**摘要：**

电影作为人们生活中不可或缺的一部分，其购票系统的设计和实现至关重要。本论文旨在探索并实现一种全新的电影购票系统，通过结合Vue3、Node.js和Python爬虫技术，旨在为用户和电影院提供更出色的服务和体验，从而推动整个电影产业的发展。

在这个系统中，前端技术Vue3被用于构建用户友好的界面。Vue3具有灵活、高效的特性，能够帮助开发者快速构建交互式的前端页面。通过Vue3，用户可以轻松地浏览电影信息，选择座位，并完成购票流程。优雅的界面设计和流畅的交互体验将大大提升用户的购票体验。

而在后端方面，Node.js被选用用于实现订单管理和数据存储。作为一种轻量级、高效的后端技术，Node.js具有非阻塞I/O、事件驱动等特性，能够处理大量并发请求，并且与前端的数据交互更为便捷。通过Node.js，系统可以实现订单的快速处理和数据的高效存储，保障了系统的稳定性和可靠性。

此外，为了获取电影信息并保持数据的准确性，本系统还集成了Python爬虫技术。借助Python强大的爬虫框架，系统能够自动化地抓取猫眼网站的电影信息，并将其存储到文件中。

本论文将详细介绍系统的设计思路、技术架构以及实际应用效果。通过深入分析系统的优势和特点，展示系统在提升用户体验、优化数据管理以及提高系统性能方面的显著优势。

**关键词：Vue3 Node.js Python爬虫**

**Abstract:**

As an indispensable part of people's lives, the design and implementation of the movie ticketing system are crucial. This paper aims to explore and implement a brand-new movie ticketing system, which combines Vue3, Node.js, and Python web scraping technology, aiming to provide better services and experiences for users and cinemas, thus promoting the development of the entire film industry.

In this system, the frontend technology Vue3 is used to build a user-friendly interface. Vue3, with its flexible and efficient features, can help developers quickly build interactive frontend pages. Through Vue3, users can easily browse movie information, select seats, and complete the ticketing process. Elegant interface design and smooth interactive experience will greatly enhance the user's ticketing experience.

In terms of the backend, Node.js is chosen to implement order management and data storage. As a lightweight and efficient backend technology, Node.js has features such as non-blocking I/O and event-driven, which can handle a large number of concurrent requests and make data interaction with the frontend more convenient. Through Node.js, the system can achieve rapid order processing and efficient data storage, ensuring the stability and reliability of the system.

Furthermore, in order to obtain movie information and maintain data accuracy, this system also integrates Python web scraping technology. With the powerful web scraping framework of Python, the system can automatically fetch movie information from the Maoyan website and store it in files.

This paper will detail the design ideas, technical architecture, and practical application effects of the system. Through in-depth analysis of the advantages and characteristics of the system, it demonstrates significant advantages in improving user experience, optimizing data management, and enhancing system performance.

**Keywords:** **Vue3 Node.js Python web scraping**

# 第1章 绪论

## 1.1 电影产业发展背景

电影作为一种重要的文化娱乐形式，自诞生以来就受到人们的青睐。随着社会经济的不断发展和科技的进步，电影产业也迎来了蓬勃的发展。从最初的黑白无声电影到如今的3D、4D影片，电影技术不断创新，内容不断丰富，观影体验也日益提升。

## 1.2 电影购票系统的重要性

随着互联网的普及和移动设备的普及，线上购票成为了观影的主流方式之一。传统的电影购票方式存在一些不便之处，如排队购票、无法提前选座等问题。因此，开发一种便捷、高效的电影购票系统对于提升用户体验具有重要意义。

## 1.3 研究目的和意义

本论文旨在设计并实现一种基于Vue3、Node.js和Python爬虫的电影购票系统，以提升用户购票体验、优化票务管理，推动电影产业的发展。通过前端技术构建用户友好的界面，后端技术实现订单管理和数据存储，以及爬虫技术获取电影信息，为用户和电影院提供更优质的服务和体验。

# 第2章 系统设计

## 2.1 系统总体架构设计

本系统采用前后端分离的架构，前端使用Vue3框架，后端采用Node.js编写，数据库使用MySQL进行存储。同时，利用Python编写爬虫程序获取电影信息。系统主要包括用户模块、电影信息模块、订单管理模块和爬虫模块。

## 2.2 前端设计与规划

### 2.2.1 Vue3框架介绍

Vue.js 是一款流行的 JavaScript 前端框架，而 Vue 3 则是 Vue.js 的最新版本，它在性能、体积和开发体验等方面都有了一系列的改进和升级。下面是 Vue 3 框架的一些主要特点和优势：

1. **性能优化**： Vue 3 在内部进行了重构和优化，采用了更加高效的虚拟 DOM 实现（即 VNode），并且引入了静态树提升（Static Tree Hoisting）和源码级优化等技术，大幅提升了性能表现，使得页面渲染更加流畅和快速。
2. **Composition API**： Vue 3 引入了全新的 Composition API，这是一个基于函数的 API，使得组件的逻辑可以更加灵活地组织和复用。相比于 Options API，Composition API 更加直观和易于维护，尤其适用于开发复杂的组件和应用。
3. **TypeScript 支持**： Vue 3 对 TypeScript 的支持更加完善，提供了更好的类型推导和类型检查机制，可以帮助开发者在项目开发过程中更早地发现并解决潜在的问题，提高代码的质量和可维护性。
4. **更小的体积**： Vue 3 的体积相比于之前的版本有所减小，同时还可以通过 Tree-shaking 等技术进一步优化打包体积，使得最终的部署文件更加精简，加快页面加载速度。
5. **更好的 TypeScript 支持**： Vue 3 提供了更好的 TypeScript 支持，包括更强大的类型推断、类型声明文件等，使得在使用 TypeScript 进行开发时更加流畅和便捷。
6. **更好的适配性**： Vue 3 在 SSR（服务器端渲染）和微前端等方面有了更好的支持和适配性，可以更加灵活地应用于各种类型的项目和场景中。

### 2.2.2 组件设计

单文件组件（SFC）是 Vue.js 中一种组织代码的方式，将模板、样式和逻辑代码组织在一个文件中。它具有以下特点和好处：

1. **清晰的结构**： 单文件组件将一个组件的相关内容放置在一个文件中，包括模板、样式和逻辑代码，使得组件的结构更加清晰明了。开发者可以一目了然地了解组件的构成和功能。
2. **提高可维护性**： 单文件组件将相关的代码放置在同一个文件中，使得组件的代码更易于维护。开发者可以更快速地找到和修改组件的相关代码，而不用在多个文件之间来回切换。
3. **提高复用性**： 单文件组件可以被轻松地复用在不同的页面或项目中。通过导入单文件组件，并在需要的地方进行注册和使用，可以快速构建出复杂的页面和应用。
4. **作用域化样式**： 单文件组件支持在 <style> 标签中编写样式，并且默认情况下使用了 CSS 模块化，样式只在当前组件中生效，不会影响到其他组件。这样可以有效地避免样式污染和命名冲突。
5. **代码分割和懒加载**： 单文件组件支持异步组件加载，可以根据需要动态加载组件，实现代码分割和懒加载，提高页面的加载性能和用户体验。
6. **单文件调试**： 单文件组件可以在一个文件中集中管理组件的所有内容，包括模板、样式和逻辑代码，使得调试更加方便。开发者可以在一个文件中查看和修改组件的所有相关内容，而不用在多个文件之间跳转。
7. **与构建工具的集成**： 单文件组件可以与现代的构建工具（如Webpack、Rollup等）进行集成，通过预处理器（如Babel、TypeScript等）进行编译，实现更高级的功能和优化，如代码压缩、打包优化等。

为了提高代码的复用性和可维护性，本系统将模板、脚本和样式组织在同一个文件中，使得组件的相关代码更加紧凑和一目了然，方便进行维护和调试。同时，单文件组件也支持通过 webpack 等构建工具进行预处理和打包，实现代码分割、懒加载等高级功能，提高项目的性能和可维护性。

## 2.3 后端设计与规划

### 2.3.1 Node.js框架介绍

Node.js是一个基于Chrome V8引擎的JavaScript运行时环境，可以用来快速构建高性能的网络应用程序。它采用事件驱动、非阻塞I/O模型，使得在处理大量并发连接时能够保持高效率。以下是一些常见的 Node.js 框架：

1. **Express.js**： Express.js 是最流行的 Node.js Web 框架之一，它简洁而灵活，适用于构建各种类型的 Web 应用程序，从简单的 API 服务到复杂的 Web 应用都能够得到很好的支持。Express.js 的设计理念是提供最少量的功能，让开发者自由选择和扩展所需的功能。
2. **Koa.js**： Koa.js 是一个新兴的 Node.js Web 框架，由 Express.js 的原作者设计开发，旨在提供更加简洁、灵活的开发体验。Koa.js 基于 async/await 特性，使得异步流程控制更加清晰和直观，同时采用了基于中间件的架构，使得应用程序的代码更加模块化、可扩展。本系统便是采用了该框架。
3. **Nest.js**： Nest.js 是一个基于 TypeScript 和 Express.js 的框架，它提供了一种面向对象（OOP）的编程方式，并且借鉴了 Angular 框架的一些设计理念，使得开发者可以更加轻松地构建可维护、可扩展的 Node.js 应用程序。
4. **Meteor.js**： Meteor.js 是一个全栈 JavaScript 框架，它包含了前端、后端和数据库等方面的功能，提供了一套完整的开发工具和库，使得开发者可以快速构建实时 Web 应用程序。
5. **Socket.io**： Socket.io 是一个用于构建实时 Web 应用程序的库，它基于 WebSocket 技术，并提供了跨浏览器的实时通信解决方案，使得开发者可以轻松实现实时聊天、实时数据更新等功能。

### 2.3.2 RESTful API设计

设计 RESTful API 是构建现代 Web 应用程序时非常重要的一步，它可以使得不同的客户端（如 Web 应用、移动应用、IoT 设备等）能够与服务器进行有效的通信。下面是一些设计 RESTful API 的常见原则和最佳实践：

1. **资源（Resources）**： RESTful API 的核心概念是资源，每个资源都应该有一个唯一的标识符（URI），并且可以通过 HTTP 方法（GET、POST、PUT、DELETE 等）对资源进行 CRUD（创建、读取、更新、删除）操作。
2. **HTTP 方法的正确使用**： 使用 HTTP 方法来表示对资源的操作，GET 方法用于获取资源，POST 方法用于创建资源，PUT 或 PATCH 方法用于更新资源，DELETE 方法用于删除资源。
3. **清晰的资源命名**： 资源的命名应该清晰、简洁，采用名词复数形式，并且遵循 RESTful API 的命名规范，例如 /users 表示用户资源，/posts 表示文章资源等。
4. **使用状态码（Status Codes）**： 使用合适的 HTTP 状态码来表示 API 请求的结果，如 200 表示成功，201 表示资源创建成功，404 表示资源未找到，500 表示服务器内部错误等。
5. **版本控制（Versioning）**： 在设计 API 时应该考虑到未来的扩展和变化，可以通过在 URI 中添加版本号或者使用 HTTP 头部来进行版本控制，以保证向后兼容性。
6. **身份验证和授权（Authentication and Authorization）**： 对于需要进行身份验证和授权的 API，应该使用合适的身份验证机制（如 OAuth）和授权方案（如 JWT）来保护 API 的安全性。
7. **错误处理（Error Handling）**： 对于 API 请求中可能出现的错误情况，应该提供清晰的错误信息和合适的错误码，以便客户端能够正确处理和反馈错误。
8. **文档和描述（Documentation and Description）**： 提供清晰、详细的 API 文档和描述，包括每个接口的用法、参数、返回值等信息，以便开发者能够快速了解和使用 API。

本系统设计了一组RESTful风格的API接口，用于处理前端发送过来的请求，并与数据库进行交互。每个接口都有对应的URL和HTTP方法，实现了对资源的增删改查操作。

### 2.3.3 数据库设计

MySQL 是一种流行的开源关系型数据库管理系统（RDBMS），由瑞典 MySQL AB 公司开发，后被 Sun Microsystems 收购，现在隶属于 Oracle 公司。MySQL 使用标准 SQL 语言进行数据库管理，支持多种操作系统，包括 Linux、Windows、macOS 等，是许多 Web 应用程序和网站的首选数据库系统之一。其特点包括：

1. **开源免费：** MySQL 是开源软件，可以免费获取和使用。这使得它成为许多个人和组织的首选数据库系统，尤其适用于中小型企业和个人开发者。
2. **跨平台支持：** MySQL 可以运行在多种操作系统上，包括 Linux、Windows、macOS 等，使得开发者可以在不同的平台上开发和部署应用程序。
3. **高性能：** MySQL 是一个高性能的数据库系统，具有快速的数据处理能力和高效的查询性能。它支持多种存储引擎（如 InnoDB、MyISAM 等），可以根据实际需求选择最适合的存储引擎。
4. **可扩展性：** MySQL 支持主从复制、分区、集群等技术，可以实现数据库的水平和垂直扩展，以满足不同规模和需求的应用程序。
5. **稳定可靠：** MySQL 经过长期的发展和测试，具有稳定可靠的性能和质量保证。它被广泛应用于各种关键业务系统和大型网站，如 Facebook、Twitter 等。
6. **易于使用：** MySQL 提供了直观的管理工具和丰富的文档资源，使得用户可以轻松地安装、配置和管理数据库系统，并进行数据的导入、导出和备份等操作。
7. **社区支持：** MySQL 拥有庞大的开发者社区和用户群体，用户可以通过官方文档、论坛、邮件列表等渠道获取支持和帮助，解决在使用过程中遇到的各种问题。

本系统选择了MySQL作为数据库存储电影信息、用户信息和订单信息等数据。通过设计合理的数据库表结构和建立索引，提高了数据的查询效率和存储空间利用率。

## 2.4 爬虫设计与规划

### 2.4.1 Python爬虫技术简介

Python爬虫技术是指利用Python编程语言编写的一种网络爬虫程序，用于从互联网上获取并提取特定网站或网页上的信息。爬虫技术通常用于数据采集、信息抓取、搜索引擎建立等应用场景，可以自动化地访问网页、解析页面内容，并将感兴趣的数据提取出来进行处理或存储。其特点包括：

1. **简单易用：** Python 语法简洁清晰，易于学习和使用，使得编写爬虫程序变得简单和快速。
2. **丰富的库支持：** Python 生态系统中有许多成熟的网络爬虫库和相关工具，如 requests、BeautifulSoup、Scrapy 等，提供了丰富的功能和强大的功能扩展性。
3. **灵活性和可扩展性：** Python 爬虫程序可以根据实际需求进行定制和扩展，支持各种网站结构和数据格式的处理。
4. **跨平台性：** Python 可以在多种操作系统上运行，包括 Windows、Linux、macOS 等，使得爬虫程序具有良好的跨平台性。
5. **社区支持和资源丰富：** Python 拥有庞大的开发者社区和丰富的学习资源，用户可以通过官方文档、在线教程、社区论坛等获取支持和帮助。

本系统利用Python编写爬虫程序，从电影网站上抓取电影信息、影院信息和排片信息。

### 2.4.2 爬虫程序设计思路

在本系统中，我最终选择爬取猫眼电影网站的数据。爬虫程序通过模拟浏览器行为，访问网页、解析网页内容，获取所需的电影信息，并将其存储到本地文件中。大致思路如下：

1. **发送请求获取网页源代码：** 程序首先构造了一个HTTP请求，模拟浏览器行为，并发送给目标网站。请求头中包含了用户代理信息，使得请求看起来像是来自于一个正常的浏览器。然后程序尝试从响应中获取网页源代码。
2. **解析网页源代码：** 通过使用正则表达式，程序解析了网页源代码，以提取出猫眼电影中每部电影的相关信息，比如排名、图片链接、电影名称、主演、上映时间和评分等。解析过程中，程序使用了正则表达式的匹配功能来定位和提取目标内容。
3. **写入文件：** 最后，程序将解析得到的电影信息以JSON格式写入到名为result.txt的文件中。每部电影信息被写入为一个JSON对象，并以逗号分隔，保证了每次运行程序都能将新获取的数据追加到文件末尾。
4. **主程序调度：** 主程序负责循环调用main()函数，每次传入不同的偏移量offset，以获取不同页面的数据。为了防止请求过于频繁被目标网站封禁，程序在每次请求后都会暂停一段时间（这里设置为3秒）。

# 第3章 技术实现

## 3.1 前端技术实现细节

### 3.1.1 Vue Router的路由管理

Vue Router 是 Vue.js 官方的路由管理器，用于构建单页面应用程序（SPA）。它可以帮助我们在 Vue.js 应用中实现页面之间的导航和跳转，并且支持路由参数、嵌套路由、路由拦截等功能，是开发 Vue.js 应用不可或缺的一部分。我根据系统的页面结构，配置了不同的路由规则，实现了页面之间的跳转和导航。代码见图1、图2：

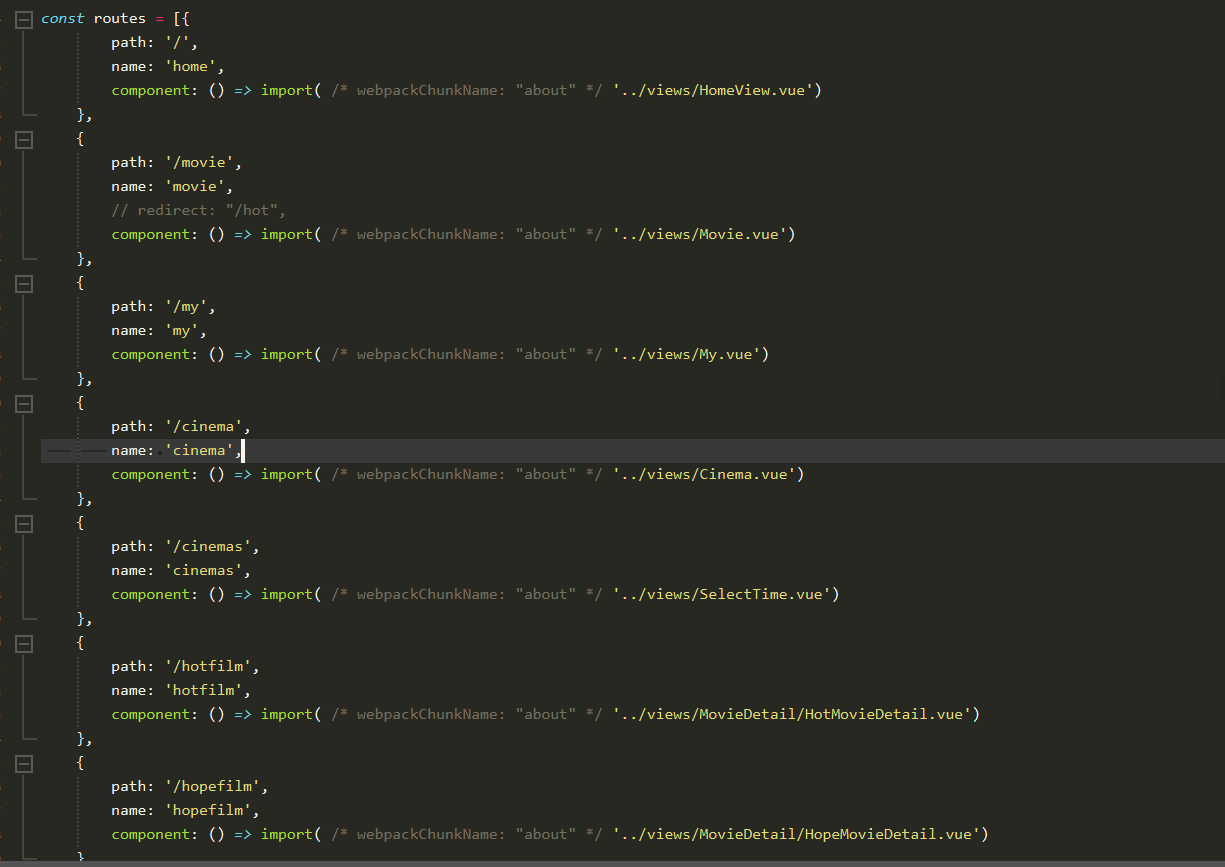


图1



图2

此外，还可以通过路由导航守卫来实现路由拦截和权限控制。Vue Router 提供了 beforeEach、beforeResolve、afterEach 等导航守卫钩子函数，可以在路由跳转前、跳转后进行相应的逻辑处理。本系统使用了beforeEach来判断用户是否已登陆，代码见图3：

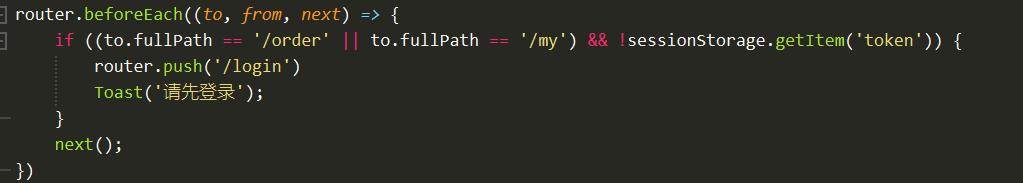


图3

### 3.1.2 Element UI组件的应用

Element UI是一套基于Vue.js的UI组件库，提供了丰富的基础组件和高质量的Vue组件，我在系统中广泛应用了Element UI的各种组件，如按钮、表格、表单等，以及自定义主题和样式，使得界面美观大方，用户操作更加友好。

### 3.1.3 Vant组件库的应用

Vant是一个基于Vue.js的轻量、可靠的移动端组件库，提供了丰富的UI组件和功能组件，我在系统中广泛应用了Vant的组件，以提升用户的交互体验和界面美观度。本系统中的按钮、图片懒加载(Lazyload)、电影的幻灯片展示等等都是基于Vant组件库来实现的。

### 3.1.4 全屏滚动效果的实现

vue-fullpage.js 是一个基于 Vue.js 的全屏滚动组件，它能够帮助开发者在网页中实现全屏滚动的效果。通常情况下，网页的内容是按照页面的高度逐段显示的，而全屏滚动则是将整个页面的内容划分为多个屏，用户可以通过滚动页面或者点击导航按钮来切换不同的屏幕内容，从而呈现出一种流畅而独特的页面浏览体验。

代码见图4、图5：

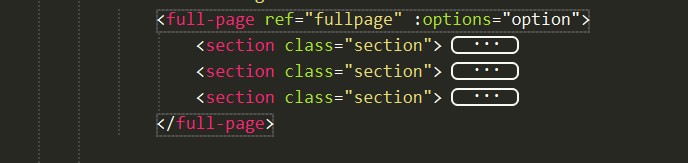


图4

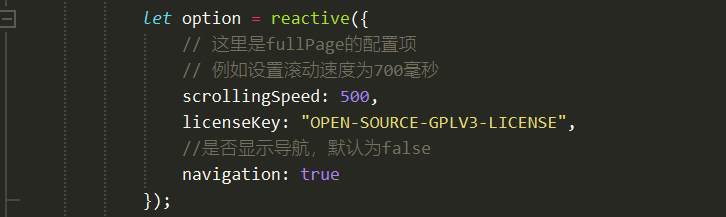


图5

### 3.1.5 实现分页功能的辅助函数

在现代的网络应用中，数据量庞大的情况下，经常需要对数据进行分页展示，以提高用户体验。为了简化开发过程，我们可以编写一些辅助函数来实现分页功能。

因此，我使用 Vue 3 提供的 ref 和 computed API 编写了一个分页辅助函数，该函数能够根据传入的数据和每页显示的数据条数来进行分页，并提供了跳转到下一页、上一页的功能。代码见图6：



图6

### 3.1.6 使用 Axios 进行数据交互

在现代前端开发中，与后端服务的数据交互是不可或缺的一部分。本项目采用了 Axios，一个基于 promise 的 HTTP 客户端，来处理与后端的 API 请求。Axios 提供了简洁的 API，并且可以在浏览器和 node.js 中使用。

1. **配置 Axios 实例**

为了简化请求过程和统一管理 API 请求的配置，我创建了一个自定义的 Axios 实例。以下是 Axios 实例的配置代码：

import axios from 'axios';

const http = axios.create({

baseURL: 'http://localhost:3000/api', *// 生产环境的基础URL*

timeout: 5000, *// 请求超时时间*

});

此配置设置了请求的基本 URL 和超时限制。baseURL属性使得每个请求都以此 URL 为基础，减少了重复代码的书写。超时设置确保了在指定时间内没有响应会自动终止请求，增强了用户体验。

1. **API 请求封装**

为了保持代码的整洁性和重用性，我们将每个具体的API请求封装成函数，并统一管理。以下是几个 API 请求函数的示例：

export const getTopMovielist = () => {

return http({

url: '/top',

method: 'get'

});

}

export const Login = (data) => {

return http({

url: '/login',

method: 'post',

data: data,

headers: {

'Content-Type': 'application/json',

}

});

}

这些函数分别处理不同的数据请求，如获取电影列表、用户登录等。通过封装，我们能够在不同的组件中轻松复用这些函数，同时保持代码的干净和易维护。

1. **异常处理**

在进行网络请求时，异常处理是不可避免的。Axios通过返回的promise提供了捕获错误的机制。每个请求函数可以通过.catch方法来处理可能发生的错误，例如：

getTopMovielist()

.then(response => {

console.log(response.data);

})

.catch(error => {

console.error('Error fetching top movie list:', error);

});

在这里，如果getTopMovielist请求失败，错误将被捕获并打印出来，便于开发者调试并向用户提供错误信息。

## 3.2 后端技术实现细节

### 3.2.1 Koa框架

Koa 是一个基于 Node.js 平台的轻量级 Web 框架，它由 Express 框架的原作者设计开发，旨在提供更加简洁、灵活的 Web 应用开发体验。Koa 的核心思想是利用 ES6 的新特性，尤其是 async/await，来优雅地处理异步代码，同时通过中间件的机制来实现更加模块化、可扩展的应用程序。

下面是 Koa 框架的一些主要特点和优势：

1. **轻量级和简洁：** Koa 框架的代码量相比于其他框架更少，核心功能精简，代码清晰易懂，使得开发者可以更加专注于业务逻辑的实现。
2. **基于中间件的架构：** Koa 框架采用了基于中间件的架构，开发者可以通过组合不同的中间件来完成各种功能，使得应用程序的代码更加模块化、易于维护。
3. **异步流程控制：** Koa 框架利用 async/await 特性来处理异步代码，消除了回调地狱（Callback Hell）的问题，使得异步流程控制更加清晰和直观。
4. **优秀的错误处理机制：** Koa 框架内置了对错误处理的支持，可以通过 try/catch 或者错误中间件来捕获和处理错误，保证应用程序的稳定性和可靠性。
5. **灵活的扩展性：** Koa 框架提供了丰富的插件和中间件库，开发者可以根据项目的需求选择合适的插件进行集成，实现自定义的功能和特性。
6. **面向未来的设计：** Koa 框架采用了 ECMAScript 6（ES6）的语法特性，如箭头函数、解构赋值等，使得代码更加现代化、可读性更高。

### 3.2.2 数据库连接与操作

我使用了Node.js的MySQL模块来连接MySQL数据库，并利用SQL语句进行数据的增删改查操作。通过封装数据库操作函数，简化了代码逻辑，提高了代码的可读性和可维护性。

数据库名称为cinema，本系统一共设计了7个数据库表，分别如下：

用户表：记录用户的信息。

热门电影表：记录热门电影的信息。

最受期待电影表：记录最受期待电影的信息。

评分前100电影表：记录评分前100电影的信息。

电影院表：记录电影院的信息。

电影院排片表：记录电影院排片的信息。

用户订单表：记录用户订单的信息。

1. 用户表的字段说明如表1所示，有3个字段。

表1

|  |  |  |
| --- | --- | --- |
| **字段名称`** | **数据类型** | **说明** |
| user\_id | int | 用户id（主键） |
| username | varchar | 用户账号 |
| password | varchar | 用户密码 |

1. 热门电影表的字段说明如表2所示，有8个字段。

表2

|  |  |  |
| --- | --- | --- |
| **字段名称** | **数据类型** | **说明** |
| movie\_id | int | 电影id（主键） |
| image | varchar | 电影封面 |
| big\_image | varchar | 电影大封面 |
| title | varchar | 电影名字 |
| actor | varchar | 演员 |
| time | varchar | 上映时间 |
| score | varchar | 电影评分 |
| detail | varchar | 电影简介 |

1. 最受期待电影表的字段说明如表3所示，有7个字段。

表3

|  |  |  |
| --- | --- | --- |
| **字段名称** | **数据类型** | **说明** |
| movie\_id | int | 电影id（主键） |
| image | varchar | 电影封面 |
| big\_image | varchar | 电影大封面 |
| title | varchar | 电影名字 |
| actor | varchar | 演员 |
| time | varchar | 上映时间 |
| detail | varchar | 电影简介 |

1. 评分前100电影表的字段说明如表4所示，有8个字段。

表4

|  |  |  |
| --- | --- | --- |
| **字段名称** | **数据类型** | **说明** |
| movie\_id | int | 电影id（主键） |
| image | varchar | 电影封面 |
| big\_image | varchar | 电影大封面 |
| title | varchar | 电影名字 |
| actor | varchar | 演员 |
| time | varchar | 上映时间 |
| score | varchar | 电影评分 |
| detail | varchar | 电影简介 |

1. 电影院表的字段说明如表5所示，有7个字段。

表5

|  |  |  |
| --- | --- | --- |
| **字段名称** | **数据类型** | **说明** |
| cinema\_id | int | 电影院id（主键） |
| cinema\_name | varchar | 影院名称 |
| cinema\_phone | varchar | 影院电话 |
| province | varchar | 影院所在省份 |
| city | varchar | 影院所在市 |
| country | varchar | 影院所在区县 |
| specified\_address | varchar | 影院详细地址 |

1. 电影院排片表的字段说明如表6所示，有10个字段。

表6

|  |  |  |
| --- | --- | --- |
| **字段名称** | **数据类型** | **说明** |
| schedule\_id | int | 排片id（主键） |
| movie\_id | int | 电影id |
| cinema\_id | int | 电影院id |
| hall\_name | varchar | 几号激光厅 |
| show\_date | varchar | 排片时间 |
| language | varchar | 语言 |
| beginTime | varchar | 电影开场时间 |
| endTime | varchar | 电影散场时间 |
| price | decimal(10,2) | 售价 |
| seat\_info | text | 已购座位 |

1. 用户订单表的字段说明如表7所示，有10个字段。

表7

|  |  |  |
| --- | --- | --- |
| **字段名称** | **数据类型** | **说明** |
| order\_id | int | 订单id（主键） |
| userid | int | 用户id |
| image | varchar | 电影封面 |
| movie\_name | varchar | 电影名字 |
| cinema\_name | varchar | 电影院名字 |
| hall\_name | varchar | 几号激光厅 |
| seats | text | 购票座位 |
| date\_time | varchar | 上映时间 |
| begin\_time | varchar | 电影开场时间 |
| price | decimal(10,2) | 售价 |

### 3.2.3 用户认证与权限管理

jsonwebtoken是 Node.js 中用于生成和验证 JSON Web Tokens（JWT）的库。JWT 是一种基于 JSON 的开放标准（RFC 7519），用于在网络上安全地传输信息。它由三部分组成：头部（Header）、载荷（Payload）和签名（Signature）。

为了保护用户的隐私和数据安全，本系统借助jsonwebtoken库实现了用户认证和权限管理功能。用户在登录时，系统会验证用户的身份信息，并颁发一个JWT令牌作为用户的身份凭证，用户在进行相关操作时需要携带有效的令牌进行身份验证。代码见图7：

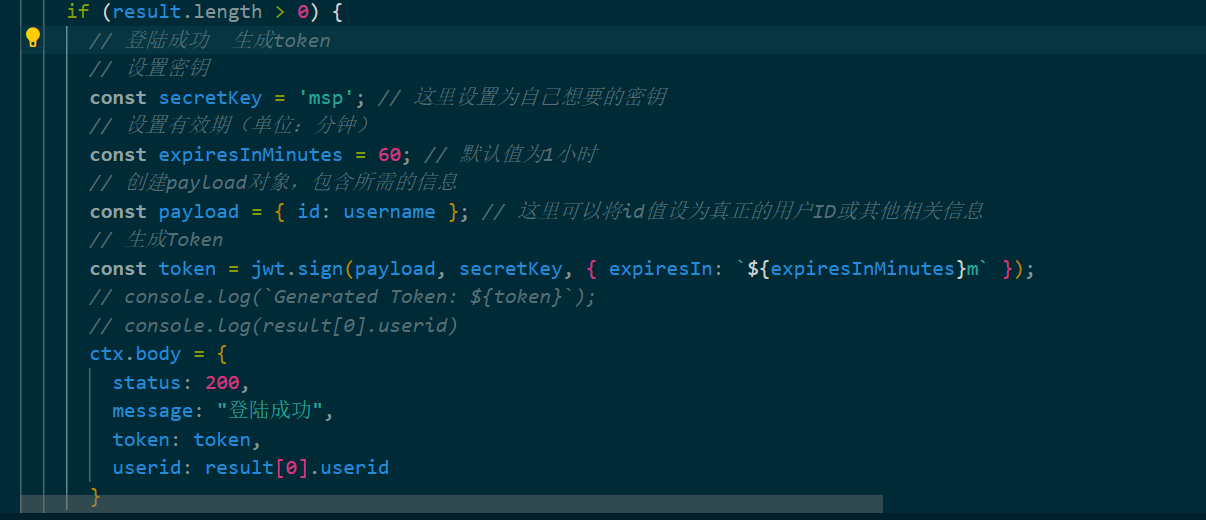


图7

## 3.3 爬虫技术实现细节

### 3.3.1 网页抓取与解析

爬虫的核心功能之一是从网页中获取所需数据。在本项目中，我们可以使用 Python 的 requests 库来抓取网页源代码，并利用正则表达式对源代码进行解析，提取出我们感兴趣的信息。

操作步骤如下：

1. **网页抓取**： 在爬虫代码中，首先定义了 get\_one\_page(url) 函数，该函数使用了 requests 库发送 HTTP 请求，并设置了合适的请求头，以模拟浏览器访问。这样可以避免被网站识别为爬虫程序而触发反爬虫机制。
2. **网页解析**： 接着，使用正则表达式对网页源代码进行解析，提取出目标数据。在 parse\_one\_page(html) 函数中，定义了正则表达式模式来匹配电影的排名、图片、标题、演员、上映时间和评分等信息。通过正则表达式的 findall 方法，即可获取所有匹配的结果，并逐个生成字典形式的电影信息。

### 3.3.2 数据存储与更新

爬取到的数据需要进行存储和更新，以便后续的分析和使用。在本项目中，我们可以选择将数据以 JSON 格式写入到本地文件中，以便后续处理。

操作步骤如下：

1. **数据写入文件**： 在 write\_to\_file(content) 函数中，我使用了 Python 的内置 json 库将字典形式的电影信息转换成 JSON 格式，并以 UTF-8 编码写入到名为 result.txt 的文本文件中。这样做的好处是数据格式清晰明了，易于读取和处理。
2. **数据更新策略**： 在实际应用中，我们可能需要定期更新已有数据或者增量式地添加新数据。这可以通过设置定时任务，定期运行爬虫程序，并将新获取的数据追加到已有文件中来实现。另外，可以通过比较已有数据和新获取的数据，更新已有数据或者标记出新增的数据。

完整实现代码如下：

import json  
import re  
import time  
  
import requests  
from requests import RequestException  
  
  
# 爬取top100  
  
def get\_one\_page(url):  
 try:  
 headers = {  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36',  
 }  
 response = requests.get(url, headers=headers) # 获取网页源代码  
 if response.status\_code == 200:  
 return response.text # 成功响应 返回源代码  
 return None # 失败 返回空值  
 except RequestException:  
 return None # 异常值处理  
  
  
def parse\_one\_page(html):  
 pattern = re.compile(  
 '<dd>.\*?board-index.\*?>(.\*?)</i>.\*?data-src="(.\*?)".\*?name.\*?a.\*?>(.\*?)</a>.\*?star.\*?>(.\*?)</p>.\*?releasetime.\*?>(.\*?)</p>.\*?integer.\*?>(.\*?)</i>.\*?fraction.\*?>(.\*?)</i>.\*?</dd>',  
 re.S) # 写一个正则提取的方法  
 items = re.findall(pattern, html) # 找出所有符合条件的内容 此时是一个列表形式  
 for item in items: # 遍历提取结果并生成字典  
 yield { # 迭代器  
 'index': item[0],  
 'image': item[1],  
 'title': item[2].strip(),  
 'actor': item[3].strip()[3:] if len(item[3]) > 3 else '',  
 'time': item[4].strip()[5:] if len(item[4]) > 5 else '',  
 'score': item[5].strip() + item[6].strip()  
 }  
  
  
def write\_to\_file(content): # 写入文件  
 with open('result.txt', 'a', encoding='utf-8') as f:  
 print(type(json.dumps(content)))  
 f.write(json.dumps(content, ensure\_ascii=False) + ',' + '\n')  
  
  
def main(offset):  
 url = 'http://maoyan.com/board/4?offset=' + str(offset) # 接收offset参数  
 html = get\_one\_page(url) # 赋值源代码给html  
 # print(html)  
 for item in parse\_one\_page(html): # 遍历字典  
 print(item)  
 write\_to\_file(item) # 写入文件  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 for i in range(10):  
 print("爬取第%d页。。。。。。。。" % (i + 1))  
 main(offset=i \* 10) # 传入offset参数  
 time.sleep(3) # 延时等待 速度过快由于反爬虫会无响应

# 第4章 系统测试与优化

## 4.1 单元测试与集成测试

为了确保系统的稳定性和可靠性，我进行了单元测试和集成测试，并通过手动测试来验证系统在各种情况下的表现。

在单元测试方面，对系统的各个模块进行了详尽的测试。通过手动点击各个不同的区域以及输入各种不同的数据和参数，检查了每个模块的功能是否正常，以及是否能够正确处理各种异常情况。这些手动测试覆盖了各种可能出现的情况，确保了系统各个模块的稳定性和正确性。

在集成测试阶段，我将各个模块组合起来，模拟了真实环境下的用户操作流程。通过手动操作系统的不同功能，验证了各个模块之间的协同工作是否流畅，以及系统在整体上的稳定性和一致性。这些手动集成测试帮助我发现了一些潜在的问题，并及时进行了调整和修复。

虽然我没有编写代码进行自动化测试，但通过手动测试，仍然能够全面地验证系统的功能和性能，确保系统能够满足用户的需求。

## 4.2 性能优化与安全加固

为了提升系统的性能和安全性，我采取了一系列措施进行优化和加固。具体包括：

1. **前端代码的压缩和合并：** 对前端代码进行压缩和合并，以减少页面加载时间，提升用户体验。
2. **后端代码的优化和缓存：** 通过对后端代码逻辑和数据库查询进行优化，降低了系统的响应时间，并引入了缓存机制，加速了数据的读取和存储。
3. **数据库的索引和分区：** 对数据库进行索引优化和分区管理，以提升数据的检索速度和存储效率，确保系统的稳定性和可靠性。
4. **用户身份认证和数据传输加密：** 加强用户身份认证机制，确保用户身份的合法性和安全性。同时，对数据传输进行了加密处理，以防止敏感信息在传输过程中被窃取和篡改，保障用户的隐私和数据安全。

# 结 论

通过本论文的研究和实践，我成功设计并实现了一种基于Vue3、Node.js和Python爬虫技术的电影购票系统，旨在提升用户购票体验、优化票务管理，以及推动电影产业的发展。通过前端技术Vue3的应用，我构建了一个用户友好的界面，使用户能够轻松浏览电影信息、选择座位并完成购票流程。与此同时，后端技术Node.js的运用实现了订单管理和数据存储，保障了系统的稳定性和可靠性。然而，尽管我尝试集成Python爬虫技术以实现数据的自动抓取和更新，但在手动调试过程中，我发现由于网站结构的变化和反爬虫机制的加强，Python爬虫并不能保证数据的及时更新，导致系统无法实现电影信息的自动更新。

在手动调试的过程中，我验证了系统的可行性和效果。我模拟了用户的操作流程，手动输入数据并测试了系统的各项功能。尽管未投入实际应用，但我对系统的设计和实现进行了全面的测试和验证。

总体而言，这个系统在提升用户体验、优化数据管理和推动电影产业发展方面具有潜力。我相信，随着技术的不断发展和完善，电影购票系统将在未来发挥更加重要的作用，为电影产业的健康发展做出更大的贡献。

# 参考文献

[1]崔庆才.Python3网络爬虫开发实战[M].人民邮电出版社,2021

[2]胡松涛.Python网络爬虫实战[M].清华大学出版社,2016

[3]李俊、罗勇胜.数据库原理与应用快速入门[M].西安电子科技大学出版社，2016

[4]霍春阳.Vue.js设计与实现[M].人民邮电出版社,2022

[4]张益珲.Vue.js3前端开发实战[M].清华大学出版社,2022