# 出租房屋管理系统详细设计

## 1. 引言

### 1.1 编写目的

本详细设计文档旨在为出租房屋管理系统的开发提供全面、精确的技术指南，明确系统各模块的具体实现细节，包括功能流程、数据结构、界面设计、算法逻辑等，确保开发团队能够依据文档高效、准确地构建系统，同时也为后续的测试、维护和优化工作奠定坚实基础。

### 1.2 背景

系统名称：出租房屋管理系统

项目提出者：客户

开发者：周春洁

用户：包括房东、租客及系统管理员

### 1.3 参考资料

《软件工程》

《出租房屋管理系统需求分析》

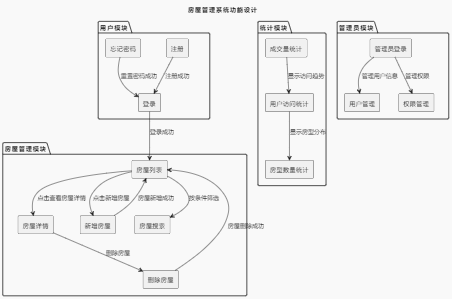
《出租房屋管理系统概要设计》

## 2. 系统概述

出租房屋管理系统是一个综合性信息管理系统，旨在帮助房东有效管理房源、租客信息及租赁合同，同时方便租客浏览房源信息并在线预约或租赁。系统的核心功能包括房源管理、租客管理、合同管理、房租支付及数据统计分析等，致力于提升房屋租赁过程的效率和用户体验。

## 系统架构设计

原型图设计

1. 

### 3.1 分层架构

表现层：采用响应式网页设计，支持不同终端设备访问。页面通过 HTML5、CSS3 和 JavaScript 等技术实现，并结合流行的前端框架（如 Vue.js 或 React），确保用户界面友好易用。

业务逻辑层：使用 Java Spring Boot 框架开发，负责实现系统的核心业务逻辑，如房源状态管理、租金计算等。

数据访问层：通过 MyBatis 实现与数据库的交互，封装数据访问接口，屏蔽底层数据库实现细节，便于维护和扩展。

数据库：采用 MySQL 数据库，存储房源、租客、合同等信息，提供高效的数据查询与管理能力。

### 3.2 技术选型

后端技术栈：Spring Boot + MyBatis + MySQL。

前端技术栈：Vue.js + Axios + Element UI。

开发语言：Java、JavaScript。

部署平台：Docker 容器，Nginx 作为反向代理服务器。

## 4. 功能模块详细设计

### 4.1 房源管理模块

功能描述：房东可通过此模块录入房源信息，包括地址、面积、租金、状态等。房东还可对房源信息进行编辑和删除。

输入输出：

- 输入：房源信息（地址、面积、租金、状态等）。

- 输出：操作结果提示（如“房源录入成功”）。

处理流程：

1. 接收房东输入的房源信息。

2. 校验信息完整性（如地址和租金为必填项）。

3. 将房源信息存入数据库。

4. 返回操作结果给用户。

### 4.2 租客管理模块

功能描述：管理员可管理租客信息，包括录入、修改和删除租客信息。

输入输出：

- 输入：租客信息（姓名、联系方式、身份证号等）。

- 输出：操作结果提示（如“租客信息更新成功”）。

处理流程：

1. 接收管理员输入的租客信息。

2. 校验信息的合法性（如身份证号格式是否正确）。

3. 更新数据库中的租客信息。

4. 返回操作结果给管理员。

### 4.3 合同管理模块

功能描述：系统可生成并管理租赁合同，支持在线签署和下载。

输入输出：

- 输入：合同相关信息（房源编号、租客编号、租赁时间、租金等）。

- 输出：合同生成成功提示，下载链接。

处理流程：

1. 接收房东和租客输入的合同信息。

2. 自动生成合同内容，存储在数据库中。

3. 返回合同生成结果并提供下载或查看功能。

## 5. 数据库设计

### 5.1 数据库表设计

# 出租房屋管理系统数据库设计

## HOUSES表（房源表）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | 类型 | 主键，外键 | 可以为空 | 默认值 | 说明 |
| HOUSE\_ID | VARCHAR2(50 BYTE) | 主键 | 否 |  | 房源的ID |
| ADDRESS | VARCHAR2(200 BYTE) |  | 否 |  | 房源的地址 |
| AREA | NUMBER(6,2) |  | 否 |  | 房源面积（㎡） |
| RENT | NUMBER(10,2) |  | 否 |  | 租金（¥） |
| STATUS | VARCHAR2(20 BYTE) |  | 否 | 空置 | 房源状态（空置、已出租、维修中） |
| CREATED\_AT | DATE |  | 否 | 当前时间 | 房源录入时间 |

## TENANTS表（租客表）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | 类型 | 主键，外键 | 可以为空 | 默认值 | 说明 |
| TENANT\_ID | VARCHAR2(50 BYTE) | 主键 | 否 |  | 租客的ID |
| NAME | VARCHAR2(50 BYTE) |  | 否 |  | 租客的姓名 |
| PHONE | VARCHAR2(15 BYTE) |  | 否 |  | 租客的联系电话 |
| ID\_NUMBER | VARCHAR2(18 BYTE) | 唯一 | 否 |  | 租客身份证号 |
| EMAIL | VARCHAR2(50 BYTE) |  | 是 | 空 | 租客的邮箱地址 |
| CREATED\_AT | DATE |  | 否 | 当前时间 | 租客注册时间 |

## CONTRACTS表（租赁合同表）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | 类型 | 主键，外键 | 可以为空 | 默认值 | 说明 |
| CONTRACT\_ID | VARCHAR2(50 BYTE) | 主键 | 否 |  | 合同的ID |
| HOUSE\_ID | VARCHAR2(50 BYTE) | 外键 | 否 |  | 房源的ID |
| TENANT\_ID | VARCHAR2(50 BYTE) | 外键 | 否 |  | 租客的ID |
| START\_DATE | DATE |  | 否 |  | 租赁开始日期 |
| END\_DATE | DATE |  | 否 |  | 租赁结束日期 |
| RENT\_AMOUNT | NUMBER(10,2) |  | 否 |  | 租金金额 |

## HOUSE\_SELECTION表（选房表）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | 类型 | 主键，外键 | 可以为空 | 默认值 | 说明 |
| SELECTION\_ID | VARCHAR2(50 BYTE) | 主键 | 否 |  | 选房记录的ID |
| TENANT\_ID | VARCHAR2(50 BYTE) | 外键 | 否 |  | 租客的ID |
| HOUSE\_ID | VARCHAR2(50 BYTE) | 外键 | 否 |  | 房源的ID |
| SELECTION\_DATE | DATE |  | 否 | 当前时间 | 选房日期 |

1. 房源表（Houses）：

- house\_id：房源编号（主键）

- address：房源地址

- area：面积

- rent：租金

- status：状态（空置、已出租等）

- created\_at：创建时间

2. 租客表（Tenants）：

- tenant\_id：租客编号（主键）

- name：租客姓名

- phone：联系电话

- id\_number：身份证号

3. 合同表（Contracts）：

- contract\_id：合同编号（主键）

- house\_id：房源编号（外键）

- tenant\_id：租客编号（外键）

- start\_date：租赁开始日期

- end\_date：租赁结束日期

- rent\_amount：租金总额

### 5.2 表结构

CREATE TABLE houses (

house\_id INT AUTO\_INCREMENT PRIMARY KEY,

address VARCHAR(255) NOT NULL,

area FLOAT NOT NULL,

rent FLOAT NOT NULL,

status ENUM("空置", "已出租", "维修中") DEFAULT "空置",

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE tenants (

tenant\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

phone VARCHAR(20) NOT NULL,

id\_number VARCHAR(18) UNIQUE NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE contracts (

contract\_id INT AUTO\_INCREMENT PRIMARY KEY,

house\_id INT NOT NULL,

tenant\_id INT NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE,

rent\_amount FLOAT NOT NULL,

FOREIGN KEY (house\_id) REFERENCES houses(house\_id),

FOREIGN KEY (tenant\_id) REFERENCES tenants(tenant\_id)

);

1. **接口用例设计**

## 1. 概述

本文档详细描述了房屋管理系统的接口设计，包括各个功能模块的接口定义、参数说明、返回值格式等内容。旨在为开发和测试人员提供清晰的接口规范，确保系统的高效开发与运行。

## 2. 接口清单

以下是房屋管理系统的接口列表：

1. 获取房屋列表接口（GET /get-houses）

2. 新增房屋接口（POST /get-houses）

3. 删除房屋接口（DELETE /get-houses）

4. 登录接口（POST /login）

5. 获取统计数据接口（GET /get-stats）

## 3. 接口设计详情

## 2. 接口清单

以下是房屋管理系统的接口列表：

1. 获取房屋列表接口（GET /get-houses）

2. 新增房屋接口（POST /get-houses）

3. 删除房屋接口（DELETE /get-houses）

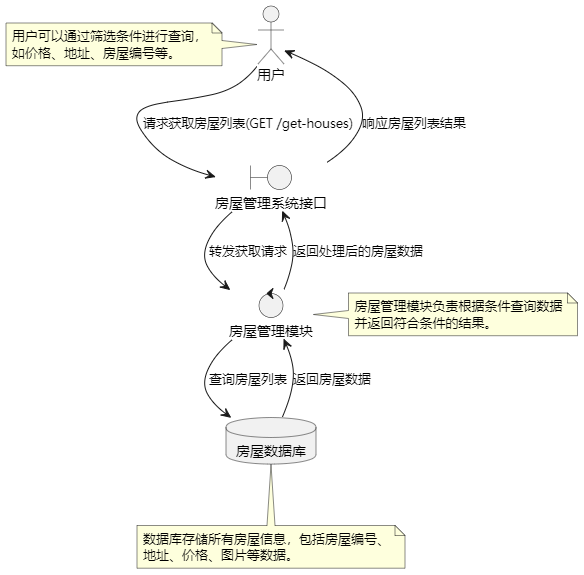
4. 登录接口（POST /login）

5. 获取统计数据接口（GET /get-stats）

## 3. 接口设计详情

### 3.1 获取房屋列表接口

流程图



功能：查询符合条件的房屋信息列表。

API 请求地址：/get-houses

请求方式：GET

#### 请求参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| address | String | 否 | 按地址模糊搜索 |
| id | Integer | 否 | 按房屋编号精确搜索 |
| minPrice | Double | 否 | 最低价格 |
| maxPrice | Double | 否 | 最高价格 |
| priceOrder | String | 否 | 价格排序方式，取值为 "asc" 或 "desc" |

#### 返回参数说明：

返回值为 JSON 格式的房屋列表，包含以下字段：

[

{

"id": 201,

"address": "北京市海淀区中关村大街",

"price": 3500.0,

"contact": "010-88888888",

"imageUrls": ["/path/1.jpg", "/path/2.jpg"]

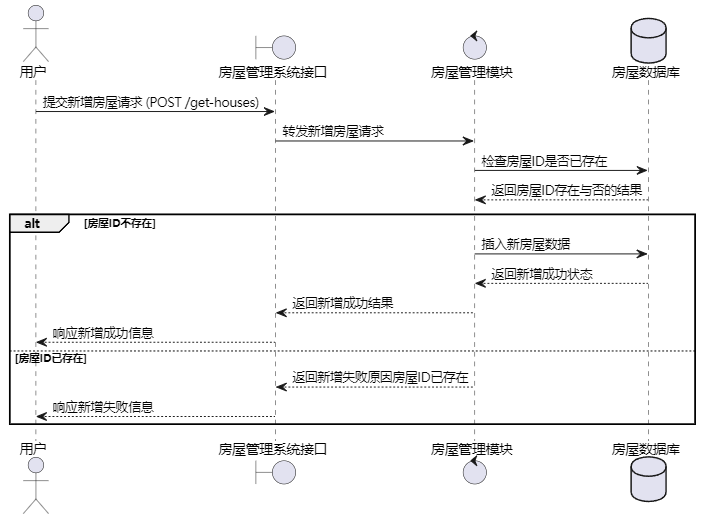
},

...

]



### 3.2 新增房屋接口



功能：添加新的房屋信息到系统中。

API 请求地址：/get-houses

请求方式：POST

#### 请求参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 类型 | 是否必填 | 说明 |
| id | Integer | 是 | 房屋编号，需唯一 |
| address | String | 是 | 房屋地址 |
| price | Double | 是 | 出租价格 |
| contact | String | 是 | 联系电话 |
| imageUrls | Array<String> | 是 | 房屋图片 URL 列表 |

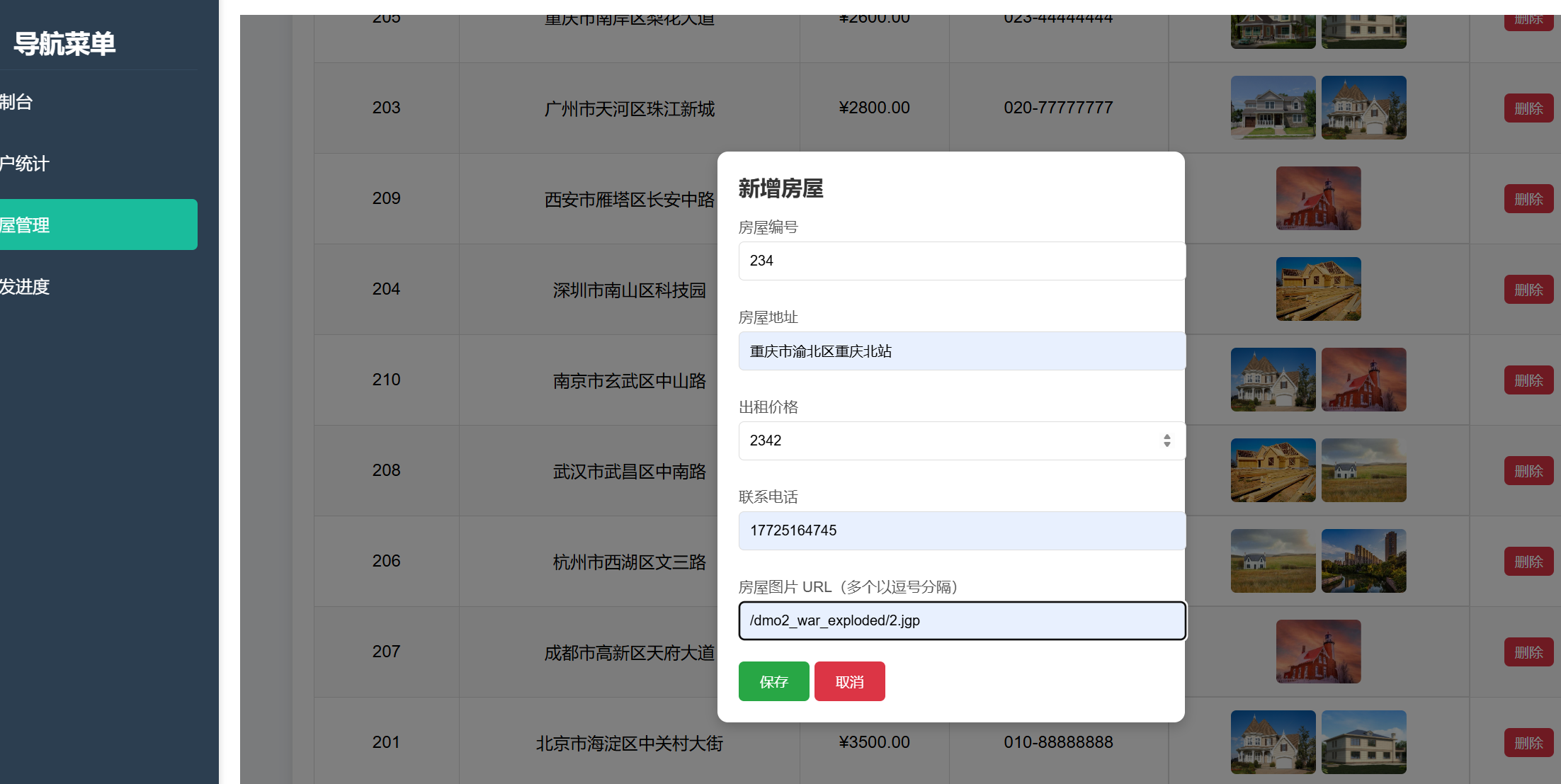
#### 返回参数说明：

{

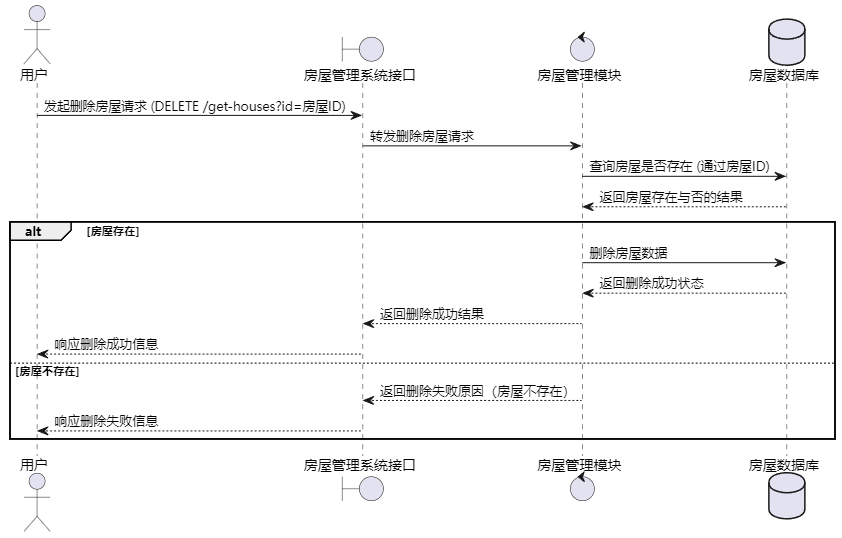
"status": "success" | "failure",

"message": "相关提示信息"

}



#### ****接口3：删除房屋接口****



* **功能**：删除指定房屋信息。
* **权限**：用户必须登录后才可以删除房屋。
* **API请求地址**：/get-houses
* **请求方式**：DELETE

##### ****请求实例****：

json

{

"id": 101}

##### ****请求参数说明****：

| **参数名称** | **类型** | **是否必填** | **说明** |
| --- | --- | --- | --- |
| id | int | 是 | 要删除的房屋ID。 |

##### ****返回实例****：

**成功**：

json

{

"status": "success"}

**失败**：

json

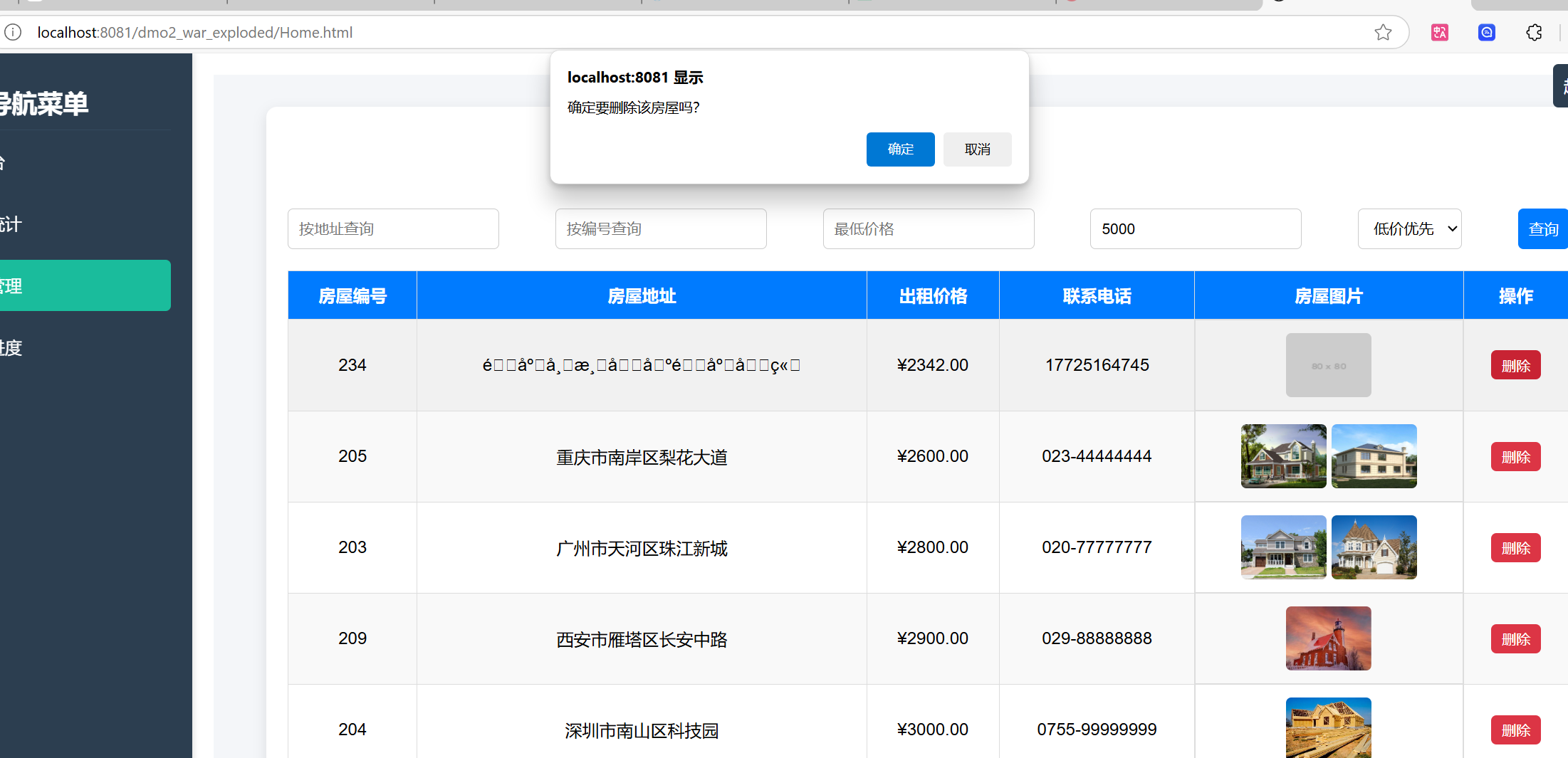
{

"status": "failure",

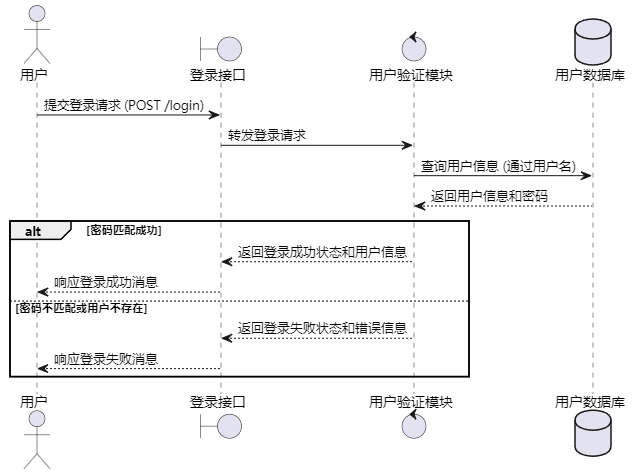
"message": "房屋ID不存在"}

##### ****返回参数说明****：

| **参数名称** | **类型** | **说明** |
| --- | --- | --- |
| status | string | success表示删除成功，failure表示删除失败。 |
| message | string | 返回失败时的具体原因。 |



#### ****接口4：登录接口****



* **功能**：实现用户的登录验证。
* **权限**：所有用户均可使用。
* **API请求地址**：/login
* **请求方式**：POST

##### ****请求实例****：

json

{

"username": "test\_user",

"password": "test\_password"}

##### ****请求参数说明****：

| **参数名称** | **类型** | **是否必填** | **说明** |
| --- | --- | --- | --- |
| username | string | 是 | 用户的用户名。 |
| password | string | 是 | 用户的登录密码。 |

##### ****返回实例****：

**成功**：

json

{

"success": true,

"message": "登录成功！"}

**失败**：

json

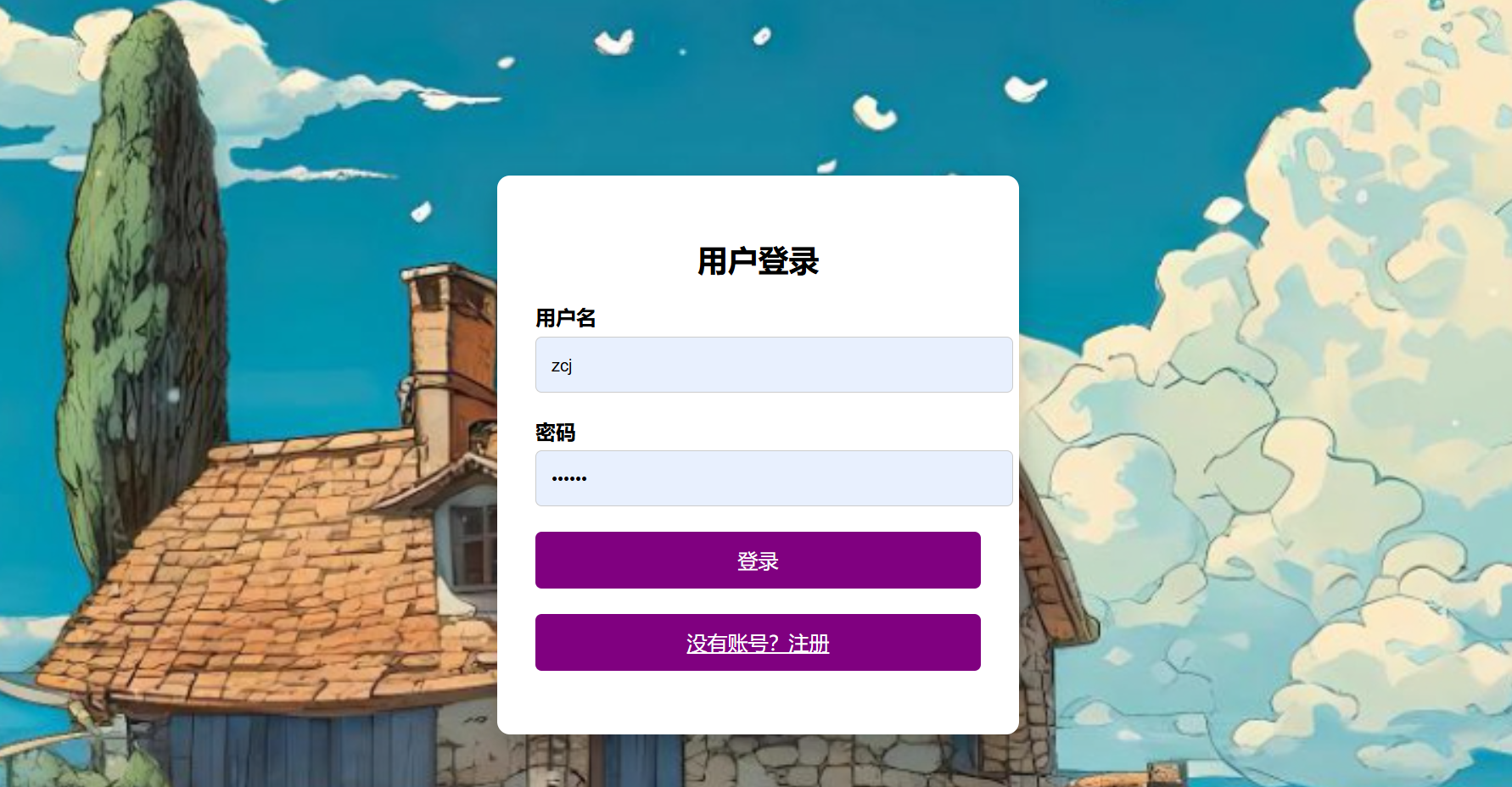
{

"success": false,

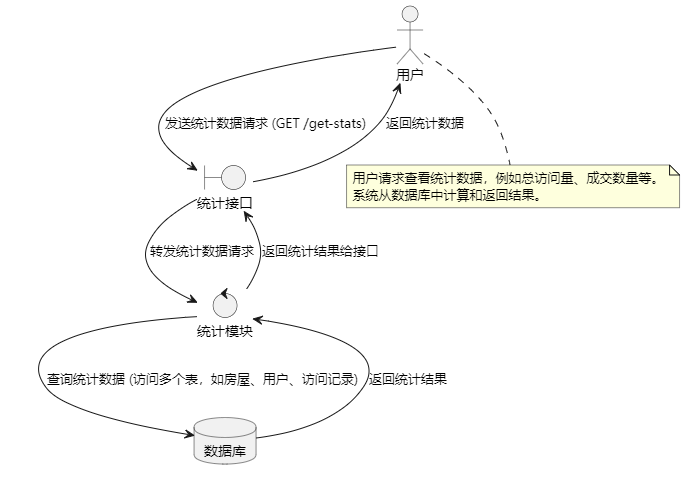
"message": "密码错误，请重试。"}

##### ****返回参数说明****：

| **参数名称** | **类型** | **说明** |
| --- | --- | --- |
| success | bool | true表示登录成功，false表示登录失败。 |
| message | string | 提示用户的登录结果信息。 |



#### ****接口5：********获取统计数据接口****



* **功能**：返回系统的统计数据，包括成交数量、访问量、房屋总数等。
* **权限**：无需登录即可调用。
* **API请求地址**：/get-stats
* **请求方式**：GET

##### ****请求实例****：

无请求参数。

##### ****返回实例****：

json

{

"totalTransactions": 45,

"totalVisits": 1200,

"totalHouses": 89,

"newUserCount": 10}

##### ****返回参数说明****：

| **参数名称** | **类型** | **说明** |
| --- | --- | --- |
| totalTransactions | int | 成交房屋的总数量。 |
| totalVisits | int | 系统访问量。 |
| totalHouses | int | 当前系统中房屋的总数量。 |
| newUserCount | int | 新增注册用户的数量。 |

1. 