API de Integração Bancastro — PIX e Boleto

Versão do documento: 0.7 · **Data:** 13/11/2026

Documentação técnica da API da **Bancastro** para sistemas e integradores externos que desejam operar pagamentos e recebimentos via **PIX** e **boleto** na conta de um correntista.

> A API descrita aqui é a mesma que os aplicativos oficiais (web e mobile) usam.

1. Visão geral

Item	Valor
Protocolo	HTTPS, REST, JSON (UTF-8)
URL base	`https://api.bancastro.com.br`
Autenticação	**Chave de API** no header `x-api-key` (recomendada) ou JWT no header `x-access-token` (ver seção 3)
Valores monetários na **requisição**	**Reais**, número decimal (ex.: `10.50`) — exceto emissão de boleto (centavos, ver 9.1)
Valores monetários na **resposta**	**Centavos**, inteiro (`amount`) e, quando houver, também em reais (`amount_brl`)
Datas	ISO-8601 em UTC (ex.: `2026-11-11T14:03:00.000Z`)
Idempotência	Campo `idempotency_key` no corpo ou header `Idempotency-Key` (seção 5)

1.1. Contas do correntista

Cada correntista tem uma **Conta Corrente** (saldo interno, chamada de *ledger* nesta documentação). É nela que acontecem os recebimentos PIX, saques PIX, pagamentos de copia-e-cola e pagamentos de boleto descritos aqui.

Alguns clientes habilitados para **multiconta** também têm uma **Conta Pagamento**, que só transfere para a mesma titularidade ou paga boleto. Ela não é o foco desta API. Use `GET /me/accounts` para saber quais contas existem.

2. Fluxo rápido (resumo)

1. Peça à Bancastro uma **chave de API** (`wlk_...`) e envie-a no header `x-api-key` em todas as chamadas (seção 3.1). Alternativa: `POST /login` (e `POST /login2fa` se o usuário tiver 2FA) → recebe o token, enviado em `x-access-token`.
2. Por chave de API **não há 2FA**: as saídas de dinheiro não pedem o `code` (TOTP) (seção 3.6).
3. **Receber via PIX:** `POST /pix/charge` → mostra o QR/copia-e-cola ao pagador → consulta `GET /pix/charge/{id}` até `status = PAID`.
4. **Pagar via PIX (chave):** `GET /pix/consultar/{chave}/{tipo}` → `POST /pix/withdraw` → consulta `GET /pix/withdraw/{id}` até `SUCCESS`.

5. **Pagar PIX copia-e-cola:** `POST /pix/pay-code/preview` → `POST /pix/pay-code` → `GET /pix/withdraw/{id}`.
6. **Pagar boleto:** `POST /bill/simulate` → `POST /boleto/pay` → `GET /pix/withdraw/{id}`.
7. **Emitir boleto:** `POST /boleto` → `GET /billing/boleto`.

---

3. Autenticação e autorização

3.1. Autenticação por chave de API (recomendada)

Para integrações entre sistemas, use a **chave de API**: uma credencial própria da conta, que dispensa login e senha e pode ser revogada sem trocar a senha.

```
x-api-key: wlk_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

- **Formato:** `wlk_` seguido de 43 caracteres (base64url), 47 no total.
- **Quem emite:** a equipe da Bancastro, pelo backoffice, a pedido do correntista. A conta precisa ter o **2FA ativo** para a Bancastro emitir a chave (regra só de emissão; o integrador nunca envia o código). Cada conta pode ter até **5 chaves ativas** ao mesmo tempo (ex.: uma por sistema).
- **Sem expiração:** a chave vale até ser revogada. Guarde-a em cofre de segredos (seção 11).
- **Revogação imediata:** pedida à Bancastro, vale a partir da chamada seguinte. Conta desativada ou que deixou de ser elegível também para de aceitar a chave.
- **Prioridade:** se a requisição trouxer `x-access-token` **e** `x-api-key`, o JWT vale e a chave é ignorada.
- A conta de origem/destino é sempre a dona da chave, como no token (seção 3.5).

Rotas liberadas para chave (as 17 desta documentação; qualquer outra rota responde `403 API_KEY_ROUTE_NOT_ALLOWED`):

#	Método e rota	Seção
1	`GET /me/account`	6.1
2	`GET /me/accounts`	6.2
3	`GET /me/transactions`	6.3
4	`POST /pix/charge`	7.1
5	`GET /pix/charge/{id}`	7.2
6	`GET /pix/consultar/{chave}/{tipo}`	8.1
7	`POST /pix/withdraw`	8.2
8	`GET /pix/withdraw/{id}`	8.3
9	`GET /pix/withdraw/{id}/comprovante`	8.4
10	`POST /pix/pay-code/preview`	8.5
11	`POST /pix/pay-code`	8.5
12	`POST /boleto`	9.1
13	`GET /billing/boleto`	9.2
14	`POST /billing/boleto/email/{id}`	9.3
15	`POST /bill/simulate`	9.4
16	`POST /boleto/pay`	9.5
17	`GET /me/boleto/{id}/comprovante`	9.5

Sem 2FA nas saídas de dinheiro: `POST /pix/withdraw`, `POST /pix/pay-code` e `POST /boleto/pay` **não exigem** o campo `code` (TOTP) quando chamadas por chave; se enviado, é ignorado (seção 3.6).

Segurança: por chave, **a chave sozinha movimenta dinheiro**: não há senha, 2FA nem restrição por IP por trás da chamada. Guarde-a em cofre de segredos, use **uma chave por sistema** e, em caso de suspeita de vazamento, peça à Bancastro a **revogação imediata**. A proteção adicional é o limite de tentativas por chave (seção 10.1).

Erros de autenticação por chave (tabela completa na seção 4.2):

HTTP	code	Quando
401	`API_KEY_INVALID`	Chave mal formada, inexistente ou revogada, ou conta que não pode usar chave (resposta idêntica nos casos, de propósito)
403	`API_KEY_ROUTE_NOT_ALLOWED`	Rota fora da lista acima
429	`RATE_LIMITED`	Mais de 20 chamadas às 3 saídas com a mesma chave em 5 minutos, contadas no início de cada chamada (header `Retry-After`). Ver seção 10.1

3.2. Login com usuário e senha (alternativa)

Alternativa à chave de API: o integrador autentica **em nome de um correntista**, com o usuário e a senha dele, e usa o token JWT resultante (mesmo fluxo dos aplicativos oficiais). Se optar por ele, crie um usuário dedicado à integração. Para sistemas, prefira a chave (seção 3.1): não expira em 63 minutos e não exige guardar a senha.

3.3. `POST /login`

```
{ "user": "usuario.integracao", "password": "********" }
```

A resposta depende da configuração de 2FA (autenticação em dois fatores) do usuário:

a) Usuário sem 2FA — token emitido direto (`200`):

```
{
  "status": "login  efetuado com sucesso!",
  "x-access-token": "eyJhbGciOi...",
  "email": "financeiro@empresa.com",
  "role": "user",
  "name": "Empresa X LTDA",
  "cpfCnpj": "12345678000199",
  "A2fa_Ativada": false,
  "required_2fa": false
}
```

b) Usuário com 2FA já configurado (`200`, sem token) — seguir para `POST /login2fa`:

```
{ "status": "Prossiga com autenticação de dois fatores", "A2fa_Ativada": true, "required_2fa": tr
ue }
```

c) Usuário com 2FA obrigatório e ainda não configurado (`200`, sem token) — a resposta traz `new_secret` (base32) e `qrCode` (imagem data-URL) para cadastrar num app autenticador (TOTP). Após cadastrar, seguir para `POST /login2fa`.

Erros: `400` campos faltando · `401` usuário ou senha incorretos · `404` usuário não encontrado.

3.4. `POST /login2fa`

```
{ "user": "usuario.integracao", "password": "********", "code": "123456" }
```

`code` é o código TOTP de 6 dígitos do app autenticador. Resposta `200` com `"x-access-token"`. Erros: `400` campos faltando · `401` senha ou código inválidos.

3.5. Uso do token (login)

Vale apenas para quem autentica por login (seção 3.2); com chave de API não há token.

```
x-access-token: eyJhbGciOi...
```

- Validade: **~63 minutos** (3800 s). Não há refresh token: ao receber `401` (`{"error": "nao autenticado"}`), faça login de novo.
- O token identifica o correntista. **A conta de origem/destino de toda operação é sempre derivada do token** — campos como `sub_account_id` no corpo são ignorados.
- Usuários com perfil `viewer` (somente leitura) recebem `403 sem permissao` em qualquer operação que movimente dinheiro.

3.6. 2FA em operações

As saídas de dinheiro (`POST /pix/withdraw`, `POST /pix/pay-code`, `POST /boleto/pay`) aceitam o campo `code` (TOTP de 6 dígitos) no corpo. A regra depende de como a chamada foi autenticada:

Endpoint	Por login (JWT, usuário com 2FA)	Por chave de API
`POST /boleto/pay`	**Obrigatório**	Não exigido (`code` ignorado)
`POST /pix/withdraw`	Conferido se enviado ⚠	Não exigido (`code` ignorado)
`POST /pix/pay-code`	Conferido se enviado ⚠	Não exigido (`code` ignorado)

- **Por chave de API:** não há 2FA. O `code` não é necessário e, se enviado, é ignorado (não gera erro). Não existe `2FA_REQUIRED` para chamadas por chave. A proteção é o limite de **20 tentativas de saída a cada 5 min por chave** (`429 RATE_LIMITED` com `Retry-After`, seção 10.1), a revogação imediata da chave e a guarda dela em cofre (seção 3.1). A conta precisa ter 2FA ativo apenas para a Bancastro **emitir** a chave.
- **Por login:** código enviado e inválido → `401 {"error": "Codigo inválido"}`, nas três rotas. ⚠ Em `POST /pix/withdraw` e `POST /pix/pay-code` o código ainda é opcional (período de transição). Ele passará a ser **obrigatório**, e a ausência retornará `401` com `code: "2FA_REQUIRED"`. **Envie o `code` desde já** em todas as saídas de usuários com 2FA. Usuário sem 2FA: o campo é ignorado.

Por login, o TOTP muda a cada 30 s e o segredo do autenticador é o do próprio correntista, cadastrado por ele ao ativar o 2FA (seção 3.3 c).

---

4. Padrão de respostas e erros

Erros retornam JSON com `error` (mensagem legível) e, quando aplicável, `code` (identificador estável para tratar no código):

```
{ "error": "Este boleto já foi pago.", "code": "BOLETO_JA_PAGO" }
```

⚠ Endpoints legados podem devolver o texto em `status` ou `erro` em vez de `error` (login), a emissão de boleto (`POST /boleto`) devolve o texto em `description` (seção 9.1) e nem todos os erros trazem `code`. Trate sempre pelo **status HTTP** primeiro e use `code` quando presente.

4.1. Status HTTP

HTTP	Significado
200	Sucesso (ou reenvio idempotente de operação já aceita)
201	Recurso criado (cobrança PIX)
202	Operação aceita e em processamento (saque/pagamento) — acompanhe pelo `id`
400	Requisição inválida / regra de negócio (saldo, chave, boleto)
401	Não autenticado, token expirado, chave de API inválida, senha ou código 2FA inválido
403	Sem permissão (perfil, segregação, restrição da conta, rota fora da chave de API)
404	Recurso não encontrado
409	Conflito de estado (ex.: boleto já pago, comprovante antes da conclusão)
423	Operação bloqueada temporariamente (trava operacional)
429	Limite de requisições excedido — aguarde o tempo do header `Retry-After` (seção 10.1)
500 / 502	Erro interno / falha no provedor — pode tentar de novo com a **mesma** `idempotency_key`. **Exceção:** em `POST /boleto` (emitir boleto) o `502` **não** deve ser repetido às cegas (seção 9.1)

4.2. Códigos de erro (`code`)

code	HTTP	Quando
`INSUFFICIENT_FUNDS`	400	Saldo disponível insuficiente (inclui tarifas a cobrar)
`PIX_KEY_NOT_FOUND`	400	Chave PIX não existe no DICT
`VALIDATION_ERROR`	400	Dado inválido detectado no domínio
`BOLETO_INVALIDO`	400	Código de barras / linha digitável inválidos
`BOLETO_VENCIDO`	400	Boleto passou do prazo de pagamento
`BOLETO_NAO_REGISTRADO`	400	Boleto ainda não registrado na CIP
`BOLETO_NAO_ENCONTRADO`	400	Boleto não encontrado
`BOLETO_JA_PAGO`	409	Boleto já pago
`SEGREGATION_VIOLATION`	403	Conta mãe tentando usar saldo de clientes
`CONTA_PAGAMENTO_RESTRITA`	403	Operação não permitida na Conta Pagamento (multiconta)
`IP_BLOQUEADO`	403	IP bloqueado temporariamente por excesso de violações de limite
`API_KEY_INVALID`	401	Chave de API mal formada, inexistente, revogada, ou conta sem direito de usar chave
`API_KEY_ROUTE_NOT_ALLOWED`	403	Rota fora das liberadas para chave de API (17 — seção 3.1)
`2FA_REQUIRED`	401	Só por login: código 2FA não enviado em saque PIX / copia-e-cola que o exige (quando passar a ser obrigatório). Nunca ocorre por chave de API. Em `POST /boleto/pay` por login, `code` ausente ou errado retorna `401` `{"error": "Codigo inválido"}` sem `code`
`RATE_LIMITED`	429	Limite de requisições excedido
`NOT_FOUND`	404	Recurso não encontrado
`CASHOUT_BLOCKED` / `OPERATION_BLOCKED`	423	Saídas de dinheiro bloqueadas temporariamente pela operação

Erros de validação de corpo retornam `400` com a primeira mensagem do campo inválido, ex.: `{"error": "amount deve ser > 0"}`.

---

5. Idempotência

Saques PIX, pagamentos copia-e-cola e pagamentos de boleto aceitam uma chave de idempotência (até 191 caracteres), recomendada como UUID v4:

- no corpo: `"idempotency_key": "b1f1c7c2-..."`, **ou**
- no header: `Idempotency-Key: b1f1c7c2-...` (o corpo tem prioridade).

Reenviar a mesma chave **não** cria uma nova operação: a resposta é `200` com `"duplicate": true` e o `id` da operação original. Sem chave, cada requisição é uma operação nova. **Sempre envie uma chave** e reutilize-a em retentativas por timeout ou erro 5xx.

---

6. Conta e extrato

6.1. `GET /me/account` — saldo da Conta Corrente

```
{
  "sub_account_id": 1024,
  "status": "ACTIVE",
  "nome": "Empresa X LTDA",
  "cpf_cnpj": "12345678000199",
  "available_balance": 150000,
  "reserved_balance": 2500,
  "total_balance": 152500,
  "available_balance_brl": 1500.00,
  "reserved_balance_brl": 25.00,
  "total_balance_brl": 1525.00
}
```

- `available_balance`: pode ser usado agora.
- `reserved_balance`: reservado para saídas em processamento.
- `status`: `ACTIVE` | `BLOCKED` | `CLOSED`.

6.2. `GET /me/accounts` — contas do correntista

Retorna `multiconta` (boolean) e `accounts[]` com `type` (`CORRENTE` | `PAGAMENTO`), saldos e `capabilities` (o que cada conta pode fazer).

6.3. `GET /me/transactions` — extrato da Conta Corrente

Query string (todos opcionais):

Parâmetro	Tipo	Descrição
`type`	enum	`CASH_IN`, `CASH_OUT`, `RESERVE`, `RELEASE`, `SETTLEMENT`, `REVERSAL`, `ADJUSTMENT`, `FEE`, `OPENING_BALANCE`, `MIGRATION`
`from`, `to`	data	`YYYY-MM-DD` ou ISO
`page`	int	padrão 1
`page_size`	int	padrão 20, máx. 100

```
{
  "items": [
    {
      "id": 88123,
      "type": "CASH_IN",
      "direction": "CREDIT",
      "amount": 5000,
      "amount_brl": 50.00,
      "balance_before": 145000,
      "balance_after": 150000,
      "reference_type": "pix_charge",
      "reference_id": "3021",
      "description": "Recebimento PIX",
      "created_at": "2026-11-11T14:03:00.000Z"
    }
  ],
  "page": 1,
  "page_size": 20,
  "total": 57
}
```

`direction`: `CREDIT` (entrada) | `DEBIT` (saída). Lançamentos de saída podem trazer dados do destino da operação.

---

7. Recebimento via PIX (cobrança com QR Code)

7.1. `POST /pix/charge` — criar cobrança

```
{ "amount": 50.00, "description": "Pedido 9981" }
```

Campo	Tipo	Obrigatório	Regra
`amount`	número (reais)	sim	> 0 e ≤ 1.000.000
`description`	string	não	até 255 caracteres

Resposta `201`:

```
{
  "id": 3021,
  "txid": "a8f3c2...",
  "emv": "00020101021226...6304ABCD",
  "image_base64": "iVBORw0KGgo...",
  "amount": 5000,
  "status": "PENDING",
  "expires_at": "2026-11-12T14:03:00.000Z"
}
```

- `emv`: código **PIX copia-e-cola** para mostrar ao pagador.
- `image_base64`: imagem PNG do QR Code (pode ser `null`).

7.2. `GET /pix/charge/{id}` — consultar cobrança

Resposta `200` com `id`, `txid`, `amount` (centavos), `status`, `emv`, `paid_at`, `expires_at`, `created_at`.

status	Significado
`PENDING`	Aguardando pagamento
`PAID`	Pago — valor já creditado na Conta Corrente
`EXPIRED`	Expirou sem pagamento
`FAILED`	Falhou

Enquanto `PENDING`, esta consulta também confere o pagamento junto ao provedor e credita na hora se já foi pago — não depende só da notificação assíncrona.

Como acompanhar: consulte `GET /pix/charge/{id}` periodicamente (sugestão: a cada 5–10 s nos primeiros minutos, depois espaçar) até sair de `PENDING`.

7.3. Fluxo

1. O integrador cria a cobrança: `POST /pix/charge` (seção 7.1).
2. A API responde com `id` e `emv` (copia-e-cola).
3. O integrador exibe o QR Code / copia-e-cola ao pagador.
4. O pagador paga via PIX no app do banco dele.
5. O integrador consulta `GET /pix/charge/{id}` periodicamente, com intervalo crescente (ex.: a cada 5–10 s nos primeiros minutos, depois espaçando), até `status = PAID` ou outro status final (`EXPIRED`, `FAILED`).

---

8. Pagamentos via PIX

Todas as saídas seguem o mesmo ciclo: a API **reserva** o valor no saldo, envia ao provedor e **liquida** quando confirmado. Se o provedor recusar ou devolver, a reserva é liberada e o valor volta ao saldo.

status (saída)	Significado
`REQUESTED`	Recebida
`RESERVED`	Valor reservado no saldo
`PROCESSING`	Enviada ao provedor
`SUCCESS`	Concluída — comprovante disponível
`FAILED`	Falhou — valor devolvido ao saldo (ver `failure_reason`)
`CANCELED`	Cancelada — valor devolvido ao saldo

`SUCCESS`, `FAILED` e `CANCELED` são finais.

8.1. `GET /pix/consultar/{chave}/{tipo}` — consultar chave (DICT)

Mostra o destinatário antes de pagar. `tipo` aceita: `CPF`, `CNPJ`, `EMAIL`, `TELEFONE`, `CHAVE_ALEATORIA`. Codifique a chave na URL (`encodeURIComponent`).

```
{
  "name": "Fulano de Tal",
  "cpf_cnpj": "***.456.789-**",
  "end2end_id": "E12345678202611111140300abcdef",
  "bank_name": "Banco Exemplo"
}
```

`404` = a chave não existe. Guarde o `end2end_id` para enviar no saque.

8.2. `POST /pix/withdraw` — transferir para chave PIX

```
{
  "amount": 120.00,
  "pix_key": "fulano@email.com",
  "pix_key_type": "EMAIL",
  "end2end_id": "E12345678202611111140300abcdef",
  "code": "123456",
  "idempotency_key": "6c1d2a2e-3b5f-4b51-9a8a-0f6a2f3e9a11"
}
```

Campo	Tipo	Obrigatório	Regra
`amount`	número (reais)	sim	> 0 e ≤ 1.000.000
`pix_key`	string	sim	até 255
`pix_key_type`	enum	sim	`CPF`, `CNPJ`, `EMAIL`, `PHONE`, `EVP`
`end2end_id`	string	não	retornado pela consulta 8.1; evita nova consulta ao DICT
`code`	string	só por login, se o usuário tem 2FA	TOTP do usuário; ignorado por chave de API — ver seção 3.6
`idempotency_key`	string	recomendado	ver seção 5

> Atenção: aqui os tipos são `PHONE` e `EVP` (na consulta 8.1 são `TELEFONE` e `CHAVE_ALEATORIA`).

Resposta `202` (nova) ou `200` (reenvio idempotente):

```
{ "id": 7781, "status": "RESERVED", "duplicate": false }
```

Sem `end2end_id`, a API consulta a chave no DICT e recusa com `400 PIX_KEY_NOT_FOUND` se ela não existir. Tarifas do plano do cliente são debitadas conforme contrato.

8.3. `GET /pix/withdraw/{id}` — acompanhar saída

Serve para **todas** as saídas (chave, copia-e-cola e boleto):

```
{
  "id": 7781,
  "amount": 12000,
  "fee": 0,
  "pix_key": "fulano@email.com",
  "pix_key_type": "EMAIL",
  "status": "SUCCESS",
  "idempotency_key": "6c1d2a2e-...",
  "end2end_id": "E12345678202611111140300abcdef",
  "failure_reason": null,
  "created_at": "2026-11-11T14:05:00.000Z",
  "processed_at": "2026-11-11T14:05:04.000Z"
}
```

8.4. `GET /pix/withdraw/{id}/comprovante` — comprovante

Disponível só com `status = SUCCESS` (antes disso: `409`). Retorna os dados do remetente e do destinatário. Para saída de boleto retorna `{ "tipo": "BOLETO", "items": [...] }`.

8.5. Pagar PIX copia-e-cola (QR Code de terceiros)

`POST /pix/pay-code/preview` — lê o código sem movimentar saldo:

```
{ "emv": "00020101021226...6304ABCD" }
```

```
{
  "amount": 4990,
  "amount_brl": 49.90,
  "fixed_amount": true,
  "receiver_name": "Loja Exemplo",
  "pix_key": "a1b2c3d4-...",
  "txid": "PEDIDO123"
}
```

`POST /pix/pay-code` — paga:

```
{ "emv": "00020101021226...6304ABCD", "amount": 49.90, "code": "123456", "idempotency_key": "..."
}
```

- `emv`: obrigatório, até 512 caracteres.
- `amount` (reais): usado **só** quando o QR é de valor aberto (`fixed_amount = false`). Em QR de valor fixo, vale o valor do código.
- `code`: só por login, se o usuário tem 2FA (TOTP); ignorado por chave de API — ver seção 3.6.

Resposta `202`/`200` igual à 8.2. Acompanhe por `GET /pix/withdraw/{id}`.

---

9. Boletos

9.1. Recebimento — `POST /boleto` (emitir boleto)

```
{
  "amount": 15000,
  "due_date": "2026-11-30",
  "expiration_date": "2026-12-30",
  "payer": {
    "name": "Cliente Pagador",
    "tax_id": "12345678909",
    "address": {
      "street": "Rua Exemplo",
      "number": "100",
      "complement": "Sala 2",
      "district": "Centro",
      "city": "São Paulo",
      "state": "SP",
      "postal_code": "01001000"
    }
  }
}
```

Campo	Regra
`amount`	**centavos** (inteiro). ⚠ Diferente dos demais endpoints, que usam reais
`due_date`	vencimento, `YYYY-MM-DD`
`expiration_date`	data limite de pagamento após o vencimento, `YYYY-MM-DD`
`payer.tax_id`	CPF (11) ou CNPJ (14), só dígitos
`payer.address.*`	`street`, `number`, `district`, `city`, `state` obrigatórios

A resposta traz os dados do boleto no provedor (incluindo `receivables[0].boleto.linha_digitavel`), `codigoDeBarras` (imagem PNG em base64) e `emailSent`. O PDF do boleto é enviado ao e-mail do correntista.

> O boleto já foi gerado mesmo se `emailSent = false`. **Não repita a chamada** (geraria outro boleto); use o reenvio 9.3.

Erros da emissão. Atenção: estas respostas trazem o texto em **`description`** (não em `error`):

HTTP	Corpo	Quando
status do provedor (padrão `502`)	`{ "description": "...", "code": "<código do provedor>", "trace_id": "..." }`	O provedor **recusou** a emissão; nenhum boleto foi criado. Ex.: `payment_method_boleto_not_allowed` → "A emissão de boleto não está habilitada para esta conta. Fale com o suporte." Informe `code` e `trace_id` ao suporte
`502`	`{ "description": "A cobrança foi criada, mas o provedor não devolveu a linha digitável. Confira a lista de boletos antes de gerar outro." }`	A cobrança foi criada, mas sem linha digitável

⚠ **No `502` não repita a chamada às cegas:** na segunda situação a cobrança pode já existir e uma nova emissão cobraria o pagador em duplicidade. Consulte antes `GET /billing/boleto` (9.2) e só emita outro boleto se o anterior não constar na lista.

O valor recebido por boleto é creditado pelo fluxo do provedor (Conta Pagamento), não pela Conta Corrente.

9.2. `GET /billing/boleto` — listar boletos emitidos

Retorna a lista do provedor com situação de cada boleto.

9.3. `POST /billing/boleto/email/{id}` — reenviar boleto por e-mail

9.4. Pagamento — `POST /bill/simulate` (consultar boleto)

```
{ "linhaDigitavel": "34191790010104351004791020150008291070026000" }
```

Consulta o boleto na CIP: valor atualizado (juros/multa/desconto), beneficiário e situação. Retorna o erro de negócio correspondente se o boleto não puder ser pago (`BOLETO_JA_PAGO`, `BOLETO_VENCIDO` etc.).

9.5. `POST /boleto/pay` — pagar boleto

```
{
  "barcode": "34191.79001 01043.510047 91020.150008 2 91070026000",
  "code": "123456",
  "idempotency_key": "..."
}
```

Campo	Regra
`barcode`	linha digitável (47/48) ou código de barras (44), com ou sem pontuação
`code`	só por login, se o usuário tem 2FA (TOTP); ignorado por chave de API
`idempotency_key`	recomendado

O valor não é enviado pelo integrador: a API cobra o valor que o boleto vale hoje na CIP. Resposta `202`/`200`:

```
{ "id": 7790, "status": "RESERVED", "duplicate": false, "amount": 15230, "amount_brl": 152.30 }
```

Acompanhe por `GET /pix/withdraw/{id}`. Comprovante (após `SUCCESS`): `GET /me/boleto/{id}/comprovante` ou `GET /pix/withdraw/{id}/comprovante`.

---

10. Limites e regras gerais

- Valor máximo por operação na API: R$ 1.000.000,00 (limites contratuais e de plano podem ser menores).
- Saídas sujeitas a saldo disponível, tarifas do plano e travas operacionais (`423`).
- Clientes multiconta: saídas abertas pela Conta Pagamento são recusadas com `CONTA_PAGAMENTO_RESTRITA` — use os endpoints desta documentação (Conta Corrente).

10.1. Limites de requisição (rate limit)

Rota	Limite	Contado por
`POST /login`	30 a cada 5 min	IP
`POST /login2fa`	30 a cada 5 min (IP) e 8 a cada 5 min (usuário)	IP e usuário
`POST /pix/withdraw`, `POST /pix/pay-code`	20 a cada 5 min (somados)	usuário da conta (vale também para chamadas por chave de API)
`POST /pix/withdraw`, `POST /pix/pay-code`, `POST /boleto/pay` por chave de API	20 tentativas a cada 5 min (somadas, contadas ao receber a chamada, qualquer resultado)	chave de API

- Excedeu: `429` com `code: "RATE_LIMITED"` e header `Retry-After` (segundos).
- Tentativas de saída por chave: a 21ª chamada às saídas da mesma chave em 5 minutos recebe `429` com `code: "RATE_LIMITED"` e `Retry-After`, mesmo que nenhuma tenha falhado. Vale também para chamadas em paralelo. Consultas, outras chaves e o acesso por login não são afetados.
- Os `429` dos limites por IP e por usuário (login, `POST /login2fa`, limite de saídas por usuário) contam como violação para o IP; o `429` do limite de tentativas de saída **por chave de API não** conta. Com 10 violações em 1 hora, o IP é bloqueado por 1 hora em **todas** as rotas (`403`, `code: "IP_BLOQUEADO"`).
- Os valores acima são o padrão e podem ser ajustados pela Bancastro. Para volumes maiores (ex.: lote de pagamentos), alinhe com a Bancastro antes.

11. Boas práticas de integração

1. Prefira a **chave de API** (seção 3.1) a login e senha. Guarde a chave em cofre de segredos (por chave, ela sozinha movimenta dinheiro); **nunca** no front-end, em aplicativo móvel, em repositório de código ou em log. Use uma chave por sistema, para revogar uma sem afetar as outras; suspeita de vazamento: peça a revogação à Bancastro na hora.
2. Se usar login: usuário dedicado à integração, com 2FA e senha em cofre; renove o token ao receber `401`, sem fazer login a cada requisição.
3. Sempre envie `idempotency_key` em saídas e reutilize-a nas retentativas.
4. Em `202`, **não** considere a operação concluída — acompanhe até um status final.
5. Faça polling com intervalo crescente e limite de tentativas.
6. Ao receber `429`, respeite o `Retry-After`; não insista (em login e nos limites por usuário, insistir pode bloquear o IP).
7. Por chave de API, não envie `code`: não há 2FA nas saídas. Por login, envie o `code` (TOTP) em todas as saídas de usuários com 2FA e **nunca repita em loop** uma saída que voltou `401` por código errado.
8. Trate por status HTTP + `code`; não dependa do texto de `error`.
9. Registre o `id` retornado de cada operação para conciliação com o extrato.

12. Collection Postman

Acompanha este documento:

- `API-Externa.postman_collection.json` — todos os endpoints, com exemplos.
- `API-Externa.postman_environment.json` — variáveis `baseUrl`, `apiKey` (secreta), `user`, `password`, `totp`.

Por padrão os requests usam a chave de API: preencha `apiKey` no ambiente e o header `x-api-key` já vai em todas as chamadas. Para autenticar por login, rode o request **Login** (grava o token em `{{token}}`), ative o header

`x-access-token` e desative o `x-api-key` nos requests. Os requests de criação gravam o `id` retornado (`{{chargeId}}`, `{{payoutId}}`) para as consultas.