SmartCityTec

Integrações realizadas com a SCT através do Data Space provido para o CinCatarina.

- integração FGTS <> SmartCityTec
- integração sefsc <> SmartCityTec

integração FGTS <> SmartCityTec

Ferramenta em Bun para consultar a API. Serve como código de exemplo:

```
#!/usr/bin/env bun
import assert from "node:assert";
import { parseArgs } from "node:util";

function makeBody(contrato: string, cnpj: string): string {
  assert.match(contrato, /^\d+$/);
  assert.match(cnpj, /^\d{14}$/);
  return `
    <?xml version="1.0"?>
    <methodCall>
      <methodName>ConsultaCRF</methodName>
      <params>
        <param><value><i4>${contrato}</i4></value></param>
        <param><value><i4>1</i4></value></param>

<param><value><array><data><value><string>1${cnpj}</string></value></data></array></value></pa
ram>
      </params>
    </methodCall>
  `.trim();
}

function parseDate(eightDigitString: string): Date {
  const match = eightDigitString.match(/^(\d{4})(\d{2})(\d{2})$/);
  assert(match);
  return new Date(...(match.slice(1).map(Number) as [number, number, number]));
}

async function parseXmlResponse(xml: string) {
  const match = xml.match(/<string>1(\d{14})(\d{8})(\d{8})(\d{24})<\/string>/);
```

```
  assert(match);
  const [, cnpj, rawEmissao, rawValidade, certificado] = match;
  return {
    cnpj,
    emissao: parseDate(rawEmissao),
    validade: parseDate(rawValidade),
    certificado,
  };
}

export async function queryCRFbyCNPJ(contrato: string, cnpj: string) {
  const url = "https://webp.caixa.gov.br/empresa/crf/consultalote/server.asp";
  const body = makeBody(contrato, cnpj);
  const response = await fetch(url, { body, method: "POST" });
  const xml = await response.text();
  return parseXmlResponse(xml);
}

if (import.meta.main) {
  try {
    const args = parseArgs({
      options: {
        contrato: { type: "string" },
        cnpj: { type: "string" },
      },
    });
    const { cnpj, contrato } = args.values;
    assert(cnpj, "missing --cnpj= argument");
    assert(contrato, "missing --contrato= argument");
    const result = await queryCRFbyCNPJ(contrato, cnpj);
    console.log(JSON.stringify(result));
  } catch (error) {
    console.error((error as Error).message);
  }
}
```

Sobre isso, eu deixei o "contrato" parametrizável. Meus testes agora não falharam com o valor `66`, mas ontem sim, então convém perguntar à Caixa se precisa utilizar um número especial.

Publiquei no Gitlab como snippet. Só fiz uma alteração pertinente: `contrato` precisa ser `number`.

https://git.ciga.sc.gov.br/-/snippets/33 – *se retornar not found, precisa verificar a permissão*

integração sefsc <> SmartCityTec

Visão Geral do Fluxo

O processo de obtenção do documento segue as seguintes etapas

```
[Início]
   |
   ├── 1. Recupera CPF do Solicitante (Contexto ou Env Var)
   ├── 2. Identifica tipo do contribuinte (CPF vs CNPJ pelo tamanho)
   ├── 3. Envia requisição POST para a API SEF-SC
   ├── 4. Valida a estrutura de retorno (ResultCode, Data)
   ├── 5. Mapeia o Tipo da Certidão (Status e Regras de Validade)
   └── 6. Decodifica o PDF (Base64) e gera o Resultado
```

Configurações e Variáveis de Ambiente

- **URL do Endpoint**: Configurada via `SEF_SC_API_URL` (padrão: `https://sat.sef.sc.gov.br/api/cnd/Certidao/Gerar`).
- **Solicitante Padrão:** `CPFCNPJ_SOLICITANTE_PADRAO`. Utilizado como *fallback* caso nenhum CPF do solicitante seja fornecido no contexto da requisição.

Estrutura de Comunicação da API

Requisição (POST)

O payload exige a identificação do contribuinte consultado e o CPF do solicitante.

```
// Identificacao.Type = "Cnpj" se len == 14, caso contrário "Cpf"
{
  "Identificacao": {
    "Type": "Cnpj",
```

```
    "Value": "12075748000132"
  },
  "CpfSolicitante": {
    "Type": "Cpf",
    "Value": "00000000000"
  }
}
```


Resposta de Sucesso (Exemplo)


```
{
  "ResultCode": "OK",
  "Data": {
    "Numero": "250140126228731",
    "Identificacao": {
      "Type": "Cnpj",
      "Value": "12075748000132"
    },
    "Nome": "CONSORCIO INTERFEDERATIVO SANTA CATARINA",
    "Tipo": "Negativa",
    "DataEmissao": "2025-04-22T15:47:40",
    "DataValidade": "2025-10-19T15:47:40",
    "Situacao": "Valida",
    "OrigemDebitos": null,
    "ArquivoCndPdf": {
      "CND": "250140126228731",
      "Nome": "CND_250140126228731.pdf",
      "Dados": "<Conteudo do arquivo em base64>",
      "ContentType": "application/pdf"
    }
  },
  "Messages": null
}
```


Regras de Negócio e Processamento de Dados

Mapeamento de Status do Documento

A partir do campo `"Tipo"` retornado na resposta, define-se o status interno:

Tipo Retornado (SEF-SC)	Status Interno	Tipo da Certidão	Observação / Mensagem
`Negativa`	APROVADO	NEGATIVA	Sem pendências.
`Positiva`	REPROVADO	POSITIVA	Apresenta pendências na SEF-SC.
`PositivaComEfeitoNegativa`	APROVADO	POSITIVA_NEGATIVA	Certidão positiva com efeito de negativa.
Qualquer outro	PENDENTE	INDEFINIDO	Tipo de certidão não reconhecido.

Tratamento de Datas

- **Timezone**: Se a string de data da API não contiver fuso horário explícito, é assumido o sufixo `-03:00` (horário de Brasília).
- **Ajuste para Certidão Positiva**: Caso a certidão seja classificada como `POSITIVA`, a data de validade é ajustada para o final do mesmo dia da emissão (`23:59:59.999999`).