```
#!/usr/bin/env python3
"""
Cliente de Autenticación SDN
Fecha: 2025-11-01 14:30:41 UTC
Usuario: Cjfs2005
"""

import requests
import json
import sys
import getpass

# Configuración
AUTH_SERVER = 'http://192.168.200.200:5000'
TIMEOUT = 10  # segundos

def print_banner():
    """Imprime banner del sistema"""
    print("=" * 60)
    print("  SISTEMA DE AUTENTICACIÓN SDN")
    print("  Universidad / Organización")
    print("  Fecha: 2025-11-01 14:30:41 UTC")
    print("=" * 60)
    print()

def get_user_choice():
    """Pregunta al usuario si es invitado o registrado"""
    print("Seleccione su tipo de acceso:")
    print("  1. Invitado (Guest)")
    print("  2. Usuario Registrado")
    print()

    while True:
        choice = input("Ingrese su opción (1 o 2): ").strip()

        if choice == '1':
            return 'guest'
        elif choice == '2':
            return 'registered'
        else:
            print("❌ Opción inválida. Por favor ingrese 1 o 2.")
            print()

def authenticate_guest():
    """Autentica un usuario invitado"""
    print()
    print("─" * 60)
    print("  ACCESO PARA INVITADOS")
print("—" * 60)
print()

email = input("Ingrese su correo electrónico: ").strip()

if not email:
    print("❌ El correo es obligatorio")
    return False

print()
print("⏳ Autenticando...")

try:
    response = requests.post(
        f"{AUTH_SERVER}/api/auth/guest",
        json={'email': email},
        timeout=TIMEOUT
    )

    if response.status_code == 200:
        data = response.json()
        print()
        print("=" * 60)
        print("  ✅ ACCESO CONCEDIDO")
        print("=" * 60)
        print(f"  Tipo de usuario: Invitado")
        print(f"  Email: {data['email']}")
        print(f"  Session ID: {data['session_id']}")
        print(f"  Tu IP: {data['client_info']['ip']}")
        print(f"  Tu MAC: {data['client_info']['mac']}")
        print("=" * 60)
        print()
        print("✓ Ya puedes navegar por Internet")
        return True

    else:
        error_data = response.json()
        print()
        print(f"❌ Error: {error_data.get('error', 'Authentication failed')}")
        return False

except requests.exceptions.ConnectionError:
    print()
    print("❌ Error: No se puede conectar al servidor de autenticación")
    print(f"   Servidor: {AUTH_SERVER}")
    print("   Verifica tu conexión de red")
    return False
    except requests.exceptions.Timeout:
        print()
        print("❌ Error: Tiempo de espera agotado")
        print("   El servidor no responde")
        return False

    except Exception as e:
        print()
        print(f"❌ Error inesperado: {e}")
        return False

def authenticate_registered():
    """Autentica un usuario registrado"""
    print()
    print("─" * 60)
    print("  ACCESO PARA USUARIOS REGISTRADOS")
    print("─" * 60)
    print()

    email = input("Ingrese su correo electrónico: ").strip()
    password = getpass.getpass("Ingrese su contraseña: ")

    if not email or not password:
        print("❌ El correo y contraseña son obligatorios")
        return False

    print()
    print("⏳ Autenticando...")

    try:
        response = requests.post(
            f"{AUTH_SERVER}/api/auth/login",
            json={'email': email, 'password': password},
            timeout=TIMEOUT
        )

        if response.status_code == 200:
            data = response.json()
            print()
            print("=" * 60)
            print("  ✅ ACCESO CONCEDIDO")
            print("=" * 60)
            print(f"  Tipo de usuario: Registrado")
            print(f"  Email: {data['email']}")
            print(f"  Session ID: {data['session_id']}")
            print(f"  Tu IP: {data['client_info']['ip']}")
            print(f"  Tu MAC: {data['client_info']['mac']}")
            print("=" * 60)
            print()
            print("✓ Ya puedes navegar por Internet")
            return True

        elif response.status_code == 401:
            print()
            print("❌ Credenciales inválidas")
            print("   Verifica tu correo y contraseña")
            return False

        else:
            error_data = response.json()
            print()
            print(f"❌ Error: {error_data.get('error', 'Authentication failed')}")
            return False

    except requests.exceptions.ConnectionError:
        print()
        print("❌ Error: No se puede conectar al servidor de autenticación")
        print(f"   Servidor: {AUTH_SERVER}")
        print("   Verifica tu conexión de red")
        return False

    except requests.exceptions.Timeout:
        print()
        print("❌ Error: Tiempo de espera agotado")
        print("   El servidor no responde")
        return False

    except Exception as e:
        print()
        print(f"❌ Error inesperado: {e}")
        return False

def main():
    """Función principal"""
    print_banner()

    # Verificar conectividad con el servidor
    try:
        response = requests.get(f"{AUTH_SERVER}/", timeout=5)
        if response.status_code != 200:
            print("⚠️  Advertencia: El servidor respondió con un código inusual")
    except:
        print("❌ ERROR: No se puede conectar al servidor de autenticación")
        print(f"   Servidor: {AUTH_SERVER}")
        print()
        print("Posibles causas:")
        print("  - No tienes conexión de red")
        print("  - El servidor está apagado")
        print("  - Firewall bloqueando el acceso")
        print()
        sys.exit(1)

    # Obtener tipo de usuario
    user_type = get_user_choice()

    # Autenticar según el tipo
    if user_type == 'guest':
        success = authenticate_guest()
    else:
        success = authenticate_registered()

    # Resultado final
    print()
    if success:
        print("=" * 60)
        print("  AUTENTICACIÓN EXITOSA")
        print("=" * 60)
        sys.exit(0)
    else:
        print("=" * 60)
        print("  AUTENTICACIÓN FALLIDA")
        print("=" * 60)
        print()
        print("Si el problema persiste, contacta al administrador")
        sys.exit(1)

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print()
        print()
        print("⚠️  Autenticación cancelada por el usuario")
        sys.exit(130)
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Portal de Autenticación SDN con Flask
Fecha: 2025-11-01 14:30:41 UTC
Usuario: Cjfs2005
"""

from flask import Flask, request, jsonify
from datetime import datetime
import logging
import time
import json
import requests   # <--- IMPORTANTE

# Configuración de logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s [%(levelname)s] %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S UTC'
)
logger = logging.getLogger(__name__)

app = Flask(__name__)

# ==============================================
# CONFIGURACIÓN
# ==============================================

REGISTERED_USERS = {
    'usuario@example.com': 'password123',
    'admin@sdn.local': 'admin2025',
    'student@university.edu': 'student123'
}

LEASE_FILE = '/var/lib/misc/dnsmasq.leases'
active_sessions = {}

# ==============================================
# FUNCIONES AUXILIARES
# ==============================================

def get_mac_from_ip(ip_address):
    try:
        with open(LEASE_FILE, 'r') as f:
            for line in f:
```

```
            parts = line.strip().split()
            if len(parts) >= 3:
                expiration = int(parts[0])
                mac = parts[1]
                ip = parts[2]
                if ip == ip_address and time.time() < expiration:
                    return mac.lower()
    except Exception as e:
        logger.error(f"Error reading lease file: {e}")
    return None


def get_switch_attachment(mac):
    """
    Obtiene DPID y puerto desde /wm/device/
    """
    try:
        url = "http://127.0.0.1:8080/wm/device/"
        response = requests.get(url, timeout=2)
        devices = response.json()

        mac = mac.lower()

        for dev in devices:
            dev_macs = [m.lower() for m in dev.get("mac", [])]
            if mac in dev_macs:
                ap_list = dev.get("attachmentPoint", [])
                if ap_list:
                    ap = ap_list[0]  # tomar primer AP
                    return {
                        "switch_dpid": ap.get("switchDPID"),
                        "switch_port": ap.get("port")
                    }
    except Exception as e:
        logger.warning(f"Error consultando Floodlight /wm/device/ para MAC {mac}: {e}")

    return {"switch_dpid": None, "switch_port": None}


def get_client_info(request):
    if request.headers.get('X-Forwarded-For'):
        client_ip = request.headers.get('X-Forwarded-For').split(',')[0].strip()
    else:
        client_ip = request.remote_addr

    client_mac = get_mac_from_ip(client_ip)
    user_agent = request.headers.get('User-Agent', 'Unknown')
    # Obtener switch y puerto
    attach = get_switch_attachment(client_mac) if client_mac else {"switch_dpid": None,
"switch_port": None}

    return {
        'ip': client_ip,
        'mac': client_mac if client_mac else 'Unknown',
        'switch_dpid': attach["switch_dpid"],
        'switch_port': attach["switch_port"],
        'user_agent': user_agent,
        'timestamp': datetime.utcnow().strftime('%Y-%m-%d %H:%M:%S UTC')
    }


def log_access(email, user_type, status, client_info):
    log_msg = (
        f"ACCESS {status} | "
        f"Type: {user_type.upper()} | "
        f"Email: {email} | "
        f"IP: {client_info['ip']} | "
        f"MAC: {client_info['mac']} | "
        f"DPID: {client_info['switch_dpid']} | "
        f"Port: {client_info['switch_port']}"
    )
    (logger.info if status == 'SUCCESS' else logger.warning)(log_msg)


# ============================================
# ENDPOINTS
# ============================================

@app.route('/', methods=['GET'])
def index():
    return jsonify({
        'service': 'SDN Authentication Portal',
        'version': '1.1 (con Floodlight attachment point)',
        'endpoints': {
            'POST /api/auth/guest': 'Autenticación para invitados',
            'POST /api/auth/login': 'Autenticación para usuarios registrados',
            'GET /api/session/<mac>': 'Verificar sesión activa',
            'GET /api/stats': 'Estadísticas de accesos'
        }
    }), 200


@app.route('/api/auth/guest', methods=['POST'])
def auth_guest():
    client_info = get_client_info(request)
    data = request.get_json()
    if not data or 'email' not in data:
        return jsonify({'success': False, 'error': 'Email is required'}), 400

    email = data['email']
    session_id = f"guest_{client_info['mac']}_{int(time.time())}"

    active_sessions[client_info['mac']] = {
        **client_info,
        'email': email,
        'type': 'guest',
        'session_id': session_id
    }

    log_access(email, 'guest', 'SUCCESS', client_info)

    return jsonify({
        'success': True,
        'user_type': 'guest',
        'session_id': session_id,
        'client_info': client_info
    }), 200


@app.route('/api/auth/login', methods=['POST'])
def auth_login():
    client_info = get_client_info(request)

    data = request.get_json()
    if not data or 'email' not in data or 'password' not in data:
        return jsonify({'success': False, 'error': 'Email and password are required'}), 400

    email, password = data['email'], data['password']

    if email in REGISTERED_USERS and REGISTERED_USERS[email] == password:
        session_id = f"registered_{client_info['mac']}_{int(time.time())}"
        active_sessions[client_info['mac']] = {
            **client_info,
            'email': email,
            'type': 'registered',
            'session_id': session_id
        }

        log_access(email, 'registered', 'SUCCESS', client_info)

        return jsonify({
            'success': True,
            'user_type': 'registered',
            'session_id': session_id,
            'client_info': client_info
        }), 200

    log_access(email, 'registered', 'FAILED', client_info)
    return jsonify({'success': False, 'error': 'Invalid credentials'}), 401


@app.route('/api/session/<mac>', methods=['GET'])
def check_session(mac):
    mac = mac.lower().replace('-', ':')
    if mac in active_sessions:
        return jsonify({'active': True, 'session': active_sessions[mac]}), 200
    return jsonify({'active': False}), 404


@app.route('/api/stats', methods=['GET'])
def get_stats():
    total = len(active_sessions)
    guests = sum(s['type'] == 'guest' for s in active_sessions.values())
    regs = total - guests
    return jsonify({
        'total_sessions': total,
        'guest_sessions': guests,
        'registered_sessions': regs,
        'sessions': list(active_sessions.values())
    }), 200


@app.route('/api/logout/<mac>', methods=['POST'])
def logout(mac):
    mac = mac.lower().replace('-', ':')
    if mac in active_sessions:
        active_sessions.pop(mac)
        return jsonify({'success': True}), 200
    return jsonify({'success': False}), 404


# ============================================
# MAIN
# ============================================

if __name__ == '__main__':
    app.run(host='192.168.200.200', port=5000, debug=False, threaded=True)
```