🛰️ SISTEMA DE ORQUESTRAÇÃO RPA - BEG TELECOMUNICAÇÕES

🎯 CONTEXTO E OBJETIVOS

Você é um Arquiteto de Software Sênior especializado em Python, FastAPI, automação RPA com Selenium. Desenvolva um sistema completo de orquestração de RPAs para gerenciamento automatizado de faturas de telecomunicações, seguindo Clean Code, SOLID, design patterns e tipagem forte.

CRÍTICO: Todo o sistema deve ser desenvolvido 100% em português brasileiro, incluindo variáveis, métodos, comentários, docstrings, mensagens de erro e interface do usuário.

Objetivo:

Criar um orquestrador robusto e escalável que integre RPAs existentes (preservando 100% do código legado) em uma arquitetura moderna com:

RPA Base Concentrador (padrão imutável de entrada/saída)

Workflow de aprovação obrigatório

Sistema de notificações avançado

Gestão completa de operadoras

Rastreabilidade total de processos

🛠️ STACK TECNOLÓGICA

Backend

Python 3.11+

FastAPI (API REST + WebSockets)

PostgreSQL (Banco principal)

Celery + Redis (Orquestração/Filas)

Pydantic (Validação/Serialização)

Alembic (Migrations)

SQLAlchemy (ORM com tipagem forte)

Selenium (RPAs existentes)

MinIO/S3 (Armazenamento arquivos)

Docker (Containerização)

EvolutionAPI (WhatsApp Business)

SMTP (Email)

Telegram, Slack (opcionais)

Frontend

Siga a inspiração do dashboard da imagem enviada pelo usuário (ngx-admin)

Next.js + TypeScript (React framework - alta performance)

Material-UI ou Ant Design

React Query, Socket.io-client, Chart.js ou Recharts

Alternativas: Nuxt.js + TypeScript, SvelteKit + TypeScript

🗄️ MODELAGEM DE DADOS (PostgreSQL)

-- Tabela de Operadoras

CREATE TABLE operadoras (

id UUID PRIMARY KEY,

nome VARCHAR(100) UNIQUE NOT NULL,

codigo VARCHAR(20) UNIQUE NOT NULL,

possui\_rpa BOOLEAN DEFAULT FALSE,

url\_portal VARCHAR(500),

instrucoes\_acesso TEXT,

status\_ativo BOOLEAN DEFAULT TRUE,

configuracao\_rpa JSONB,

classe\_rpa VARCHAR(100),

data\_criacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

data\_atualizacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Tabela de Clientes

CREATE TABLE clientes (

id UUID PRIMARY KEY,

hash\_unico VARCHAR(50) UNIQUE NOT NULL,

razao\_social VARCHAR(255) NOT NULL,

nome\_sat VARCHAR(255) NOT NULL,

cnpj VARCHAR(20) NOT NULL,

operadora\_id UUID REFERENCES operadoras(id),

filtro VARCHAR(255),

servico VARCHAR(255),

dados\_sat TEXT,

unidade VARCHAR(100) NOT NULL,

site\_emissao VARCHAR(255),

login\_portal VARCHAR(100),

senha\_portal VARCHAR(100),

cpf VARCHAR(20),

status\_ativo BOOLEAN DEFAULT TRUE,

data\_criacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

data\_atualizacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UNIQUE (cnpj, operadora\_id, unidade, servico)

);

-- Tabela de Processos Mensais

CREATE TABLE processos (

id UUID PRIMARY KEY,

cliente\_id UUID REFERENCES clientes(id),

mes\_ano VARCHAR(7) NOT NULL,

status\_processo VARCHAR(50) DEFAULT 'AGUARDANDO\_DOWNLOAD',

url\_fatura VARCHAR(500),

caminho\_s3\_fatura VARCHAR(500),

data\_vencimento DATE,

valor\_fatura DECIMAL(15,2),

aprovado\_por\_usuario\_id UUID,

data\_aprovacao TIMESTAMP,

enviado\_para\_sat BOOLEAN DEFAULT FALSE,

data\_envio\_sat TIMESTAMP,

upload\_manual BOOLEAN DEFAULT FALSE,

criado\_automaticamente BOOLEAN DEFAULT TRUE,

observacoes TEXT,

data\_criacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

data\_atualizacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UNIQUE (cliente\_id, mes\_ano)

);

-- Tabela de Execuções

CREATE TABLE execucoes (

id UUID PRIMARY KEY,

processo\_id UUID REFERENCES processos(id),

tipo\_execucao VARCHAR(50) CHECK (tipo\_execucao IN ('DOWNLOAD\_FATURA', 'UPLOAD\_SAT', 'UPLOAD\_MANUAL')),

status\_execucao VARCHAR(50) CHECK (status\_execucao IN ('EXECUTANDO', 'CONCLUIDO', 'FALHOU', 'TENTANDO\_NOVAMENTE')),

classe\_rpa\_utilizada VARCHAR(100),

parametros\_entrada JSONB,

resultado\_saida JSONB,

data\_inicio TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

data\_fim TIMESTAMP,

mensagem\_log TEXT,

url\_arquivo\_s3 VARCHAR(500),

numero\_tentativa INTEGER DEFAULT 1,

detalhes\_erro JSONB,

executado\_por\_usuario\_id UUID REFERENCES usuarios(id),

ip\_origem VARCHAR(45),

user\_agent TEXT

);

-- Tabela de Usuários

CREATE TABLE usuarios (

id UUID PRIMARY KEY,

nome\_completo VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

telefone VARCHAR(20),

perfil\_usuario VARCHAR(50) CHECK (perfil\_usuario IN ('ADMINISTRADOR', 'APROVADOR', 'OPERADOR')),

status\_ativo BOOLEAN DEFAULT TRUE,

data\_criacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Tabela de Notificações

CREATE TABLE notificacoes (

id UUID PRIMARY KEY,

tipo\_notificacao VARCHAR(50) CHECK (tipo\_notificacao IN ('EMAIL', 'WHATSAPP', 'TELEGRAM', 'SLACK')),

destinatario VARCHAR(255) NOT NULL,

assunto VARCHAR(255),

mensagem TEXT NOT NULL,

status\_envio VARCHAR(50) DEFAULT 'PENDENTE',

tentativas\_envio INTEGER DEFAULT 0,

data\_envio TIMESTAMP,

mensagem\_erro TEXT,

data\_criacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Tabela de Agendamentos

CREATE TABLE agendamentos (

id UUID PRIMARY KEY,

nome\_agendamento VARCHAR(255) NOT NULL,

descricao TEXT,

cron\_expressao VARCHAR(100) NOT NULL,

tipo\_agendamento VARCHAR(50) CHECK (tipo\_agendamento IN ('CRIAR\_PROCESSOS\_MENSAIS', 'EXECUTAR\_DOWNLOADS', 'ENVIAR\_RELATORIOS')),

status\_ativo BOOLEAN DEFAULT TRUE,

proxima\_execucao TIMESTAMP,

ultima\_execucao TIMESTAMP,

parametros\_execucao JSONB,

data\_criacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

🔧 RPA BASE CONCENTRADOR

Padronize todos os RPAs herdando de uma classe RPABase (imutável para entrada/saída) e registre todos em um ConcentradorRPA que direciona a operação baseada no filtro/operadora.

from abc import ABC, abstractmethod

from dataclasses import dataclass

from typing import Dict, Any, Optional, List

from enum import Enum

import logging

from datetime import datetime

class TipoOperacao(Enum):

DOWNLOAD\_FATURA = "download\_fatura"

UPLOAD\_SAT = "upload\_sat"

class StatusExecucao(Enum):

INICIADO = "iniciado"

EM\_PROGRESSO = "em\_progresso"

SUCESSO = "sucesso"

ERRO = "erro"

TIMEOUT = "timeout"

@dataclass(frozen=True)

class ParametrosEntradaPadrao:

id\_processo: str

id\_cliente: str

operadora\_codigo: str

url\_portal: str

usuario: str

senha: str

cpf: Optional[str] = None

filtro: Optional[str] = None

nome\_sat: str = ""

dados\_sat: str = ""

unidade: str = ""

servico: str = ""

@dataclass

class ResultadoSaidaPadrao:

sucesso: bool

status: StatusExecucao

mensagem: str

arquivo\_baixado: Optional[str] = None

url\_s3: Optional[str] = None

dados\_extraidos: Dict[str, Any] = None

tempo\_execucao\_segundos: float = 0.0

tentativa\_numero: int = 1

timestamp\_inicio: Optional[datetime] = None

timestamp\_fim: Optional[datetime] = None

logs\_execucao: List[str] = None

screenshots\_debug: List[str] = None

dados\_especificos: Dict[str, Any] = None

class RPABase(ABC):

@abstractmethod

def executar\_download(self, parametros: ParametrosEntradaPadrao) -> ResultadoSaidaPadrao:

pass

@abstractmethod

def executar\_upload\_sat(self, parametros: ParametrosEntradaPadrao) -> ResultadoSaidaPadrao:

pass

class ConcentradorRPA:

def \_\_init\_\_(self):

self.logger = logging.getLogger("ConcentradorRPA")

self.rpas\_registrados: Dict[str, RPABase] = {}

self.\_registrar\_rpas\_disponiveis()

def \_registrar\_rpas\_disponiveis(self) -> None:

from .rpas.embratel\_rpa import EmbratelRPA

from .rpas.digitalnet\_rpa import DigitalnetRPA

# ...demais RPAs...

self.rpas\_registrados = {

"EMB": EmbratelRPA(),

"DIG": DigitalnetRPA(),

# ...

}

def executar\_operacao(self, operacao: TipoOperacao, parametros: ParametrosEntradaPadrao) -> ResultadoSaidaPadrao:

codigo\_rpa = "SAT" if operacao == TipoOperacao.UPLOAD\_SAT else parametros.operadora\_codigo

if codigo\_rpa not in self.rpas\_registrados:

return ResultadoSaidaPadrao(

sucesso=False, status=StatusExecucao.ERRO,

mensagem=f"RPA não encontrado: {codigo\_rpa}"

)

rpa = self.rpas\_registrados[codigo\_rpa]

if operacao == TipoOperacao.DOWNLOAD\_FATURA:

return rpa.executar\_download(parametros)

elif operacao == TipoOperacao.UPLOAD\_SAT:

return rpa.executar\_upload\_sat(parametros)

return ResultadoSaidaPadrao(

sucesso=False, status=StatusExecucao.ERRO,

mensagem=f"Operação não suportada: {operacao}"

)

🖥️ LAYOUT DO FRONTEND (INSPIRAÇÃO: IMAGE1)

O frontend deve seguir a inspiração visual do dashboard ngx-admin (ver imagem do usuário).

Sugestão de organização:

Sidebar à esquerda:

Dashboard

Processos

Execuções

Clientes

Operadoras

Relatórios

Notificações

Usuários

Agendamentos

Topo fixo:

Logo

Alternância de tema (claro/escuro)

Perfil/logoff

Notificações (ícone)

Links rápidos de contato/ajuda

Dashboard principal (cards e gráficos):

Total de processos do mês (card)

Processos por status (gráfico de barras)

Faturas pendentes por operadora/cliente

Atividades recentes (timeline)

Métricas de sucesso e falha (donut/pie chart)

Cards de ações rápidas (criar processo, upload manual, aprovar fatura, etc.)

Tabelas detalhadas (com filtros avançados, botões de ação em linha):

Processos: rastreabilidade completa (todas execuções)

Clientes: CRUD e importação CSV

Operadoras: CRUD e configuração de RPA

Validação manual de faturas:

Modal com preview da fatura, campos para observação, botões "Aprovar" e "Rejeitar"

Logs em tempo real (WebSocket)

Sessão de logs detalhados com filtro por processo, cliente e operadora

🔍 ANÁLISE E REUSO DE CÓDIGO LEGADO

LEIA ARQUIVO POR ARQUIVO todo código legado fornecido

PRESERVE 100% das lógicas de scraping, XPaths, seletores

MANTENHA arquivos de teste individuais para debug VSCode

OBSERVE as chamadas de self.pm para adaptar à nova orquestração (focada em processos, execuções etc)

Implemente o RPA Base como concentrador, garantindo padrão imutável de entrada e saída

🚨 CRITÉRIOS DE ACEITE ATUALIZADOS

✅ RPA Base Concentrador implementado e funcional

✅ Padrão imutável de entrada/saída respeitado por todos RPAs

✅ 100% código legado preservado (XPaths, seletores, lógica)

✅ Arquivos de teste isolados para debug VSCode

✅ Adaptação de self.pm para nova orquestração

✅ Sistema funcional com todas operadoras

✅ Hash único funcionando corretamente

✅ Workflow de aprovação obrigatório implementado

✅ EvolutionAPI WhatsApp operacional

✅ Sistema email SMTP funcional

✅ Cadastro de operadoras com gestão RPA

✅ Upload manual com validação SAT

✅ Gestão de processos com unicidade e rastreabilidade

✅ Sistema de agendamentos para automação

✅ Frontend responsivo e intuitivo (dashboard conforme inspiração)

✅ Clean Architecture com tipagem forte

✅ Código 100% português (variáveis, métodos, comentários)

✅ Testes automatizados cobertura >85%

✅ Documentação completa em português

✅ Deploy automatizado com Docker

📋 ENTREGÁVEIS FINAIS

RPA Base Concentrador (sistema padronizado de entrada/saída)

Adaptação RPAs Legados (preservando 100% código existente)

Backend Completo (FastAPI + Clean Architecture + Tipagem forte)

Sistema Orquestração (Celery + Redis + Workflow)

Sistema Notificações (EvolutionAPI WhatsApp + Email SMTP)

Frontend Moderno (Next.js/Nuxt.js + TypeScript + Material-UI)

Gestão Operadoras (CRUD + configuração RPA)

Gestão Processos (criação mensal + rastreabilidade + unicidade)

Upload Manual (faturas + validação SAT)

Sistema Agendamentos (cron jobs + automação)

Arquivos de Teste Isolados (debug individual RPAs)

Scripts Deployment (Docker + docker-compose)

Documentação Técnica (arquitetura + APIs em português)

Testes Automatizados (unitários + integração + E2E)

OBRIGATÓRIO:

Todo código, comentários, variáveis, métodos, mensagens e documentação em português brasileiro

Analise cada arquivo anexo detalhadamente para maximizar reuso do código legado dos RPAs

Observe especialmente as chamadas self.pm para adaptar à nova orquestração QUE LIDA APENAS COM PROCESSOS, EXECUÇÕES, ETC.

Mantenha arquivos de teste individuais para debug VSCode

Implemente RPA Base como concentrador com padrão imutável

Desenvolvido por: Tiago Pereira Ramos

# Regras de Negócio — Sistema de Orquestração RPA BEG Telecomunicações

---

## 1. Cadastro e Gerenciamento de Operadoras

- Cada operadora utilizada pelo sistema deve possuir cadastro independente, registrado em tabela própria no banco de dados.

- O cadastro da operadora deve incluir, obrigatoriamente:

- \*\*Nome da operadora\*\* (ex: Embratel, Vivo, Oi, etc)

- \*\*Código identificador\*\* (sigla curta e única)

- \*\*Possui RPA homologado\*\*: Indica se a operadora já possui automação integrada e homologada ao sistema.

- \*\*Status ativo/inativo\*\*: Define se a operadora está disponível para uso em cadastros de clientes e processos.

- \*\*URL de acesso ao portal\*\* (opcional)

- \*\*Instruções/configuração\*\* (opcional)

- CRUD completo das operadoras deve ser disponibilizado no frontend administrativo.

- Clientes só podem ser associados a operadoras ativas.

---

## 2. Cadastro de Clientes Vinculado à Operadora

- Cada cliente precisa estar vinculado a uma operadora cadastrada e ativa.

- O vínculo de cliente segue a regra de unicidade: \*\*CNPJ + Operadora + Unidade (Filial) + Serviço\*\*.

- Não é permitido cadastrar clientes em operadoras inativas.

- Todos os dados necessários para automação e/ou upload manual devem estar presentes no cadastro do cliente.

---

## 3. Controle de RPAs e Upload Manual

- Se a operadora \*\*possui RPA homologado\*\* (`possui\_rpa = True`):

- Os processos de download/upload de fatura devem ser realizados automaticamente pelo sistema, via execução do RPA respectivo, sempre respeitando o padrão imutável de entrada/saída do RPA Base.

- Se a operadora \*\*não possui RPA homologado\*\* (`possui\_rpa = False`):

- O sistema habilita o fluxo de \*\*upload manual de fatura\*\* para clientes desta operadora.

- O upload manual só é permitido se o cliente estiver cadastrado e com todos os dados obrigatórios completos.

- Uploads manuais também estão sujeitos ao workflow de aprovação de fatura.

---

## 4. Criação, Unicidade, Importância da Hash e Rastreabilidade de Processos

- \*\*Identificador Único (Hash):\*\*

- Cada cliente é identificado de forma única no sistema através de uma hash gerada a partir da combinação de múltiplos atributos-chave do cliente e da operação:

- nome\_filtro, operadora, servico, dados\_sat, filtro, unidade (ver anexo de função generate\_hash\_cad).

- \*\*Importância:\*\*

- A hash garante unicidade, integridade e rastreabilidade dos clientes, mesmo quando nomes comerciais e CNPJs podem se repetir em ambientes legados (ex: o sistema SAT).

- Toda consulta, integração e vinculação de processos, execuções e uploads depende da utilização correta desta hash.

- O uso da hash evita duplicidades e inconsistências em processos mensais e históricos.

- \*\*Unicidade de Processos:\*\*

- Cada processo é único por combinação: \*\*Cliente (hash) + Operadora + Mês/Ano\*\*.

- Não pode haver mais de um processo aberto para o mesmo cliente, operadora e mês/ano.

- \*\*Rastreabilidade:\*\*

- Cada processo pode ter múltiplas execuções (tentativas), tanto para download quanto para upload.

- Todas as execuções (sucesso, falha, tentativas) devem ser rastreadas com:

- data/hora

- status

- usuário executor (quando aplicável)

- logs completos

- resultados (incluindo parâmetros de entrada/saída padronizados pelo RPA Base)

- O sistema deve permitir exportar e auditar o histórico completo de execuções por processo.

---

## 5. Workflow de Aprovação de Faturas

- Todo processo, seja ele automatizado (RPA) ou manual, deve obrigatoriamente passar pelo workflow de aprovação antes do envio ao SAT.

- \*\*Workflow padrão:\*\*

1. Processo criado (automático ou manual).

2. Download automático via RPA, ou upload manual da fatura.

3. Fatura fica \*\*pendente de aprovação\*\*.

4. Usuário com perfil de aprovador revisa, visualiza e pode adicionar observações antes de decidir entre “Aprovar” ou “Rejeitar”.

5. Apenas após aprovação, a fatura é liberada para envio ao SAT (upload via RPA SAT).

6. Aprovação/rejeição gera log, rastreabilidade e pode disparar notificações.

7. Em caso de rejeição, o processo pode ser reaberto para nova tentativa de download/upload e aprovação.

---

## 6. Notificações, Auditoria e Controle Manual

- Qualquer upload manual, aprovação, rejeição ou alteração relevante gera registro no log de auditoria do sistema.

- Notificações automáticas (e-mail, WhatsApp EvolutionAPI, etc.) devem ser enviadas para usuários responsáveis sempre que houver ações pendentes, falhas, aprovações, rejeições ou conclusões.

- O painel administrativo deve permitir consulta, filtragem e exportação da rastreabilidade de processos, execuções e aprovações.

- O controle manual é obrigatório para operadoras sem RPA homologado, mas com as mesmas garantias de unicidade, aprovação e rastreabilidade dos processos automatizados.

---

\*\*Resumo Final:\*\*

O sistema deve garantir unicidade e integridade dos clientes e processos através da hash, controlar de maneira clara o cadastro e status das operadoras (com ou sem RPA), exigir workflow de aprovação para todas as faturas, e manter logs e rastreabilidade detalhados para total segurança, transparência e auditabilidade.

Data: 29/05/2025

Sistema: Orquestrador RPA BEG Telecomunicações v2.0