

# MANUAL TÉCNICO

## **MoskoGás Backend v2**

Sistema de Gestão de Pedidos  
Integração Bling ERP + IzChat WhatsApp

Versão: 2.7.0 | Data: 16/02/2026

# ÍNDICE

1. VISÃO GERAL DO SISTEMA
2. ARQUITETURA TÉCNICA
3. SCHEMA DO BANCO DE DADOS (D1)
4. ENDPOINTS DA API
5. SISTEMA DE PAGAMENTOS
6. INTEGRAÇÃO BLING ERP
7. INTEGRAÇÃO IZCHAT (WhatsApp)
8. FLUXOS DE NEGÓCIO
9. DECISÕES ARQUITETURAIS
10. ERROS COMUNS E SOLUÇÕES
11. CHECKLIST DE DEPLOYMENT
12. CONFIGURAÇÕES IMPORTANTES

# 1. VISÃO GERAL DO SISTEMA

O MoskoGás é um sistema web para gestão de pedidos de gás e água desenvolvido para otimizar operações de delivery. O sistema prioriza velocidade operacional, permitindo atendentes processarem pedidos em 15-30 segundos.

## Características principais:

- Interface ultra-rápida com poucos cliques
- Integração automática com Bling ERP (fiscal)
- Notificação WhatsApp para entregadores via IzChat
- Sistema de pagamentos com múltiplas modalidades
- Gestão de status de entrega com cores visuais

## 2. ARQUITETURA TÉCNICA

### Stack Tecnológico:

Componente	Tecnologia	URL/Binding
Backend	Cloudflare Worker (ES Module)	api.moskogas.com.br
Banco de Dados	Cloudflare D1 (SQLite)	moskogas_ops
Storage	Cloudflare R2	moskogas-comprovantes
Frontend	HTML Estático	GitHub Pages
ERP Fiscal	Bling API v3	www.bling.com.br/Api/v3
WhatsApp	IzChat API	chatapi.izchat.com.br

### Secrets Cloudflare:

- BLING\_CLIENT\_ID
- BLING\_CLIENT\_SECRET
- IZCHAT\_TOKEN
- APP\_API\_KEY (protege endpoints internos)

### 3. SCHEMA DO BANCO DE DADOS (D1)

#### Tabela: orders (pedidos)

```
id INTEGER PRIMARY KEY
phone_digits TEXT
customer_name TEXT
address_line TEXT
bairro TEXT
complemento TEXT
referencia TEXT
items_json TEXT
total_value REAL
notes TEXT
status TEXT (novo|encaminhado|whatsapp_enviado|entregue|cancelado)
sync_status TEXT (pending|synced)
driver_name_cache TEXT
created_at INTEGER
bling_pedido_id INTEGER (ID interno do Bling)
bling_pedido_num INTEGER (número visível no Bling)
tipo_pagamento TEXT (dinheiro|pix_vista|pix_receber|mensalista|boleto)
pago INTEGER DEFAULT 0 (0=não pago, 1=pago)
```

#### Tabela: customers\_cache

```
phone_digits TEXT PRIMARY KEY
name TEXT
address_line TEXT
bairro TEXT
complemento TEXT
referencia TEXT
bling_contact_id INTEGER
updated_at INTEGER
```

#### Tabela: bling\_tokens

```
id INTEGER PRIMARY KEY (sempre 1)
access_token TEXT
refresh_token TEXT
expires_in INTEGER
obtained_at INTEGER (timestamp Unix)
```

## 4. ENDPOINTS DA API

Método	Endpoint	Descrição
GET	/health	Health check do Worker
GET	/bling/ping	Testa conexão Bling
GET	/bling/oauth/start	Inicia fluxo OAuth Bling
GET	/bling/oauth/callback	Callback OAuth Bling
POST	/api/order/create	Criar novo pedido
GET	/api/pagamentos	Listar pedidos não pagos
PATCH	/api/pagamentos/:id	Marcar pedido como pago
POST	/izchat/notificar-entrega	Enviar WhatsApp entregador
GET	/izchat/teste	Testar envio WhatsApp

### Endpoints Debug (público, sem auth):

- GET /api/pub/bling-status
- GET /api/pub/test-criar-pedido?order\_id=X
- GET /api/pub/test-nfce?bling\_pedido\_id=Y

## 5. SISTEMA DE PAGAMENTOS (v2.7.0)

O sistema de pagamentos controla quando criar vendas no Bling e quando marcar como pago.

Tipo Pagamento	Cria Bling?	Marca Pago?	Aparece Pagamentos?
■ Dinheiro	SIM	SIM	NÃO
■ PIX à vista	SIM	SIM	NÃO
■ PIX a receber	SIM	NÃO	SIM
■ Mensalista	NÃO	NÃO	SIM
■ Boleto/Órgão	NÃO	NÃO	SIM

### Lógica no Worker:

```
const criarBling = ['dinheiro', 'pix_vista',  
'pix_receber'].includes(tipo_pagamento);  
const pago = ['dinheiro', 'pix_vista'].includes(tipo_pagamento) ? 1 : 0;  
  
if (criarBling) {  
  // Cria pedido no Bling via API  
  const result = await criarPedidoBling(env, orderId, orderData);  
  await DB.prepare('UPDATE orders SET bling_pedido_id=?, pago=? WHERE id=?')  
    .bind(result.bling_pedido_id, pago, orderId).run();  
}
```

## 6. INTEGRAÇÃO BLING ERP

Base URL: <https://www.bling.com.br/Api/v3>

### IDs Importantes:

Entidade	ID	Uso
Consumidor Final	726746364	Cliente padrão para NFCe
Forma Pgto: Dinheiro	23368	Parcelas em dinheiro
Forma Pgto: PIX	23465	Parcelas PIX
Forma Pgto: Débito	23369	Parcelas débito
Forma Pgto: Crédito	23370	Parcelas crédito
Forma Pgto: Fiado	23373	Parcelas fiado

### IMPORTANTE: NFCe e Bling API v3

- ENDPOINT /nfces NÃO EXISTE na API v3 do Bling. NFCe não pode ser emitida via API.
- SOLUÇÃO: Sistema cria pedido de venda via API. Operador emite NFCe manualmente no painel Bling 1x/dia em lote (Vendas > NFC-e > Emitir NFCes selecionadas).

### Token OAuth:

- Expira em 6 horas
- Refresh automático via cron: 0 \*/5 \* \* \* (a cada 5h)
- Renovado quando restam <90 minutos
- Armazenado em D1: bling\_tokens (id=1)

## 7. INTEGRAÇÃO IZCHAT (WhatsApp)

Endpoint: POST <https://chatapi.izchat.com.br/api/messages/send>

Payload de envio:

```
{
  "number": "5567999999999", // Telefone E.164
  "body": "Mensagem com link Google Maps"
}
```

Template da mensagem ao entregador:

```
■ NOVA ENTREGA - MoskoGás
■ Endereço: [endereço completo]
■ Cliente: [nome]
■ Tel: [telefone]
■ Valor: R$ [total]

■ Observação: [obs do operador]

■■ Google Maps: https://maps.google.com/maps/search?api=1&query;=[endereço]
```

## 8. FLUXOS DE NEGÓCIO

### Fluxo 1: Pedido À Vista (Dinheiro/PIX)

1. Atendente cria pedido (tipo\_pagamento='dinheiro' ou 'pix\_vista')
2. Worker cria pedido de venda no Bling via API
3. Worker salva bling\_pedido\_id + pago=1 no D1
4. Pedido NÃO aparece em Pagamentos (já pago)
5. Operador emite NFCe em lote 1x/dia no Bling

### Fluxo 2: PIX a Receber

1. Atendente cria pedido (tipo\_pagamento='pix\_receber')
2. Worker cria pedido de venda no Bling via API
3. Worker salva bling\_pedido\_id + pago=0 no D1
4. Pedido APARECE em Pagamentos (aguardando confirmação PIX)
5. Cliente paga → Operador marca como pago em Pagamentos
6. Pedido some de Pagamentos

### Fluxo 3: Mensalista/Boleto

1. Atendente cria pedido (tipo\_pagamento='mensalista' ou 'boleto')
2. Worker NÃO cria no Bling (bling\_pedido\_id=NULL)
3. Worker salva pago=0 no D1
4. Pedido APARECE em Pagamentos (pendente emissão)
5. Fim do mês: Operador cria NFe mensal agrupada no Bling

## 9. DECISÕES ARQUITETURAIS

**NFCe sem API:** Endpoint /nfces não existe. Emissão manual 1x/dia resolve sem complexidade.

**Webhook descartado:** Webhook NFCe seria complexidade desnecessária. Regra simples: bling\_pedido\_id existe = venda criada.

**Cidade hardcoded:** Sistema opera exclusivamente em Campo Grande/MS. Não exibir campos na UI.

**Token auto-refresh:** Cron 5h mantém token vivo (expira 6h, renova com 1.5h margem).

**Cache de clientes:** customers\_cache evita buscas constantes no Bling.

**IzChat direto:** Integração direta via API (não webhook) simplifica arquitetura.

**Status por cores:** Vermelho/Amarelo/Verde/Azul/Cinza identificam status visualmente.

## 10. ERROS COMUNS E SOLUÇÕES

**Erro: Endpoint /nfces retorna 404**

SOLUÇÃO: NFCE não tem endpoint API. Emitir manualmente no Bling.

**Erro: Versão não atualizada**

SOLUÇÃO: SEMPRE incrementar versão em TODO arquivo editado.

**Erro: Confundir pedido\_num com pedido\_id**

SOLUÇÃO: bling\_pedido\_num = número visível. bling\_pedido\_id = ID interno da API.

**Erro: Form reset quebra com null**

SOLUÇÃO: Sempre usar if (el) el.value = "

**Erro: Token expirado**

SOLUÇÃO: Verificar cron está ativo. Forçar refresh via /bling/oauth/start

**Erro: Cliente não encontrado**

SOLUÇÃO: Sincronizar cache com botão 'Sync Bling' em pedido.html

**Erro: IzChat não envia**

SOLUÇÃO: Verificar IZCHAT\_TOKEN nas secrets Cloudflare

## 11. CHECKLIST DE DEPLOYMENT

- Incrementar versão em TODOS arquivos editados
- Testar localmente: wrangler dev
- Verificar schema D1 atualizado (migrations se necessário)
- Deploy Worker: wrangler deploy
- Upload HTML para GitHub Pages
- Verificar versão visível no badge da página
- Testar endpoints críticos (/health, /bling/ping)
- Testar fluxo completo de pedido
- Verificar cron ativo no Cloudflare Dashboard
- Validar secrets atualizadas (se alteradas)

## 12. CONFIGURAÇÕES IMPORTANTES

### Cloudflare Worker:

- Nome: moskogas-backend-v2
- Bindings: DB=moskogas\_ops, BUCKET=moskogas-comprovantes
- Cron: 0 \*/5 \* \* \* (refresh token Bling)

### GitHub Pages:

- Repositório: moskogas/app
- URL: <https://moskogas.github.io/app/>
- Arquivos: pedido.html, gestao.html, pagamentos.html, nav.html

### Bling OAuth:

- Redirect URI: <https://api.moskogas.com.br/bling/oauth/callback>
- Scopes necessários: pedidos.vendas, contatos

### Contatos:

- Bling Developer: <https://developer.bling.com.br>
- Cloudflare Docs: <https://developers.cloudflare.com>



Documento gerado automaticamente em 16/02/2026  
MoskoGás Backend v2.7.0 | Luis Cesar Mosko