MANUAL TECNICO

MoskoGas Sistema de Gestao

Versao 3.0 - 17/02/2026

Cloudflare Workers + D1 + Bling ERP + IzChat

Mosko Ltda - Campo Grande/MS

INDICE

1. VISAO GERAL DO SISTEMA

O MoskoGas e um sistema web para gestao de pedidos de gas de cozinha e agua mineral, desenvolvido para a Mosko Ltda em Campo Grande/MS. O sistema substitui parcialmente o GLP Master (desktop) e integra com Bling ERP para gestao fiscal e IzChat para envio de mensagens via WhatsApp aos entregadores.

Prioridade absoluta: velocidade operacional (15-30 segundos por pedido), poucos cliques, UX otimizada para atendente. O sistema roda inteiramente na nuvem via Cloudflare Workers com banco D1 e frontend em GitHub Pages.

Funcionalidades Principais

- Cadastro rapido de pedidos com busca de cliente por telefone
- Integracao automatica com Bling ERP (vendas, contatos, NFCe)
- Envio de notificacao WhatsApp para entregadores via IzChat
- Gestao de pagamentos com 5 tipos (dinheiro, PIX, mensalista, boleto)
- Sistema de autenticacao com 3 niveis de acesso (Admin, Operador, Entregador)
- Gestao de usuarios com ativacao/desativacao
- Resumo de produtos vendidos no painel de gestao
- Impressao de recibos A4 (2 vias)
- Relatorios de vendas por periodo
- Configuracao de ruas e bairros via OpenStreetMap

2. INFRAESTRUTURA E STACK

Componente	Tecnologia	Detalhes
Backend	Cloudflare Worker (ES Module)	api.moskogas.com.br
Banco de Dados	Cloudflare D1 (SQLite)	moskogas_ops (binding DB)
Storage	Cloudflare R2	moskogas-comprovantes (binding BUCKET)
Frontend	GitHub Pages + HTML/JS	moskogas-app.pages.dev
ERP	Bling v3 API	OAuth 2.0 + JWT
WhatsApp	IzChat API	Notificacoes para entregadores
Repositorio	GitHub	github.com/luismosko/moskogas-app

Secrets Cloudflare (wrangler secret)

```
BLING_CLIENT_ID, BLING_CLIENT_SECRET, IZCHAT_TOKEN, APP_API_KEY, JWT_SECRET
```

Cidade padrao: Campo Grande/MS (hardcoded — NAO exibir campos cidade/UF na UI)

3. AUTENTICACAO E SEGURANCA

O sistema possui autenticacao baseada em JWT com 3 niveis de acesso. Todas as paginas (exceto login.html) verificam sessao e redirecionam para login se nao autenticado.

Role	Acesso	Paginas
Admin	Acesso total	Todas + usuarios.html
Operador	Pedidos e gestao	pedido, gestao, pagamentos, relatorio, config
Entregador	Apenas entregas	entregador.html

Fluxo de Autenticacao

1. Usuario acessa index.html que verifica token no localStorage. 2. Se nao tem token, redireciona para login.html. 3. Login envia POST /auth/login com username+password. 4. Worker valida com bcrypt e retorna JWT (expira em 24h). 5. Token e dados do usuario salvos no localStorage (mg_session_token, mg_user). 6. Cada pagina usa shared.js para verificar sessao e incluir token nos requests. 7. Worker valida JWT em todos os endpoints (exceto /auth/login, /health, /api/pub/*).

Tabela D1: users

```
id INTEGER PRIMARY KEY, username TEXT UNIQUE, password_hash TEXT, display_name TEXT, role TEXT (admin/operador/entregador), active INTEGER DEFAULT 1, created_at TEXT
```

4. ARQUIVOS E VERSOES ATUAIS

Estado em 17/02/2026:

Arquivo	Versao	Funcao
pedido.html	v2.7.4	Insercao de pedido (atendente)
gestao.html	v2.5	Gestao de pedidos + resumo produtos (admin)
pagamentos.html	v1.3.0	Gestao de pagamentos pendentes
config.html	v2.2.0	Configuracao (ruas, bairros, produtos)
relatorio.html	v1.1.0	Relatorios de vendas
entregador.html	s/v	Painel do entregador
print.html	s/v	Impressao recibo A4 (2 vias)
login.html	v1.0.0	Tela de login
index.html	-	Redirect por role (entry point)
nav.html	v1.0.0	Navegacao (legado)
usuarios.html	v1.2.0	Gestao de usuarios (admin)
shared.js	v1.3.0	Utilitarios compartilhados (auth, api, toast)
worker.js	v2.8.0+	Backend Cloudflare Worker (deploy via wrangler)

REGRA: Worker.js e deployado via wrangler (nao vai no GitHub Pages). Os HTMLs vao no repo github.com/luismosko/moskogas-app.

5. SCHEMA D1 (BANCO DE DADOS)

Tabela: orders

```
id INTEGER PRIMARY KEY AUTOINCREMENT, 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 DEFAULT 'novo', sync_status TEXT DEFAULT 'pending', driver_name_cache TEXT, created_at TEXT, bling_pedido_id TEXT, bling_pedido_num TEXT, tipo_pagamento TEXT DEFAULT 'dinheiro', pago INTEGER DEFAULT 0, vendedor TEXT
```

Tabela: customers_cache

```
phone_digits TEXT PRIMARY KEY, name TEXT, address_line TEXT, bairro TEXT, complemento TEXT, referencia TEXT, bling_contact_id TEXT
```

Tabela: bling_tokens

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

Tabela: users

```
id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT UNIQUE NOT NULL, password_hash TEXT NOT NULL, display_name TEXT, role TEXT DEFAULT 'operador', active INTEGER DEFAULT 1, created_at TEXT DEFAULT (datetime('now'))
```

Outras tabelas: streets_cache, neighborhoods_cache, drivers, products, config

6. ENDPOINTS DA API (Worker.js)

Base URL: https://api.moskogas.com.br

Saude e Diagnostico

Metodo	Endpoint	Descricao
GET	/health	Health check do worker
GET	/bling/ping	Testa conexao com Bling
GET	/api/bling/diagnostico	Lista depositos e formas pgto Bling
GET	/api/pub/*	Endpoints debug (sem auth)

Autenticacao

Metodo	Endpoint	Descricao
POST	/auth/login	Login (username + password) retorna JWT
POST	/auth/logout	Logout (invalida sessao)
GET	/auth/me	Retorna dados do usuario logado
GET	/usuarios	Lista usuarios (admin only)
POST	/usuarios	Cria usuario (admin only)
PATCH	/usuarios/:id	Edita usuario (admin only)

Pedidos

Metodo	Endpoint	Descricao
POST	/api/order/create	Cria pedido (com logica Bling condicional)
GET	/api/orders	Lista pedidos do dia
PATCH	/api/order/:id/update	Edita pedido existente
PATCH	/api/order/:id/status	Atualiza status do pedido
PATCH	/api/order/:id/select-driver	Atribui entregador
PATCH	/api/order/:id/cancel	Cancela pedido

Pagamentos

Metodo	Endpoint	Descricao
GET	/api/pagamentos	Lista pedidos com pago=0

PATCH	/api/pagamentos/:id	Marca como pago (cria Bling se necessario)
POST	/api/pagamentos/gerar-nfe	Gera NFe agrupada (batch)

Bling OAuth

Metodo	Endpoint	Descricao
GET	/bling/oauth/start	Inicia fluxo OAuth
GET	/bling/oauth/callback	Callback OAuth
CRON	0 */5 * * *	Refresh token automatico (cada 5h)

IzChat / WhatsApp

Metodo	Endpoint	Descricao
POST	/izchat/notificar-entrega	Envia WhatsApp ao entregador
GET	/izchat/teste	Testa conexao IzChat

Clientes e Config

Metodo	Endpoint	Descricao
GET	/api/customers/search?phone=	Busca cliente por telefone
GET	/api/drivers	Lista entregadores
GET	/api/products	Lista produtos
GET	/api/streets	Lista ruas do cache
POST	/api/streets/import	Importa ruas do OpenStreetMap

7. SISTEMA DE PAGAMENTOS

Tipo	Cria Bling?	Marca Pago?	Em Pagamentos?
Dinheiro	SIM	SIM	NAO
PIX a vista	SIM	SIM	NAO
PIX a receber	SIM	NAO	SIM
Mensalista	NAO	NAO	SIM
Boleto/Orgao	NAO	NAO	SIM

Logica no Worker (ao criar pedido):

```
const criarBling = ['dinheiro','pix_vista','pix_receber'].includes(tipo_pagamento); const pago = ['dinheiro','pix_vista'].includes(tipo_pagamento) ? 1 : 0;
```

Ao marcar como pago em pagamentos.html (PATCH /api/pagamentos/:id): se o pedido NAO tem bling_pedido_id, o worker cria a venda no Bling automaticamente antes de marcar pago=1. Isso garante que mensalistas e boletos tenham a venda criada no momento certo.

NFCe

IMPORTANTE: NFCe NAO tem endpoint direto na API Bling v3. Pedidos criados via API sao emitidos em lote 1x/dia no painel Bling. NAO implementar webhook de NFCe.

8. STATUS DO PEDIDO

Status	Cor	Significado
NOVO	Vermelho	Sem entregador atribuido
ENCAMINHADO	Amarelo	Entregador escolhido
WHATS ENVIADO	Verde	IzChat confirmou envio
ENTREGUE	Azul	Finalizado
CANCELADO	Cinza	Cancelado

Pedidos entregues e cancelados podem ser editados (sem restricao de status). Entregador pode ser trocado mesmo apos entrega.

9. INTEGRACAO BLING ERP v3

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

Auth: OAuth 2.0 (PKCE) | Headers: Authorization: Bearer {token}, enable-jwt: 1

Docs: https://developer.bling.com.br/home

IDs Importantes

Item	ID Bling
Consumidor Final (contato padrao)	726746364
Forma Pgto: Dinheiro	23368
Forma Pgto: PIX	23465
Forma Pgto: Debito	23369
Forma Pgto: Credito	23370
Forma Pgto: Fiado	23373

Endpoints Bling Utilizados

Endpoint	Uso
POST /pedidos/vendas	Criar pedido de venda
GET /contatos	Buscar clientes por nome/telefone
POST /contatos	Criar novo contato
GET /depositos	Listar depositos de estoque
GET /formas-pagamentos	Listar formas de pagamento

Token Bling

Refresh automatico via cron (0 */5 * * *). Tabela bling_tokens (id=1). Expira em 6h, renova com 1.5h de margem. O pedido.html faz check a cada 60s e auto-recovery invisivel se o token estiver expirado.

10. INTEGRACAO IZCHAT (WhatsApp)

O IzChat envia mensagens WhatsApp para entregadores quando um pedido e encaminhado. A mensagem inclui dados do pedido + link Google Maps com o endereco.

Payload:

```
POST /izchat/notificar-entrega { "order_id": 123, "driver_phone": "5567999999999",
"message": "texto da mensagem", "observacao": "obs adicional" }
```

11. FRONTEND - TELAS DO SISTEMA

Tela	Arquivo	Funcao
Login	login.html	Autenticacao do usuario
Entry Point	index.html	Redirect por role apos login
Novo Pedido	pedido.html	Insercao rapida de pedidos (atendente)
Gestao	gestao.html	Painel completo de pedidos + resumo produtos
Pagamentos	pagamentos.html	Gestao de pagamentos pendentes
Entregador	entregador.html	Painel simplificado para entregador
Impressao	print.html	Recibo A4 em 2 vias
Relatorio	relatorio.html	Relatorios de vendas por periodo
Config	config.html	Ruas, bairros, produtos, entregadores
Usuarios	usuarios.html	Gestao de usuarios (admin only)
Navegacao	nav.html	Menu de navegacao (legado)

Biblioteca Compartilhada: shared.js

Contem funcoes utilitarias: api() para requests com auth, getSessionToken(), getCurrentUser(), showToast() com animacao, verificacao de sessao, e constantes. Todas as paginas incluem shared.js para consistencia.

12. REGRAS DE UX

Modais NUNCA fecham ao clicar fora

So fecham por botao X, Cancelar ou Salvar. Implementado em shared.js.

Toasts grandes e visiveis

Fundo colorido, texto grande, animacao slide-in. Duracao padrao 3s.

Tooltips em todos os botoes de acao

Atributo title descritivo para UX. Ex: 'Enviar WhatsApp para entregador'.

Redirect apos salvar pedido

Apos sucesso, toast 1.2s e redireciona automaticamente para gestao.html.

Consumidor Final sem endereco obrigatorio

Ao editar, pula validacao de endereco se nome contem 'CONSUMIDOR FINAL'.

Versao visivel em todas as paginas

Badge com versao no toolbar. Essencial para verificar cache.

Check Bling silencioso

A cada 60s, pedido.html verifica token Bling com auto-recovery invisivel.

Cidade hardcoded

Campo Grande/MS. NAO exibir campos cidade/UF na interface.

13. DEPLOY E VERSIONAMENTO

Regras de Versionamento (OBRIGATORIO)

```
SEMPRE incrementar versao em TODO arquivo editado: HTML: badge visivel + title + h1 JS
(worker.js): comentario // vX.Y.Z no topo NUNCA entregar sem versao atualizada
```

Workflow de Deploy

1. Incrementar versao em TODOS arquivos editados
2. Testar localmente: wrangler dev (worker) / Live Server (HTML)
3. Deploy worker: wrangler deploy
4. Git push HTMLs para GitHub (Claude faz via HTTPS+token)
5. GitHub Pages deploya automaticamente
6. Verificar versao visivel no badge de cada pagina

Git Push (Claude)

Claude faz push direto via HTTPS+token (ghp_xxx). Token solicitado no inicio de cada sessao. Arquivos tambem salvos em /mnt/user-data/outputs/ como backup.

14. ERROS CONHECIDOS (NAO REPETIR)

Usar endpoint /nfce ou /nfces

NAO EXISTE na API Bling v3. NFCe e emitida em lote no painel.

Esquecer de incrementar versao

SEMPRE atualizar em todos os 3 lugares do HTML (title, h1, badge).

Usar pedido_numero ao inves de bling_pedido_id

ID interno != numero visivel. Usar sempre o ID.

Criar webhook de NFCe

Complexidade desnecessaria. Emissao em lote e suficiente.

Form reset sem null check

Sempre usar: if (el) el.value = ''

Modal fecha ao clicar fora

Viola regra UX. Usar shared.js para prevenir.

Token Bling expirado sem recovery

Implementar check silencioso com auto-refresh.

Versao em 1 lugar so

HTML deve ter versao em title, h1 E badge — todos sincronizados.

Nao usar parseInt/parseFloat

items_json pode ter strings. Sempre converter antes de somar.

15. PROXIMOS PASSOS

Versionar arquivos sem versao

entregador.html, print.html e index.html precisam de badge de versao.

AI para contas a pagar

App que interpreta fotos de recibos/notas e lanca automaticamente no Bling.

Melhorias no relatorio

Graficos, filtros por entregador, exportacao Excel.

PWA / Offline

Service worker para funcionar offline (entregador em area sem sinal).

Integracao completa GLP Master

Bridge MySQL local <-> D1/Bling para sincronizacao bidirecional.
