Teste prático

Leandro Steffens de Oliveira

02/09/2024

Sumário

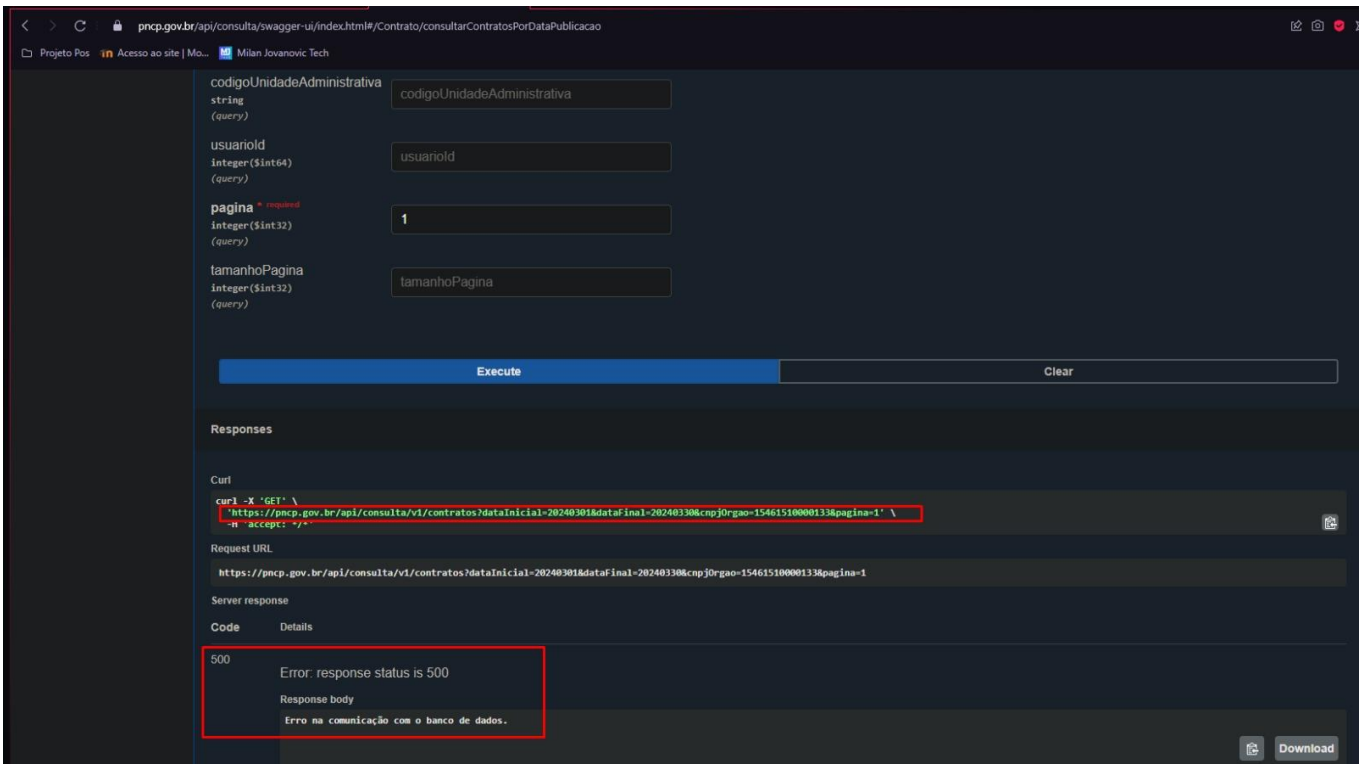
Introdução

Este relatório detalha o desenvolvimento e a implementação de um sistema para visualizar e gerenciar informações sobre contratos públicos utilizando a API REST do Portal Nacional de Contratações Públicas (PNCP) e também ensinará como rodar a aplicação. O objetivo principal deste projeto é criar uma aplicação web que permita aos usuários consultar contratos de um órgão público específico, fornecendo um CNPJ e um intervalo de datas.

A aplicação foi desenvolvida com Node.js e Express e EJS para renderização de páginas dinâmicas. A solução foi projetada para cumprir os seguintes requisitos:

1. **Consumir a API do PNCP**: A aplicação faz uso da API REST para obter contratos de um órgão público, filtrando-os com base no CNPJ e no período especificado.
2. **Exibir Informações Relevantes**: Em uma página web, são mostradas as informações do órgão e uma tabela com os contratos encontrados que atendem aos critérios de data. Para cada contrato, são apresentadas as seguintes informações: data de vigência inicial e final, razão social do fornecedor, objeto do contrato e valor inicial.
3. **Calcular e Exibir o Valor Total**: A aplicação calcula o valor total dos contratos obtidos e o exibe na interface.
4. **Armazenar Dados no Banco de Dados**: As informações coletadas são armazenadas em um banco de dados SQLite, permitindo a persistência e a futura análise dos dados.

Obs: Para facilitar a configuração inicial e estruturar o projeto, utilizei o pacote express-generator. Este pacote cria todos os arquivos essenciais para o desenvolvimento, configurando automaticamente a aplicação para rodar na porta 3000. O SQLite é ideal para este projeto devido à sua simplicidade e conveniência. Ele não requer um servidor separado, o que facilita a configuração e manutenção. Devido à instabilidade da API, vi a necessidade de criar um objeto *simuleResponse* que simula a resposta da API para fins de teste.



Como rodar a aplicação

A priori é necessário ter o Node.js instalado, após instalação basta clonar o repositório https://github.com/LeandroSteffens/pncp_contratos.git ou baixar os arquivos. Abra o terminal, dentro do diretório do da aplicação '*/pncp_contratos/app' digite "npm install" para instalar as dependências do projeto. Após concluir digite "npm start" para iniciar a aplicação. Agora para acessar a aplicação basta digitar este endereço em seu navegador http://localhost:3000/ .

Para testar a funcionalidade da aplicação, inicie o servidor e navegue até a página inicial. Em seguida, preencha o formulário com um CNPJ e um período para buscar contratos. Assim a aplicação responderá exibindo todos os contratos do CNPJ informado dentro da data requisitada.

Evidencia do funcionamento:

Informações do Órgão

CNPJ: 33004540000100

Razão Social: FUNDACAO UNIVERSIDADE FEDERAL DE MATO GROSSO

Contratos

Data de Vigência Inicial	Data de Vigência Final	Razão Social do Fornecedor	Objeto do Contrato	Valor Inicial
2024-03-04	2025-03-04	INOVAWAY TECNOLOGIA LTDA	.	564
2024-03-13	2026-03-13	W.A. EQUIPAMENTOS E SERVICOS LTDA	O objeto do presente instrumento é a contratação de solução de tecnologia da informação e comunicação, consistente na prestação de serviços de outsourcing de impressão, na modalidade fornecimento de equipamentos de impressão com pagamento de páginas impressas. A prestação do serviço abrange o fornecimento de equipamentos, gestão de impressão e bilhetagem, manutenção preventiva e corretiva com fornecimento de consumíveis e componentes (exceto papel), transporte, instalação e configuração dos equipamentos e capacitação para gestão e operação dos equipamentos instalados, para atendimento à demanda da FUFMT.	611201.38

Valor Total dos Contratos: 611765.38

Realizar Nova Busca

CNPJ:

Data Inicial:

dd/mm/aaaa

Data Final:

dd/mm/aaaa

Buscar

Evidência do funcionamento com o simuleResponse:

Informações do Órgão

CNPJ: 12345678000101

Razão Social: Orgao Entidade 0

Contratos

Data de Vigência Inicial	Data de Vigência Final	Razão Social do Fornecedor	Objeto do Contrato	Valor Inicial
2024-09-01	2024-09-01	Fornecedor 0	Objeto do contrato 0	1000
2024-09-01	2024-09-01	Fornecedor 1	Objeto do contrato 1	1100
2024-09-01	2024-09-01	Fornecedor 2	Objeto do contrato 2	1200
2024-09-01	2024-09-01	Fornecedor 3	Objeto do contrato 3	1300
2024-09-01	2024-09-01	Fornecedor 4	Objeto do contrato 4	1400
2024-09-01	2024-09-01	Fornecedor 5	Objeto do contrato 5	1500
2024-09-01	2024-09-01	Fornecedor 6	Objeto do contrato 6	1600
2024-09-01	2024-09-01	Fornecedor 7	Objeto do contrato 7	1700
2024-09-01	2024-09-01	Fornecedor 8	Objeto do contrato 8	1800
2024-09-01	2024-09-01	Fornecedor 9	Objeto do contrato 9	1900

Valor Total dos Contratos: 14500

Realizar Nova Busca

CNPJ:

Data Inicial:

dd/mm/aaaa

Data Final:

dd/mm/aaaa

Buscar

Descrição

db. js

```
// Arquivo responsavel por gerenciar o banco de dados sqlite
const sqlite3 = require("sqlite3").verbose();
const db = new sqlite3.Database("./contracts.db"); // Inicializa o banco de dados de contrato

db.serialize(() => {
  db.run(`CREATE TABLE IF NOT EXISTS contracts (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    cnpj TEXT,
    razaoSocial TEXT,
    dataVigenciaInicio TEXT,
    dataVigenciaFim TEXT,
    nomeRazaoSocialFornecedor TEXT,
    objetoContrato TEXT,
    valorInicial REAL
  )`);
});

module.exports = db;
```

Instância o banco de dados chamado *contracts.db.* Em seguida, define a criação de uma tabela *contracts.*

index. js

```
// Aquivo responsavel por criar pagina inicial na rota /
const express = require('express');
const router = express.Router();

// Rota para pagina inicial
router.get('/', (req, res) => {
  res.render('index');
});

module.exports = router;
```

Utilizei o express para criar rotas, definindo um método GET para a página inicial (/) que renderiza a visualização *index.ejs*.

searchContracts.js

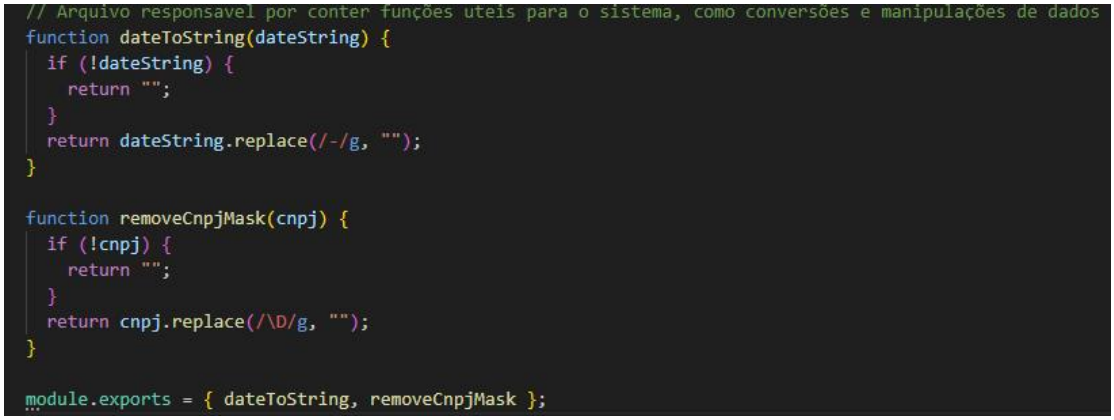
```
// Arquivo responsavel por expor rota por buscar contratos e inserir no banco de dados

// Realiza as importações necessárias
const express = require("express");
const router = express.Router();
const axios = require("axios");
const db = require("../db/db");
const { dateToString } = require("../utils/utils");

// Função para inserir contratos no banco de dados
const insertContractsIntoDB = (contracts) => {
  if (!Array.isArray(contracts)) {
    console.error(
      "Expected contracts to be an array, but got:",
      typeof contracts
    );
    return;
  }
  contracts.forEach((contract) => {
    db.run(
      `INSERT INTO contracts (cnpj, razaoSocial, dataVigenciaInicio, dataVigenciaFim, nomeRazaoSocialFornecedor, objetoContrato, valorInicial) VALUES (?, ?, ?, ?, ?, ?, ?)`,
      [
        contract.orgaoEntidade.cnpj,
        contract.orgaoEntidade.razaoSocial,
        contract.dataVigenciaInicio,
        contract.dataVigenciaFim,
        contract.nomeRazaoSocialFornecedor,
        contract.objetoContrato,
        contract.valorInicial,
      ]
    );
  });
};

// Função para calcular o valor total dos contratos
const calculateTotalValue = (contracts) => {
  return contracts.reduce((sum, contract) => sum + contract.valorInicial, 0);
};
// Função para calcular o valor total dos contratos
const calculateTotalValue = (contracts) => {
  return contracts.reduce((sum, contract) => sum + contract.valorInicial, 0);
};

const makeRequest = async (url, mockResponse) => {
  let response;
  try {
    response = await axios.get(url, { timeout: 5000 });
  } catch (error) {
    response = mockResponse;
  }
  return response;
};
```

```
// Rota para buscar contratos e salvar no banco de dados
router.get("/buscarContratos", async (req, res, next) => {
  const { cnpj, dataInicial, dataFinal } = req.query;
  const url = `https://pncp.gov.br/api/consulta/v1/contratos?dataInicial=${dateToString(
    dataInicial
  )}&dataFinal=${dateToString(dataFinal)}&cnpjOrgao=${cnpj}&pagina=1`;

  try {
    const response = await makeRequest(url, simuleResponse); // Realiza a request para o endpoint da API, simula o resultado em caso de falha
    const contracts = response.data;

    insertContractsIntoDB(contracts); // Chama funcao que realiza a inserção dos contratos no banco de dados
    const totalValue = calculateTotalValue(contracts); // Calcula o valor total dos contratos da busca realizada

    // renderiza na tela a tela de resultados baseado na resposta da API
    res.render("results", {
      orgao: contracts[0].orgaoEntidade,
      contracts: contracts,
      totalValue: totalValue,
    });
  } catch (error) {
    next(error);
  }
});
```

```
// Simule a resposta caso a API não esteja disponível ou devolva algum erro não esperado
const simuleResponse = {
  data: [
    data: Array.from({ length: 10 }, (_, i) => ({
      numeroControlePncpCompra: `12345${i}`,
      codigoPaisFornecedor: "BR",
      orgaoSubRogado: {
        cnpj: `1234567800010${i}`,
        razaoSocial: `Orgao SubRogado ${i}`,
        poderId: "1",
        esferaId: "1",
      },
      orgaoEntidade: {
        cnpj: `1234567800010${i + 1}`,
        razaoSocial: `Orgao Entidade ${i}`,
        poderId: "1",
        esferaId: "1",
      },
      anoContrato: 2023,
      tipoContrato: {
        id: 1,
        nome: `Tipo ${i}`,
      },
      numeroContratoEmpenho: `123456${i}`,
      dataAssinatura: "2024-09-01",
      dataVigenciaInicio: "2024-09-01",
      dataVigenciaFim: "2024-09-01",
      niFornecedor: `12345678${i}`,
      tipoPessoa: "PJ",
      categoriaProcesso: {
        id: 1,
        nome: `Categoria ${i}`,
      },
      dataPublicacaoPncp: "2024-08-31T10:44:08",
      dataAtualizacao: "2024-08-31T10:44:08",
      sequencialContrato: i + 1,
      unidadeOrgao: {
        ufNome: "São Paulo",
        codigoUnidade: `SP0${i}`,
        nomeUnidade: `Unidade ${i}`,
        ufSigla: "SP",
        municipioNome: "São Paulo",
        codigoIbge: "3550308",
      },
      informacaoComplementar: `Informação complementar ${i}`,
      processo: `Processo ${i}`,
      unidadeSubRogada: {
        ufNome: "Rio de Janeiro",
        codigoUnidade: `RJ0${i}`,
        nomeUnidade: `Unidade ${i + 1}`,
        ufSigla: "RJ",
        municipioNome: "Rio de Janeiro",
        codigoIbge: "3304557",
      },
      nomeRazaoSocialFornecedor: `Fornecedor ${i}`,
      niFornecedorSubContratado: `87654321${i}`,
      nomeFornecedorSubContratado: `Fornecedor Subcontratado ${i}`,
      numeroControlePNCP: `54321${i}`,
      receita: true,
      tipoPessoaSubContratada: "PJ",
      objetoContrato: `Objeto do contrato ${i}`,
      valorInicial: 1000 + i * 100,
      numeroParcelas: 10,
      valorParcela: 100,
      valorGlobal: 1000 + i * 100,
      valorAcumulado: 1000 + i * 100,
      numeroRetificacao: 0,
      identificadorCipi: `CIPI12345${i}`,
      urlCipi: `http://example.com/cipi12345${i}`,
      usuarioNome: `Usuário ${i}`,
    })),
    totalRegistros: 10,
    totalPaginas: 1,
    numeroPagina: 1,
    paginasRestantes: 0,
    empty: false,
  ],
};

module.exports = router;
```

Importei o express para gerenciamento de rotas, axios para fazer solicitações HTTP, db para interagir com o banco de dados, e dateToString para manipulação de datas. Lembrando que a aplicação sempre vai tentar usar a API e se ela falhar ou demorar mais de 5 segundos para responder será usada a resposta simulada.

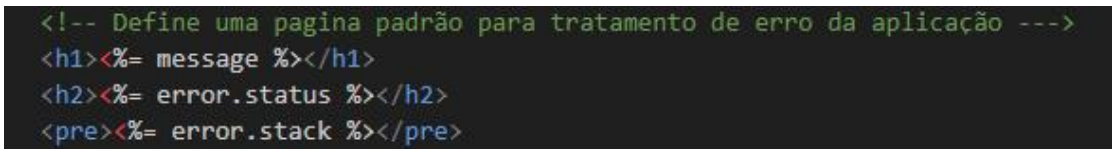
utils.js



```
// Arquivo responsavel por conter funções uteis para o sistema, como conversões e manipulações de dados
function dateToString(dateString) {
  if (!dateString) {
    return "";
  }
  return dateString.replace(/-/g, "");
}

function removeCnpjMask(cnpj) {
  if (!cnpj) {
    return "";
  }
  return cnpj.replace(/\D/g, "");
}

module.exports = { dateToString, removeCnpjMask };
```

A função *dateToString* converte uma data no formato YYYY-MM-DD para YYYYMMDD, removendo os hífens. E a função *removeCnpjMask* remove todos os caracteres que não são dígitos do cnpj.

error.ejs



```
<!-- Define uma pagina padrão para tratamento de erro da aplicação --->
<h1><%= message %></h1>
<h2><%= error.status %></h2>
<pre><%= error.stack %></pre>
```

A página de erro que informa qual o erro que ocorreu na aplicação, criado para debug.

index. ejs

```ejs
<!DOCTYPE html>
<html>
  <head>
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
    />
  </head>
  <body class="container">
    <h1 class="my-4">Consulta de contratos por CNPJ no PNCP</h1>
    <form action="/buscarContratos" method="get">
      <div class="form-group">
        <label for="cnpj">CNPJ:</label>
        <input
          type="text"
          id="cnpj"
          name="cnpj"
          class="form-control"
          required
        />
      </div>
      <div class="form-group">
        <label for="dataInicial">Data Inicial:</label>
        <input
          type="date"
          id="dataInicial"
          name="dataInicial"
          class="form-control"
          required
        />
      </div>
      <div class="form-group">
        <label for="dataFinal">Data Final:</label>
        <input
          type="date"
          id="dataFinal"
          name="dataFinal"
          class="form-control"
          required
        />
      </div>
      <button type="submit" class="btn btn-primary">Buscar</button>
    </form>
  </body>
</html>
```

A página inicial contém um formulário de pesquisa para buscar contratos com base em CNPJ e datas.

results.ejs

```ejs
<!DOCTYPE html>
<html>
  <head>
    <title>Contratos encontrados</title>
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
    />
  </head>
  <body class="container">
    <h1 class="my-4">Informações do Órgão</h1>
    <p><strong>CNPJ:</strong> <%= orgao.cnpj %></p>
    <p><strong>Razão Social:</strong> <%= orgao.razaoSocial %></p>

    <h2 class="my-4">Contratos</h2>
    <table class="table table-bordered">
      <thead class="thead-dark">
        <tr>
          <th>Data de Vigência Inicial</th>
          <th>Data de Vigência Final</th>
          <th>Razão Social do Fornecedor</th>
          <th>Objeto do Contrato</th>
          <th>Valor Inicial</th>
        </tr>
      </thead>
      <tbody>
        <% contracts.forEach(contract => { %>
        <tr>
          <td><%= contract.dataVigenciaInicio %></td>
          <td><%= contract.dataVigenciaFim %></td>
          <td><%= contract.nomeRazaoSocialFornecedor %></td>
          <td><%= contract.objetoContrato %></td>
          <td><%= contract.valorInicial %></td>
        </tr>
        <% }) %>
      </tbody>
    </table>

    <h3 class="my-4">Valor Total dos Contratos: <%= totalValue %></h3>

    <!-- Para realizar uma nova busca -->
    <h2 class="my-4">Realizar Nova Busca</h2>
    <form action="/buscarContratos" method="get">
      <div class="form-group">
        <label for="cnpj">CNPJ:</label>
        <input
          type="text"
          id="cnpj"
          name="cnpj"
          class="form-control"
          required
        />
      </div>
      <div class="form-group">
        <label for="dataInicial">Data Inicial:</label>
        <input
          type="date"
          id="dataInicial"
          name="dataInicial"
          class="form-control"
          required
        />
      </div>
      <div class="form-group">
        <label for="dataFinal">Data Final:</label>
        <input
          type="date"
          id="dataFinal"
          name="dataFinal"
          class="form-control"
          required
        />
      </div>
      <button type="submit" class="btn btn-primary">Buscar</button>
    </form>
  </body>
</html>
```

A página de resultados exibe informações sobre os contratos encontrados e oferece a opção de realizar uma nova busca.

app. js

```
const createError = require("http-errors");
const express = require("express");
const path = require("path");
const cookieParser = require("cookie-parser");
const logger = require("morgan");

const indexRouter = require("./routes/index");
const searchContractsRouter = require("./routes/searchContracts");

const app = express();

// Configura a aplicação para usar o diretório views como a pasta para buscar os arquivos de visualização na pasta views
app.set("views", path.join(__dirname, "views"));
app.set("view engine", "ejs"); // Define que sera usado EJS para visualização

app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, "public")));

app.use("/", indexRouter);
app.use("/", searchContractsRouter);

// Configura a aplicação para sempre lançar o status code 404 para quando um recurso nao for encontrado
app.use(function (req, res, next) {
  next(createError(404));
});

// Define um manipulador de erro global para aplicação
app.use(function (err, req, res, next) {
  res.locals.message = err.message;
  res.locals.error = req.app.get("env") === "development" ? err : {};

  res.status(err.status || 500);
  res.render("error");
});

module.exports = app;
```

O código inicia com a importação dos módulos essenciais para configurar o servidor e rodar a aplicação.

Conclusão

O sistema desenvolvido para consulta de contratos através da API REST do Portal Nacional de Contratações Públicas (PNCP) demonstra uma integração eficaz entre diversas tecnologias.

A implementação das funcionalidades propostas atende aos requisitos definidos e proporciona uma solução eficiente e de fácil utilização para a visualização de contratos públicos. O sistema atende todos os requisitos funcionais.