```
# Fragmento actualizado del endpoint add_payment @payment_router.post( "/payments/add",
summary="Registrar un nuevo pago", description="**Permisos requeridos: `administrador`**.
Registra un pago para la factura pendiente más antigua de un usuario, la marca como pagada y
genera un recibo en PDF.", ) def add_payment( payment_data: InputPayment, admin_user: dict =
Depends(is_admin), db: Session = Depends(get_db), ): try: # 1. Buscar la factura pendiente más
antigua invoice_to_pay = ( db.query(Invoice) .join(User, Invoice.user_id ==
User.id) .options( joinedload(Invoice.user).joinedload(User.userdetail),
joinedload(Invoice.subscription).joinedload(Subscription.plan), ) .filter(User.id ==
payment_data.user_id, Invoice.status == "pending") .order_by(Invoice.issue_date) .first() ) if not
invoice_to_pay: return JSONResponse( status_code=404, content={"message": "No se encontró
una factura pendiente."}, ) # 2. Actualizar estado y crear el pago invoice_to_pay.status = "paid"
new_payment = Payment( user_id=payment_data.user_id, amount=payment_data.amount,
invoice_id=invoice_to_pay.id, ) db.add(new_payment) db.flush() db.refresh(new_payment) # 3.
Preparar los datos para el PDF con el formato del recibo de referencia user_details =
invoice_to_pay.user.userdetail plan_details = invoice_to_pay.subscription.plan # Formatear el
mes del servicio en español meses = { 1: "Enero", 2: "Febrero", 3: "Marzo", 4: "Abril", 5:
"Mayo", 6: "Junio", 7: "Julio", 8: "Agosto", 9: "Septiembre", 10: "Octubre", 11: "Noviembre", 12:
"Diciembre" } mes_servicio = f"{meses[invoice_to_pay.issue_date.month]}
{invoice_to_pay.issue_date.year}" # Obtener la ruta absoluta del logo (buscar en diferentes
formatos) logo_formats = ['logo.png', 'logo.jpg', 'logo.jpeg', 'logo.svg'] logo_path = None for
logo_name in logo_formats: potential_path = os.path.abspath(os.path.join("templates",
logo_name)) if os.path.exists(potential_path): logo_path = potential_path break if not logo_path:
print("Advertencia: No se encontró el logo de la empresa en la carpeta templates/") # Formatear
el número de recibo receipt_number = f"F{new_payment.payment_date.year}-
{invoice_to_pay.id:03d}" pdf_data = { "company_name": "NetSys Solutions",
"company_address": "Calle Ficticia 123, Ciudad Ejemplo", "company_contact": "Tel: 900 123
456 | Email: contacto@netsys.com", "logo_path": logo_path, "client_name":
f"{user_details.firstname} {user_details.lastname}", "client_dni": user_details.dni,
"client_address": user_details.address, "client_barrio": user_details.barrio, "client_city":
user_details.city, "client_phone": user_details.phone, "client_email": invoice_to_pay.user.email,
"receipt_number": receipt_number, "payment_date": new_payment.payment_date.strftime("%d/
%m/%Y"), "due_date": invoice_to_pay.due_date.strftime("%d/%m/%Y"), "item_description":
f"Servicio Internet Premium Fibra {plan_details.speed_mbps}MB - {mes_servicio}",
"base_amount": invoice_to_pay.base_amount, "late_fee": invoice_to_pay.late_fee, "total_paid":
new_payment.amount, "invoice_id": invoice_to_pay.id, } # 4. Generar el PDF pdf_filename =
create_invoice_pdf(pdf_data) invoice_to_pay.receipt_pdf_url =
os.path.join( str(new_payment.payment_date.year), f"{new_payment.payment_date.month:
02d}", pdf_filename, ).replace("\\", "/") # 5. Confirmar y responder db.commit() return
JSONResponse( status_code=201, content={ "message": "Pago registrado exitosamente.",
"receipt_number": receipt_number, "pdf_filename": pdf_filename, "total_paid":
new_payment.amount } ) except Exception as e: db.rollback() print(f"Error en /payments/add:
{e}\n{traceback.format_exc()}") return JSONResponse( status_code=500, content={"error":
"Ocurrió un error interno al procesar el pago."} )
```