型类型

电影名称

电影海报

电影名称

系统中交互形式：

管理员界面

1.管理员登陆界面



密码：

用户：

登录

重置

影院售票管理系统

1. 管理员用户管理界面

添加

搜索框

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 用户ID | 用户名 | 性别 | 手机 | 个人签名 | 操作 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

注销

编辑

编辑

注销

编辑

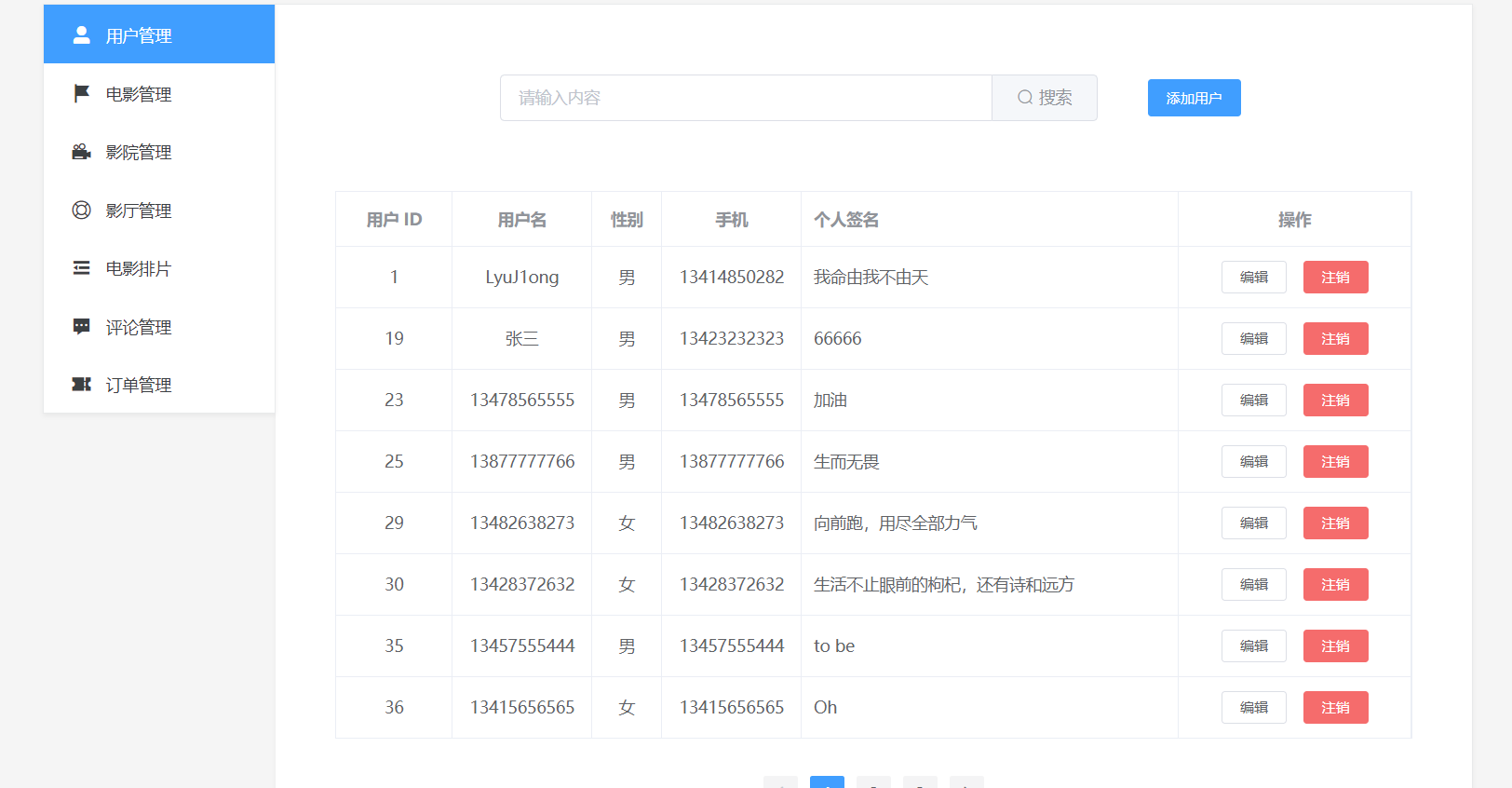
注销

编辑

注销

编辑

注销



4.管理员电影管理界面

添加

搜索框

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 电影ID | 电影名 | 导演 | 类型 | 上映时间 | 操作 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

注销

编辑

编辑

注销

编辑

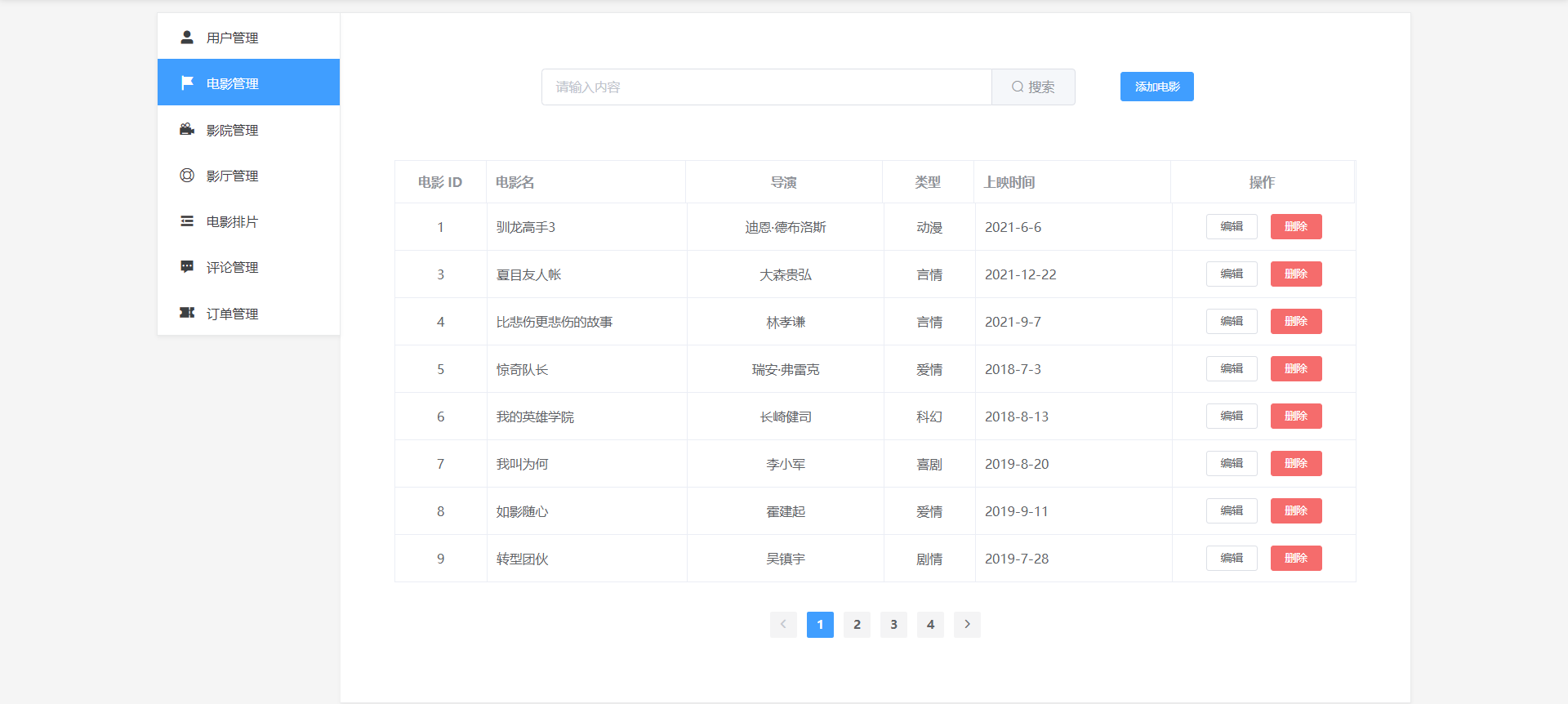
注销

编辑

注销

编辑

注销



5.管理员电影排片界面

添加

搜索框

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 排片ID | 电影名 | 影院名 | 放映大厅 | 放映日期 | 放映时间 | 售价 | 操作 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

注销

编辑

编辑

注销

编辑

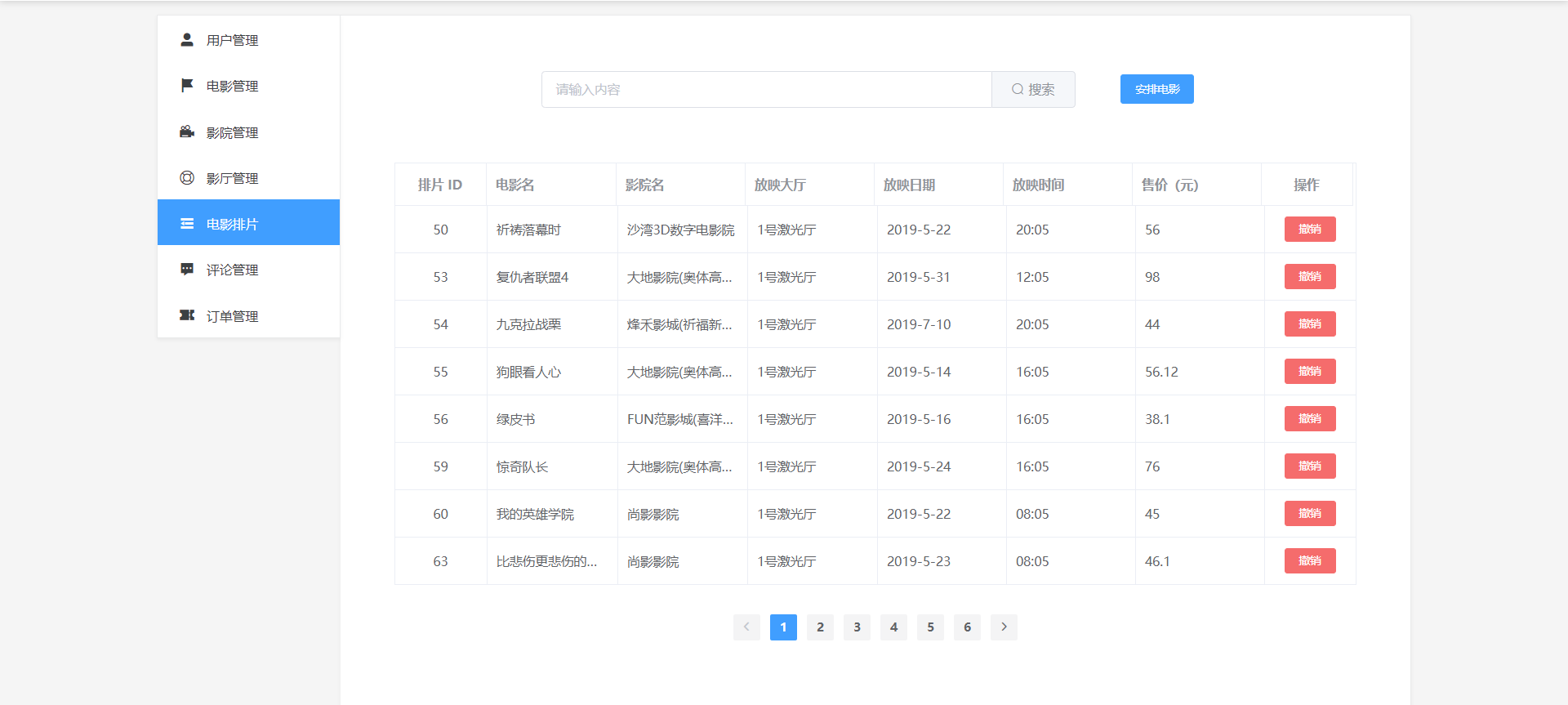
注销

编辑

注销

编辑

注销



6．管理员评价管理界面

添加

搜索框

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 评论ID | 用户名 | 电影名 | 评论时间 | 用户评分 | 评论内容 | 操作 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

注销

编辑

编辑

注销

编辑

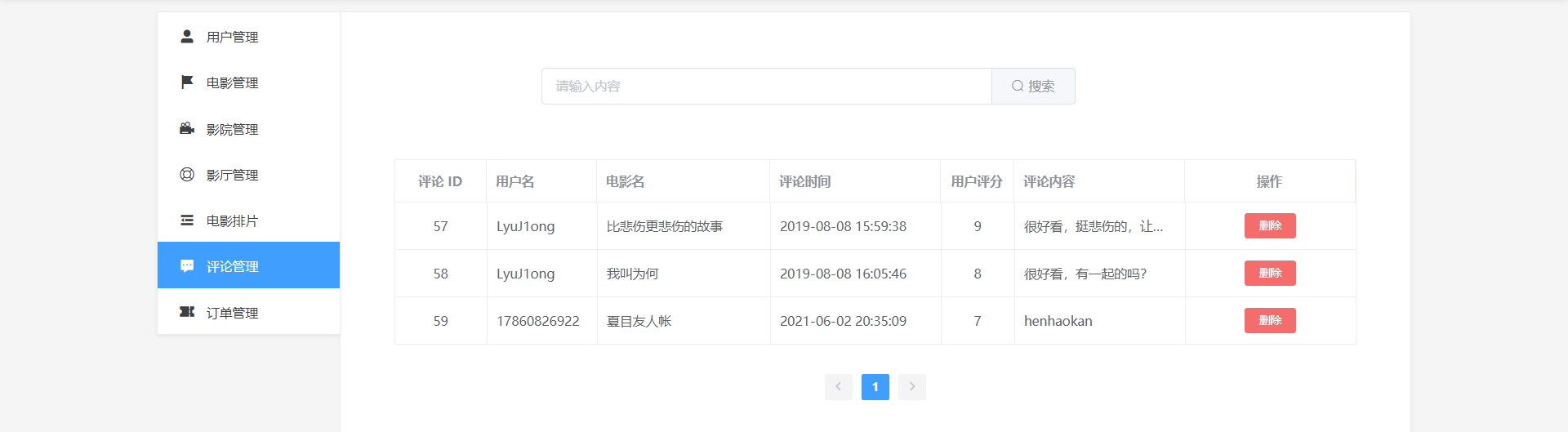
注销

编辑

注销

编辑

注销



7.管理员订单管理界面

添加

搜索框

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 订单ID | 用户名 | 电影名 | 影院名 | 票数 | 总价 | 取票码 | 下单时间 | 操作 |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

注销

编辑

编辑

注销

编辑

注销

编辑

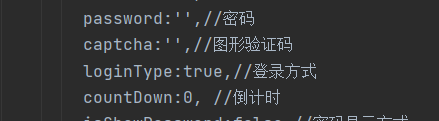
注销

编辑

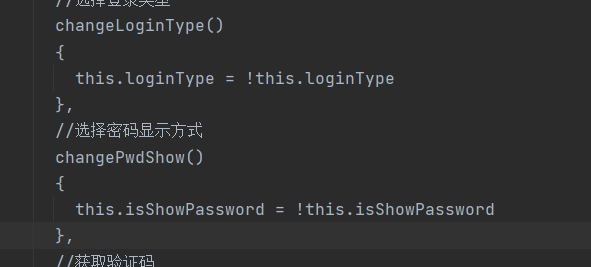
注销



1. 上网查询“阿里 编程规范 （如：阿里巴巴JAVA开发手册）；华为 编程军规”等，对照自己的代码看有哪些不符合规范的地方，修改。



常量命名不规范，countDown应该为COUNT\_DOWN。正确的常量命名格式如下：常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。



大括号使用不规范，，规范的使用方法如下：大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行；如果 是非空代码块则： 1） 左大括号前不换行。 2） 左大括号后换行。 3） 右大括号前换行。 4） 右大括号后还有 else 等代码则不换行；表示终止的右大括号后必须换行。

代码中左括号前换行，属于大括号适应不规范。

缩进格式不规范（同上图）：若使用缩进要求使用四个空格缩进，若使用tab缩进则必须设置tab缩进为四个空格，但代码中并未设置tab为四个空格的缩进。



多个参数传入时，在最后一个参数之前的参数，应该在参数后加空格，但代码中并未加空格。正确的使用规范如下：方法参数在定义和传入时，多个参数逗号后边必须加空格。

此外一些不符合规范的代码与上述所示类型大致相同，故不再赘述。