# 引言

当前我国经济运行总体平稳、稳中有进，势头向好，国民可支配收入逐渐增 加，消费结构悄然发生改变，精神文明消费比重逐渐增大。酒店作为旅游业中的 重要部分，面临着巨大的发展机遇。在目前中国浓郁的创新氛围中，许多传统酒店正在转型 “智慧酒店”，成为各大企业的投资、发展热点。

在这样的背景下，酒店都迫切的需求一套包含用户端和管理端的酒店系统。该系统能够提高用户的入住体验也可以提高酒店方的管理效率。本项目将人工智能（计算机视觉）与物联网（智慧家居）结合，打破传统的酒店运营与服务模式：将酒店的管理由被动变为主动，提高酒店的入住率，转化率；酒店管理人员通过智能化的云管理平台，可实现对酒店设备和客房的智能管理，便于酒店的升级和维护，极大地降低酒店运维成本；客户则可以便捷地实现从入住到最后的一键退房，不仅为客户节省时间，而且带来智能化体验，提高客人满意度。

本项目通过充分利用手机端和管理员端的多端优势，扩展酒店功能，提升酒店的管理效率，减少人力成本，同时为住客提供更人性化的入住体验。本项目通过小程序与住客及管理员端交互，简单易用，适用于任何年龄的住客用户及管理者；本项目在整体功能的构思方面，考虑到用户的隐私保护，系统安全保障等，使得酒店管理者无后顾之忧；本项目只需要在目前的酒店设施上增加少量的硬件，对于酒店转型不需太高成本，便于项目推广。

# 参考文献

[1]吕治江. 中小型酒店管理系统的设计与实现[D].厦门大学,2017.

[2]朱政. 中小型酒店智能化管理系统的设计与实现[D].湖南大学,2018.

[3]Alais David,Xu Yiben,Wardle Susan G,Taubert Jessica. A shared mechanism for facial expression in human faces and face pareidolia.[J]. Proceedings. Biological sciences,2021,288(1954).

[4]贾沃斯基,莱德. Python高级编程[M].人民邮电出版社:{3}, 201710.423.

[5]Kunal Relan. Building REST APIs with Flask[M].Apress, Berkeley, CA:2019-01-01.

[6]刘刚. 微信小程序开发图解案例教程[M].人民邮电出版社:{3}, 201705.357.

[7]陈陆扬. Vue.js前端开发快速入门与专业应用[M].人民邮电出版社:{3}, 201702.207.

[8]单东林,张晓菲,魏然. 锋利的jQuery[M].人民邮电出版社:{3}, 201206.393.

[9]张海藩,吕云翔. 软件工程[M].人民邮电出版社:{3}, 201309.348.

# 

# 附录-项目部分代码

项目代码量较大，下列出后端部分逻辑代码。

"""  
Web端房间管理部分数据库查询代码  
"""  
from . import Model  
from datetime import datetime  
  
  
class RoomOption(Model):  
 def \_\_init\_\_(self):  
 super(RoomOption, self).\_\_init\_\_()  
  
 def search\_order(self, page, limit, sort, id, rtype):  
 start = (page - 1) \* limit  
 # 计算订单数量  
 self.cursor.execute('SELECT count(\*) FROM room;')  
 total\_num = self.cursor.fetchone()[0]  
 if id != None:  
 self.cursor.execute(  
 f"SELECT \* FROM room WHERE id={id};")  
 elif rtype != None:  
 self.cursor.execute(  
 f"SELECT \* FROM room WHERE rtype='{rtype}';")  
 elif sort == '+id':  
 self.cursor.execute(  
 f"SELECT \* FROM room ORDER BY id DESC LIMIT {start},{limit};")  
 else:  
 self.cursor.execute(  
 f"SELECT \* FROM room ORDER BY id LIMIT {start},{limit};")  
 data = self.cursor.fetchall()  
 return data, total\_num  
  
 # 更新房间数据表  
 def update(self, id, rtype, bedtype, maxnum, area, rwin, rlock, money, temperature, humidity):  
 self.cursor.execute(  
 f"UPDATE room SET id='{id}',rtype='{rtype}',bedtype='{bedtype}',maxnum='{maxnum}',area='{area}',rwin='{rwin}',rlock='{rlock}',money='{money}',temperature='{temperature}',humidity='{humidity}' WHERE id={id};")  
 try:  
 self.db.commit()  
 except:  
 self.db.rollback()  
 self.cursor.close()  
 self.db.close()  
 return False  
 self.cursor.close()  
 self.db.close()  
 return True  
  
 def add(self, rtype, bedtype, maxnum, area, rwin, rlock, money):  
 self.cursor.execute("SELECT MAX(\*) FROM room;")  
 id = int(self.cursor.fetchone()[0]) + 1  
 data = (id, rtype, bedtype, maxnum, area, rwin, money, 20, 20, rlock)  
 self.cursor.execute(  
 f"INSERT INTO room values {data};")  
 try:  
 self.db.commit()  
 except:  
 self.db.rollback()  
 self.cursor.close()  
 self.db.close()  
 return False  
 self.cursor.close()  
 self.db.close()  
 return True  
  
 # 删除房间信息  
 def delete(self, id):  
 self.cursor.execute(  
 f"DELETE FROM room WHERE id={id};")  
 try:  
 self.db.commit()  
 except:  
 self.db.rollback()  
 self.cursor.close()  
 self.db.close()  
 return False  
 self.cursor.close()  
 self.db.close()  
 return True  
  
 def search\_guest(self, page, limit, bedtype, start\_data, end\_data):  
 start = (page - 1) \* limit  
 print(start\_data, end\_data)  
 # 查询在时间段内可以住的房间  
 if start\_data is not None and end\_data is not None:  
 if bedtype is not None:  
 self.cursor.execute(  
 f"SELECT \* FROM room WHERE bedtype = '{bedtype}' AND id NOT IN (SELECT room\_id FROM `order` WHERE id\_status IN (0,3) AND scid >= '{start\_data}' AND scid <= '{end\_data}' AND sgo >= '{start\_data}' AND sgo <= '{end\_data}') ORDER BY id LIMIT {start},{limit};")  
 else:  
 self.cursor.execute(  
 f"SELECT \* FROM room WHERE id NOT IN (SELECT room\_id FROM `order` WHERE id\_status IN (0,3) AND scid >= '{start\_data}' AND scid <= '{end\_data}' AND sgo >= '{start\_data}' AND sgo <= '{end\_data}') ORDER BY id LIMIT {start},{limit};")  
 else:  
 # 默认查询最近三天内可以住的房间  
 self.cursor.execute(  
 f"SELECT \* FROM room WHERE id NOT IN (SELECT room\_id FROM `order` WHERE id\_status IN (0,3) AND scid > NOW() AND scid < DATE\_ADD(NOW(),INTERVAL 3 DAY ) AND sgo > NOW() AND sgo < DATE\_ADD(NOW(),INTERVAL 3 DAY )) ORDER BY id LIMIT {start},{limit};")  
 data = self.cursor.fetchall()  
 return data  
   
"""  
Web端 用户管理模块，数据库查询部分代码  
"""  
  
from . import Model  
from datetime import datetime  
  
  
class UserOption(Model):  
 def \_\_init\_\_(self):  
 super(UserOption, self).\_\_init\_\_()  
  
 def search\_user(self, page, limit, level, name, phone, search\_level):  
 start = (page - 1) \* limit  
 # 获取用户数量  
 self.cursor.execute('SELECT MAX(id) FROM room;')  
 total\_num = self.cursor.fetchone()[0]  
 if phone != None:  
 self.cursor.execute(  
 f"SELECT \* FROM user WHERE phone={phone} AND level>={level} LIMIT {start},{limit};")  
 elif name != None:  
 self.cursor.execute(  
 f"SELECT \* FROM user WHERE name='{name}' AND level>={level} LIMIT {start},{limit};")  
 elif search\_level != None:  
 self.cursor.execute(  
 f"SELECT \* FROM user WHERE level={search\_level} AND level>={level} LIMIT {start},{limit};")  
 else:  
 self.cursor.execute(  
 f"SELECT \* FROM user WHERE level>={level} LIMIT {start},{limit};")  
 data = self.cursor.fetchall()  
 return data, total\_num  
  
 # 更新用户数据表  
 def update(self, wecharid, name, sex, id\_card, phone, level):  
 print(wecharid)  
 try:  
 self.cursor.execute(  
 f"UPDATE user SET name='{name}',sex='{sex}',id\_card='{id\_card}',phone='{phone}',level={level} WHERE wecharid={wecharid};")  
 self.db.commit()  
 except Exception as e:  
 print(e)  
 self.db.rollback()  
 self.cursor.close()  
 self.db.close()  
 return False  
 self.cursor.close()  
 self.db.close()  
 return True  
  
 def add(self, wecharid, name, sex, id\_card, phone, level):  
 self.cursor.execute("SELECT COUNT(\*) FROM room;")  
 data = (wecharid, name, sex, id\_card, phone, level)  
 self.cursor.execute(  
 f"INSERT INTO user values {data};")  
 try:  
 self.db.commit()  
 except:  
 self.db.rollback()  
 self.cursor.close()  
 self.db.close()  
 return False  
 self.cursor.close()  
 self.db.close()  
 return True  
  
 # 删除用户信息  
 def delete(self, id):  
 self.cursor.execute(  
 f"DELETE FROM room WHERE wecharid='{id}';")  
 try:  
 self.db.commit()  
 except:  
 self.db.rollback()  
 self.cursor.close()  
 self.db.close()  
 return False  
 self.cursor.close()  
 self.db.close()  
 return True  
  
"""  
Web端用户模块，路由响应代码  
"""  
import datetime  
import time  
import json  
  
from flask import Blueprint  
from flask import request  
from ..models.get\_info import GetInfo  
from ..models.room\_option import RoomOption  
from ..utils.utils import black2none  
  
room\_list = Blueprint('room\_list', \_\_name\_\_)  
  
from ..utils.utils import catch\_except  
  
  
@room\_list.route('/room/list', methods=['GET'])  
@catch\_except  
def index():  
 # 获取参数  
 token = request.args.get("token")  
 page = request.args.get("page")  
 limit = request.args.get("limit")  
 sort = request.args.get("sort")  
 id = request.args.get("id")  
 rtype = request.args.get("rtype")  
 # 去除无意义参数  
 id, rtype = black2none(id, rtype)  
 if token is None or page is None or limit is None or sort is None:  
 return json.dumps({"code": 20004, "message": "参数错误，请检查参数"})  
 try:  
 page = int(page)  
 limit = int(limit)  
 except:  
 return json.dumps({"code": 20004, "message": "参数错误，请检查参数"})  
 db = GetInfo()  
 room\_db = RoomOption()  
 user\_info = db.search(token)  
 if user\_info is None:  
 return json.dumps({"code": 20005, "message": "用户不存在"})  
 level = int(user\_info[-1])  
 # 身份等级验证  
 if level > 2:  
 return json.dumps({"code": 20006, "message": "权限不足"})  
 rooms, total\_num = room\_db.search\_order(page, limit, sort, id, rtype)  
 data = {  
 "code": 20000,  
 "data": {  
 "total": total\_num,  
 "items": []  
 }  
 }  
 for room in rooms:  
 item = {  
 "id": room[0],  
 "rtype": room[1],  
 "bedtype": room[2],  
 "maxnum": room[3],  
 "area": room[4],  
 "rwin": room[5],  
 "money": room[6],  
 "temperature": room[7],  
 "humidity": room[8],  
 "rlock": room[9],  
 }  
 data["data"]["items"].append(item)  
 return json.dumps(data)