Guia Completo: Criar Novo Fluxo com Nosso Motor BPMN

> **Objetivo:** Usar o Workflow Engine já implementado no projeto para criar novos fluxos de processos sem programar lógica de navegação manual.

📋 Índice

---

1. Visão Geral do Motor

🎯 O Que é o Motor BPMN

Nosso sistema possui um **Workflow Engine** que:

- ☑ Controla navegação entre steps automaticamente
- ☑ Salva progresso no banco de dados
- ☑ Permite voltar/avançar steps
- ☑ Gerencia validações
- ☑ Persiste dados em cada etapa

Você NÃO precisa programar:

- Lógica de navegação (próximo/anterior)
- Controle de estado (qual step está ativo)
- Salvamento de progresso
- Validação de steps completados

Você SÓ precisa:

- Definir os steps do fluxo
- Criar as páginas (UI) de cada step
- Chamar a API do motor

📦 Componentes Principais

```
Frontend:
├── InscricaoWizardMotor.tsx          # Wrapper principal do motor
├── InscricaoStepperMotor.tsx         # Barra de progresso
```

```
├── pages/inscricao/workflow/         # Páginas de cada step
│   ├── ParticipantesWorkflowPageMotor.tsx
│   ├── ImovelWorkflowPageMotor.tsx
│   └── EmpreendimentoWorkflowPageMotor.tsx
└── services/workflowApi.ts           # API do motor

Backend:
├── routes/workflowRoutes.js          # Endpoints do motor
├── services/workflowService.js       # Lógica do motor
└── config/workflowDefinitions.js     # Definição dos fluxos
```

🔄 Como Funciona (Fluxo Automático)

```
1. Usuario clica "Novo Processo Motor"
   ↓
2. Frontend chama POST /workflow/instances/start
   ↓
3. Backend cria workflow_instance no banco
   ↓
4. Backend retorna primeiro step: "participantes"
   ↓
5. Frontend renderiza ParticipantesWorkflowPageMotor
   ↓
6. Usuario preenche e clica "Próximo"
   ↓
7. Frontend chama POST /workflow/instances/{id}/steps/{stepId}/complete
   ↓
8. Backend salva dados e retorna próximo step: "imovel"
   ↓
9. Frontend renderiza ImovelWorkflowPageMotor
   ↓
... repete automaticamente até finalizar
```

---

2. Criar Novo Fluxo

Exemplo: Vamos criar fluxo de "Licença de Operação"

Passo 1: Definir Fluxo no Backend

Edite: `backend/config/workflowDefinitions.js`

```
// Adicione seu novo fluxo
const WORKFLOW_DEFINITIONS = {
  // Fluxo existente
  licenciamento_ambiental: {
    name: 'Licenciamento Ambiental',
    steps: [
      { id: 'participantes', name: 'Participantes', order: 1 },
```

```
      { id: 'imovel', name: 'Imóvel', order: 2 },
      { id: 'empreendimento', name: 'Empreendimento', order: 3 },
      { id: 'formulario', name: 'Formulário', order: 4 },
      { id: 'documentacao', name: 'Documentação', order: 5 },
      { id: 'revisao', name: 'Revisão', order: 6 }
    ]
  },

  // ⬇ SEU NOVO FLUXO
  licenca_operacao: {
    name: 'Licença de Operação',
    steps: [
      { id: 'dados_empresa', name: 'Dados da Empresa', order: 1 },
      { id: 'atividades', name: 'Atividades Operacionais', order: 2 },
      { id: 'equipamentos', name: 'Equipamentos', order: 3 },
      { id: 'controle_ambiental', name: 'Controle Ambiental', order: 4 },
      { id: 'monitoramento', name: 'Monitoramento', order: 5 },
      { id: 'conclusao', name: 'Conclusão', order: 6 }
    ]
  }
};

module.exports = WORKFLOW_DEFINITIONS;
```

Passo 2: API Já Está Pronta!

O endpoint `/workflow/instances/start` já suporta múltiplos fluxos:

```
// backend/routes/workflowRoutes.js
router.post('/instances/start', async (req, res) => {
  const { processId, workflowType = 'licenciamento_ambiental' } = req.body;
  //                   ⬆ Basta passar o workflowType

  const result = await workflowService.startWorkflow(processId, workflowType);
  res.json(result);
});
```

Passo 3: Testar Backend

```
# Reinicie o backend
cd backend
npm restart

# Teste a API
curl -X POST http://localhost:3000/api/v1/workflow/instances/start \
  -H "Content-Type: application/json" \
  -d '{
    "processId": "test-123",
```

```
    "workflowType": "licenca_operacao"
  }'
```

Resposta esperada:

```
{
  "instanceId": "uuid-gerado",
  "currentStep": {
    "id": "dados_empresa",
    "key": "dados_empresa",
    "label": "Dados da Empresa",
    "path": "/inscricao/dados_empresa"
  }
}
```

☑ **Backend pronto!** Agora vamos criar as páginas.

---

3. Adicionar Páginas ao Fluxo

Passo 1: Criar Componentes das Páginas

Para cada step definido, crie um componente em:

```
src/pages/inscricao/workflow/
```

Exemplo: `DadosEmpresaWorkflowPageMotor.tsx`

```
import { useState } from 'react';
import { useInscricaoStore } from '../../../lib/store/inscricao';
import { completeStep } from '../../../services/workflowApi';
import { toast } from 'react-toastify';

export default function DadosEmpresaWorkflowPageMotor() {
  const {
    workflowInstanceId,
    currentStepId,
    processId
  } = useInscricaoStore();

  const [formData, setFormData] = useState({
    razao_social: '',
    cnpj: '',
    inscricao_estadual: '',
    endereco: ''
  });
```

```
  const [loading, setLoading] = useState(false);

  const handleNext = async () => {
    // Validação
    if (!formData.razao_social || !formData.cnpj) {
      toast.error('Preencha todos os campos obrigatórios');
      return;
    }

    if (!workflowInstanceId || !currentStepId) {
      toast.error('Workflow não inicializado');
      return;
    }

    setLoading(true);
    try {
      // 1. Salvar dados no banco (opcional - pode salvar no completeStep)
      await fetch(`/api/v1/processos/${processId}/dados-empresa`, {
        method: 'PUT',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(formData)
      });

      // 2. Avançar step no workflow (automático!)
      await completeStep(workflowInstanceId, currentStepId, {
        dados_empresa: formData
      });

      // Motor avança automaticamente para próximo step
      toast.success('Dados salvos com sucesso!');
    } catch (error) {
      console.error('Erro ao salvar:', error);
      toast.error('Erro ao salvar dados');
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="p-6 space-y-6">
      <div className="border-b pb-4">
        <h2 className="text-2xl font-bold text-gray-900">
          Dados da Empresa
        </h2>
        <p className="text-gray-600 mt-1">
          Preencha as informações da empresa solicitante
        </p>
      </div>

      <div className="space-y-4">
        <div>
          <label className="block text-sm font-medium text-gray-700 mb-1">
            Razão Social *
          </label>
```

```
            <input
              type="text"
              value={formData.razao_social}
              onChange={(e) => setFormData({ ...formData, razao_social:
e.target.value })}
              className="w-full px-3 py-2 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-blue-500"
              required
            />
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">
              CNPJ *
            </label>
            <input
              type="text"
              value={formData.cnpj}
              onChange={(e) => setFormData({ ...formData, cnpj: e.target.value })}
              className="w-full px-3 py-2 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-blue-500"
              required
            />
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">
              Inscrição Estadual
            </label>
            <input
              type="text"
              value={formData.inscricao_estadual}
              onChange={(e) => setFormData({ ...formData, inscricao_estadual:
e.target.value })}
              className="w-full px-3 py-2 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-blue-500"
            />
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 mb-1">
              Endereço
            </label>
            <textarea
              value={formData.endereco}
              onChange={(e) => setFormData({ ...formData, endereco: e.target.value
})}
              rows={3}
              className="w-full px-3 py-2 border border-gray-300 rounded-lg
focus:ring-2 focus:ring-blue-500"
            />
          </div>
        </div>
```

```
      <div className="flex justify-end gap-3 pt-4 border-t">
        <button
          onClick={handleNext}
          disabled={loading}
          className="bg-blue-600 hover:bg-blue-700 text-white px-6 py-2 rounded-lg
disabled:opacity-50 disabled:cursor-not-allowed"
        >
          {loading ? 'Salvando...' : 'Próximo'}
        </button>
      </div>
    </div>
  );
}
```

Crie uma página para cada step:

- AtividadesWorkflowPageMotor.tsx
- EquipamentosWorkflowPageMotor.tsx
- ControleAmbientalWorkflowPageMotor.tsx
- MonitoramentoWorkflowPageMotor.tsx
- ConclusaoWorkflowPageMotor.tsx

Passo 2: Registrar Páginas no Wizard

Edite: src/components/InscricaoWizardMotor.tsx

```
// ⬇ Adicione os imports das novas páginas
import DadosEmpresaWorkflowPageMotor from
'../pages/inscricao/workflow/DadosEmpresaWorkflowPageMotor';
import AtividadesWorkflowPageMotor from
'../pages/inscricao/workflow/AtividadesWorkflowPageMotor';
import EquipamentosWorkflowPageMotor from
'../pages/inscricao/workflow/EquipamentosWorkflowPageMotor';
import ControleAmbientalWorkflowPageMotor from
'../pages/inscricao/workflow/ControleAmbientalWorkflowPageMotor';
import MonitoramentoWorkflowPageMotor from
'../pages/inscricao/workflow/MonitoramentoWorkflowPageMotor';
import ConclusaoWorkflowPageMotor from
'../pages/inscricao/workflow/ConclusaoWorkflowPageMotor';

// No método renderCurrentStep()
const renderCurrentStep = () => {
  if (!currentStep || !currentProcessoId) return null;

  const stepKey = currentStep.key?.toLowerCase();

  switch (stepKey) {
    // Steps existentes (licenciamento ambiental)
    case 'participantes':
      return <ParticipantesWorkflowPageMotor />;
    case 'imovel':
```

```
        return <ImovelWorkflowPageMotor />;
      case 'empreendimento':
        return (
          <EnterpriseProvider>
            <EmpreendimentoWorkflowPageMotor />
          </EnterpriseProvider>
        );
      case 'formulario':
        return <FormularioWorkflowPageMotor />;

      // ⬇ NOVOS STEPS (licença de operação)
      case 'dados_empresa':
        return <DadosEmpresaWorkflowPageMotor />;
      case 'atividades':
        return <AtividadesWorkflowPageMotor />;
      case 'equipamentos':
        return <EquipamentosWorkflowPageMotor />;
      case 'controle_ambiental':
        return <ControleAmbientalWorkflowPageMotor />;
      case 'monitoramento':
        return <MonitoramentoWorkflowPageMotor />;
      case 'conclusao':
        return <ConclusaoWorkflowPageMotor />;

      default:
        return (
          <div className="p-8 text-center">
            <p className="text-red-600">
              ⚠ Step não implementado: <code>{stepKey}</code>
            </p>
          </div>
        );
    }
  };
```

Passo 3: Atualizar Stepper (Barra de Progresso)

Edite: `src/components/InscricaoStepperMotor.tsx`

```
// Fazer o stepper dinâmico baseado no workflow
const getStepsForWorkflow = (workflowType: string) => {
  if (workflowType === 'licenca_operacao') {
    return [
      { number: 1, title: 'Dados da Empresa', key: 'dados_empresa' },
      { number: 2, title: 'Atividades', key: 'atividades' },
      { number: 3, title: 'Equipamentos', key: 'equipamentos' },
      { number: 4, title: 'Controle Ambiental', key: 'controle_ambiental' },
      { number: 5, title: 'Monitoramento', key: 'monitoramento' },
      { number: 6, title: 'Conclusão', key: 'conclusao' }
    ];
  }
```

```
  // Default: licenciamento ambiental
  return [
    { number: 1, title: 'Participantes', key: 'participantes' },
    { number: 2, title: 'Imóvel', key: 'imovel' },
    { number: 3, title: 'Empreendimento', key: 'empreendimento' },
    { number: 4, title: 'Formulário', key: 'formulario' },
    { number: 5, title: 'Documentação', key: 'documentacao' },
    { number: 6, title: 'Revisão', key: 'revisao' }
  ];
};
```

---

4. Configurar Navegação Automática

☑ O Motor Já Faz Isso Automaticamente!

Você **NÃO precisa** programar navegação. O motor cuida de:

1. Avançar para Próximo Step

```
// Na sua página, apenas chame:
await completeStep(workflowInstanceId, currentStepId, dados);

// O backend automaticamente:
// 1. Salva os dados
// 2. Marca step como completado
// 3. Retorna o PRÓXIMO step
// 4. Frontend atualiza a UI automaticamente
```

2. Atualizar UI Automaticamente

```
// InscricaoWizardMotor.tsx já tem isso implementado:
const handleNext = async (stepData?: any) => {
  const response = await completeStep(workflowInstanceId, currentStep.id,
stepData);

  if (response.nextStep) {
    setCurrentStep(response.nextStep);  // ← UI atualiza sozinha!
    toast.success(`Avançado para: ${response.nextStep.label}`);
  }
};
```

3. Salvar Progresso no Banco

```
// Backend salva automaticamente em workflow_steps:
await db.workflow_steps.create({
  instance_id: instanceId,
  step_id: stepId,
  step_data: data,
  completed_at: new Date()
});
```

Como Personalizar Navegação (Avançado)

Se precisar de lógica condicional (ex: pular steps):

Backend: `workflowService.js`

```
async completeStep(instanceId, stepId, data) {
  // Exemplo: Lógica condicional
  if (data.tipo_licenca === 'simplificada') {
    // Pula steps 3 e 4, vai direto para step 5
    return this.goToStep(instanceId, 5);
  }

  // Navegação normal (próximo step)
  return this.nextStep(instanceId);
}
```

---

5. Testar o Fluxo

Teste Manual Completo

1. **Inicie o backend**

   ```
   cd backend
   npm run dev
   ```

2. **Inicie o frontend**

   ```
   npm run dev
   ```

3. **Acesse o sistema**

   ```
   http://localhost:5173
   ```

4. **Navegue para o fluxo**
    - Login
    - Menu → "Processos Motor"
    - Clique "Novo Processo Motor"
5. **Teste cada step** (navegação automática!)
    - ☐ Step 1: Dados da Empresa → preencher → "Próximo"
    - ☐ Step 2: Atividades → preencher → "Próximo"
    - ☐ Step 3: Equipamentos → preencher → "Próximo"
    - ☐ Step 4: Controle Ambiental → preencher → "Próximo"
    - ☐ Step 5: Monitoramento → preencher → "Próximo"
    - ☐ Step 6: Conclusão → "Finalizar"
6. **Verifique no banco**

```
-- Instância criada
SELECT * FROM workflow_instances
WHERE workflow_type = 'licenca_operacao'
ORDER BY created_at DESC LIMIT 1;

-- Steps completados
SELECT * FROM workflow_steps
WHERE instance_id = 'SEU_INSTANCE_ID'
ORDER BY created_at;

-- Dados salvos
SELECT * FROM processos_licenciamento
WHERE id = 'SEU_PROCESSO_ID';
```

Teste Automatizado

`tests/test_licenca_operacao.py`

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

def test_fluxo_licenca_operacao():
    driver = webdriver.Chrome()
    wait = WebDriverWait(driver, 10)

    try:
        # 1. Login
        driver.get('http://localhost:5173/login')
        driver.find_element(By.ID, 'email').send_keys('admin@test.com')
        driver.find_element(By.ID, 'password').send_keys('senha123')
```

```
        driver.find_element(By.XPATH, '//button[text()="Entrar"]').click()
        time.sleep(2)

        # 2. Processos Motor
        driver.find_element(By.XPATH, '//a[text()="Processos Motor"]').click()
        time.sleep(1)

        # 3. Novo Processo
        driver.find_element(By.XPATH, '//button[text()="Novo Processo
Motor"]').click()
        time.sleep(2)

        # 4. Step 1: Dados da Empresa
        wait.until(EC.presence_of_element_located((By.XPATH, '//h2[text()="Dados
da Empresa"]')))
        driver.find_element(By.NAME, 'razao_social').send_keys('Empresa Teste
LTDA')
        driver.find_element(By.NAME, 'cnpj').send_keys('12.345.678/0001-90')
        driver.find_element(By.XPATH, '//button[text()="Próximo"]').click()
        time.sleep(2)

        # 5. Step 2: Atividades (motor avança automaticamente!)
        wait.until(EC.presence_of_element_located((By.XPATH,
'//h2[text()="Atividades Operacionais"]')))
        driver.find_element(By.XPATH, '//button[text()="Próximo"]').click()
        time.sleep(2)

        # Continuar para todos os steps...

        # Verificar conclusão
        wait.until(EC.presence_of_element_located((By.XPATH, '//*[text()="Processo
concluído"]')))

        print("✅ Teste do fluxo passou!")

    finally:
        driver.quit()

if __name__ == '__main__':
    test_fluxo_licenca_operacao()
```

Execute:

```
cd tests
python test_licenca_operacao.py
```

---

6. Publicar

Checklist Pré-Publicação

- [ ] Definição do fluxo no `workflowDefinitions.js`
- [ ] Todas as páginas criadas e registradas
- [ ] Stepper atualizado com novos steps
- [ ] Testes manuais passando
- [ ] Navegação automática funcionando
- [ ] Dados salvando no banco
- [ ] Performance < 2s por step
- [ ] Logs sem erros

Deploy

```
# 1. Commit
git add .
git commit -m "feat: adicionar fluxo Licença de Operação ao motor BPMN"

# 2. Push
git push origin sua-branch

# 3. Merge (após review)
git checkout main
git merge sua-branch
git push origin main

# 4. Deploy backend
cd backend
git push heroku main

# 5. Deploy frontend
npm run build
netlify deploy --prod
```

---

7. Troubleshooting

Problema: "Workflow não inicia"

Causa: Definição não encontrada

Solução:

```
// Verificar se workflowType existe em workflowDefinitions.js
const def = WORKFLOW_DEFINITIONS['licenca_operacao'];
console.log(def); // deve retornar objeto com steps
```

Problema: "Step não renderiza"

Causa: Página não registrada no switch

Solução:

```
// InscricaoWizardMotor.tsx
case 'dados_empresa':  // ← Adicionar case
  return <DadosEmpresaWorkflowPageMotor />;
```

Problema: "Navegação não avança"

Causa: `completeStep` não chamado ou erro na API

Solução:

```
// Verificar se está chamando
await completeStep(workflowInstanceId, currentStepId, dados);

// Verificar console do navegador
// Deve mostrar: "✅ Step completado" e próximo step
```

Problema: "Dados não salvam"

Causa: API de salvamento falhando

Solução:

```
// Verificar endpoint existe
await fetch(`/api/v1/processos/${processId}/dados-empresa`, {...});

// Verificar backend recebe os dados
console.log('Dados recebidos:', req.body);
```

🎯 Resumo Rápido

```
# 1. Definir fluxo (backend/config/workflowDefinitions.js)
licenca_operacao: {
  steps: [...]
}

# 2. Criar páginas (src/pages/inscricao/workflow/)
DadosEmpresaWorkflowPageMotor.tsx
AtividadesWorkflowPageMotor.tsx
...

# 3. Registrar no wizard (src/components/InscricaoWizardMotor.tsx)
case 'dados_empresa':
  return <DadosEmpresaWorkflowPageMotor />;
```

```
# 4. Atualizar stepper (src/components/InscricaoStepperMotor.tsx)
{ number: 1, title: 'Dados da Empresa', key: 'dados_empresa' }

# 5. Testar
npm run dev
# Ir para Processos Motor → Novo Processo

# 6. Deploy
git commit -am "feat: novo fluxo"
git push
npm run build
```

🚀 Vantagens do Motor

1. **Zero Programação de Navegação**: Motor controla tudo
2. **Reuso de Componentes**: Mesmas páginas em múltiplos fluxos
3. **Persistência Automática**: Progresso salvo no banco
4. **Extensível**: Fácil adicionar novos fluxos
5. **Testável**: API padronizada facilita testes
6. **Manutenível**: Mudanças no fluxo sem alterar código

Criado em: 2025-11-12
Versão: 1.0
Autor: GitHub Copilot
Projeto: Sistema de Licenciamento Ambiental - Motor BPMN