**Formulário de Entrada de Dados DO Plano de Tesouraria Mensal**

**1. Formulário de Entrada de Dados (React Dinâmico)**

**VER:**

[**Formulário Plano de Tesouraria - Integrado com Orçamento | Claude | Claude**](https://claude.ai/public/artifacts/c484b15b-26fc-455e-8754-7c90ed36810b)

**CODIGO:**

import React, { useState, useEffect } from 'react';

import { Calendar, Plus, Trash2, Save, Calculator, AlertCircle, ChevronRight, DollarSign, TrendingUp, TrendingDown, FileText, Download, RefreshCw, Database, CheckCircle } from 'lucide-react';

const TreasuryPlanForm = () => {

*// Dados simulados do orçamento aprovado (viriam do backend/API)*

const approvedBudget = {

year: 2025,

company: 'ENDIAGRO, Lda',

status: 'approved',

approvedBy: 'Conselho de Gerência',

approvalDate: '2024-09-15',

revenues: [

{ pgc: '714', name: 'Vendas de Produtos Agrícolas', description: 'Milho', annual: 8000000, q1: 1500000, q2: 2000000, q3: 2500000, q4: 2000000 },

{ pgc: '714', name: 'Vendas de Produtos Agrícolas', description: 'Feijão', annual: 4500000, q1: 800000, q2: 1200000, q3: 1500000, q4: 1000000 },

{ pgc: '715', name: 'Vendas de Produtos Pecuários', description: 'Gado Bovino', annual: 2500000, q1: 400000, q2: 600000, q3: 800000, q4: 700000 },

{ pgc: '72', name: 'Prestação de Serviços', description: 'Serviços Técnicos', annual: 1200000, q1: 300000, q2: 300000, q3: 300000, q4: 300000 }

],

costs: [

{ pgc: '611', name: 'Matérias-Primas', description: 'Sementes e Adubos', annual: 3000000, q1: 800000, q2: 1200000, q3: 600000, q4: 400000 },

{ pgc: '612', name: 'Materiais Diversos', description: 'Ração e Medicamentos', annual: 1700000, q1: 400000, q2: 500000, q3: 450000, q4: 350000 },

{ pgc: '613', name: 'Combustíveis e Lubrificantes', description: 'Diesel e Óleos', annual: 1000000, monthly: 83333 },

{ pgc: '622', name: 'Serviços Especializados', description: 'Consultoria Agronómica', annual: 600000, monthly: 50000 },

{ pgc: '624', name: 'Energia e Fluidos', description: 'Energia e Água', annual: 475000, monthly: 39583 },

{ pgc: '625', name: 'Transportes e Deslocações', description: 'Logística', annual: 600000, q1: 150000, q2: 150000, q3: 175000, q4: 125000 },

{ pgc: '632', name: 'Remunerações do Pessoal', description: 'Salários', annual: 2600000, monthly: 216667 },

{ pgc: '635', name: 'Encargos sobre Remunerações', description: 'INSS (8%)', annual: 208000, monthly: 17333 }

]

};

const [formData, setFormData] = useState({

metadata: {

month: new Date().getMonth() + 1,

year: new Date().getFullYear(),

company: approvedBudget.company,

preparedBy: '',

approvedBy: '',

budgetReference: `ORÇ/${approvedBudget.year}/001`

},

initialBalance: 2500000,

inflows: [],

outflows: [],

financing: {

endiama: 0,

bank: 0,

other: 0

},

importedFromBudget: false

});

const [currentStep, setCurrentStep] = useState(0);

const [validationErrors, setValidationErrors] = useState({});

*// Mapeamento PGC unificado (consistente com o orçamento)*

const pgcMapping = {

revenues: {

'711': { name: 'Vendas de Mercadorias', type: 'operacional' },

'712': { name: 'Vendas de Produtos Acabados', type: 'operacional' },

'714': { name: 'Vendas de Produtos Agrícolas', type: 'operacional' },

'715': { name: 'Vendas de Produtos Pecuários', type: 'operacional' },

'72': { name: 'Prestação de Serviços', type: 'operacional' },

'78': { name: 'Proveitos Financeiros', type: 'financeiro' },

'79': { name: 'Proveitos Extraordinários', type: 'extraordinario' },

'80': { name: 'Suprimentos/Empréstimos', type: 'financiamento' }

},

costs: {

'611': { name: 'Matérias-Primas', type: 'operacional', priority: 3 },

'612': { name: 'Materiais Diversos', type: 'operacional', priority: 3 },

'613': { name: 'Combustíveis e Lubrificantes', type: 'operacional', priority: 3 },

'621': { name: 'Subcontratos', type: 'operacional', priority: 3 },

'622': { name: 'Serviços Especializados', type: 'operacional', priority: 3 },

'623': { name: 'Materiais', type: 'operacional', priority: 3 },

'624': { name: 'Energia e Fluidos', type: 'operacional', priority: 2 },

'625': { name: 'Deslocações, Estadas e Transportes', type: 'operacional', priority: 3 },

'626': { name: 'Serviços Diversos', type: 'operacional', priority: 4 },

'632': { name: 'Remunerações do Pessoal', type: 'folha', priority: 1 },

'635': { name: 'Encargos sobre Remunerações (INSS)', type: 'folha', priority: 1 },

'641': { name: 'Amortizações do Exercício', type: 'nao-desembolsavel', priority: 5 },

'681': { name: 'Juros Suportados', type: 'financeiro', priority: 2 },

'69': { name: 'Outros Custos Operacionais', type: 'operacional', priority: 4 }

}

};

const steps = [

{ id: 0, title: 'Importação Orçamento', icon: '📥' },

{ id: 1, title: 'Informações Gerais', icon: '📋' },

{ id: 2, title: 'Saldo Inicial', icon: '💰' },

{ id: 3, title: 'Entradas Previstas', icon: '📈' },

{ id: 4, title: 'Saídas Previstas', icon: '📉' },

{ id: 5, title: 'Financiamento', icon: '🏦' },

{ id: 6, title: 'Resumo e Validação', icon: '✅' }

];

*// Função para importar dados do orçamento*

const importFromBudget = () => {

const targetMonth = formData.metadata.month;

const targetYear = formData.metadata.year;

const quarter = Math.ceil(targetMonth / 3);

const quarterKey = `q${quarter}`;

*// Converter receitas do orçamento*

const importedInflows = approvedBudget.revenues.map((revenue, idx) => {

const quarterAmount = revenue[quarterKey] || revenue.annual / 4;

const monthlyAmount = quarterAmount / 3;

const dayOfMonth = 15 + (idx \* 3); *// Distribuir ao longo do mês*

return {

id: Date.now() + idx,

pgcCode: revenue.pgc,

pgcName: revenue.name,

description: revenue.description,

amount: monthlyAmount,

expectedDate: new Date(targetYear, targetMonth - 1, dayOfMonth).toISOString().split('T')[0],

probability: 90,

source: 'Orçamento Aprovado',

paymentMethod: 'TED',

notes: `Importado do ${formData.metadata.budgetReference}`,

fromBudget: true,

budgetAnnual: revenue.annual,

budgetQuarter: quarterAmount

};

});

*// Converter custos do orçamento*

const importedOutflows = approvedBudget.costs.map((cost, idx) => {

let monthlyAmount;

let scheduledDay = 10;

if (cost.monthly) {

monthlyAmount = cost.monthly;

*// Definir dias específicos para pagamentos recorrentes*

if (cost.pgc === '632') scheduledDay = 25; *// Salários no dia 25*

if (cost.pgc === '635') scheduledDay = 10; *// INSS no dia 10*

if (cost.pgc === '624') scheduledDay = 5; *// Energia no dia 5*

} else {

const quarterAmount = cost[quarterKey] || cost.annual / 4;

monthlyAmount = quarterAmount / 3;

scheduledDay = 10 + (idx \* 2); *// Distribuir pagamentos*

}

const costMapping = pgcMapping.costs[cost.pgc] || {};

return {

id: Date.now() + idx + 1000,

pgcCode: cost.pgc,

pgcName: cost.name,

description: cost.description,

amount: monthlyAmount,

scheduledDate: new Date(targetYear, targetMonth - 1, scheduledDay).toISOString().split('T')[0],

priority: costMapping.priority || 3,

supplier: 'Diversos',

paymentMethod: 'TED',

recurring: ['632', '635', '624'].includes(cost.pgc),

frequency: cost.monthly ? 'mensal' : 'variável',

notes: `Importado do ${formData.metadata.budgetReference}`,

fromBudget: true,

budgetAnnual: cost.annual

};

});

setFormData(prev => ({

...prev,

inflows: importedInflows,

outflows: importedOutflows,

importedFromBudget: true,

metadata: {

...prev.metadata,

budgetImportDate: new Date().toISOString()

}

}));

};

*// Adicionar entrada manual*

const addInflow = () => {

const newInflow = {

id: Date.now(),

pgcCode: '714',

pgcName: pgcMapping.revenues['714'].name,

description: '',

amount: 0,

expectedDate: new Date(formData.metadata.year, formData.metadata.month - 1, 15).toISOString().split('T')[0],

probability: 90,

source: '',

paymentMethod: 'TED',

notes: '',

fromBudget: false

};

setFormData(prev => ({

...prev,

inflows: [...prev.inflows, newInflow]

}));

};

*// Adicionar saída manual*

const addOutflow = () => {

const newOutflow = {

id: Date.now(),

pgcCode: '611',

pgcName: pgcMapping.costs['611'].name,

description: '',

amount: 0,

scheduledDate: new Date(formData.metadata.year, formData.metadata.month - 1, 10).toISOString().split('T')[0],

priority: 3,

supplier: '',

paymentMethod: 'TED',

recurring: false,

frequency: 'mensal',

notes: '',

fromBudget: false

};

setFormData(prev => ({

...prev,

outflows: [...prev.outflows, newOutflow]

}));

};

*// Atualizar entrada*

const updateInflow = (id, field, value) => {

setFormData(prev => ({

...prev,

inflows: prev.inflows.map(inf => {

if (inf.id === id) {

const updated = { ...inf, [field]: value };

if (field === 'pgcCode') {

updated.pgcName = pgcMapping.revenues[value]?.name || '';

}

return updated;

}

return inf;

})

}));

};

*// Atualizar saída*

const updateOutflow = (id, field, value) => {

setFormData(prev => ({

...prev,

outflows: prev.outflows.map(out => {

if (out.id === id) {

const updated = { ...out, [field]: value };

if (field === 'pgcCode') {

const costInfo = pgcMapping.costs[value];

if (costInfo) {

updated.pgcName = costInfo.name;

updated.priority = costInfo.priority;

}

}

return updated;

}

return out;

})

}));

};

const removeInflow = (id) => {

setFormData(prev => ({

...prev,

inflows: prev.inflows.filter(inf => inf.id !== id)

}));

};

const removeOutflow = (id) => {

setFormData(prev => ({

...prev,

outflows: prev.outflows.filter(out => out.id !== id)

}));

};

const calculateTotals = () => {

const totalInflows = formData.inflows.reduce((sum, inf) =>

sum + (parseFloat(inf.amount) || 0) \* (inf.probability / 100), 0

);

const totalOutflows = formData.outflows.reduce((sum, out) =>

sum + (parseFloat(out.amount) || 0), 0

);

const totalFinancing = parseFloat(formData.financing.endiama || 0) +

parseFloat(formData.financing.bank || 0) +

parseFloat(formData.financing.other || 0);

const netFlow = totalInflows + totalFinancing - totalOutflows;

const finalBalance = parseFloat(formData.initialBalance || 0) + netFlow;

*// Análise de liquidez por período*

const firstWeek = formData.outflows

.filter(out => new Date(out.scheduledDate).getDate() <= 7)

.reduce((sum, out) => sum + parseFloat(out.amount || 0), 0);

const criticalPayments = formData.outflows

.filter(out => out.priority === 1)

.reduce((sum, out) => sum + parseFloat(out.amount || 0), 0);

return {

totalInflows,

totalOutflows,

totalFinancing,

netFlow,

finalBalance,

firstWeek,

criticalPayments,

liquidityRatio: totalInflows > 0 ? (totalInflows / totalOutflows) : 0

};

};

const formatCurrency = (value) => {

return new Intl.NumberFormat('pt-AO', {

style: 'currency',

currency: 'AOA',

minimumFractionDigits: 2

}).format(value).replace('AOA', 'Kz');

};

const renderStepContent = () => {

switch(currentStep) {

case 0: *// Importação do Orçamento*

return (

<div className="space-y-6">

<div className="bg-gradient-to-r from-indigo-50 to-purple-50 rounded-xl p-6 border border-indigo-200">

<h3 className="text-lg font-semibold text-gray-900 mb-4">

Importação de Dados do Orçamento Aprovado

</h3>

{!formData.importedFromBudget ? (

<>

<div className="bg-white rounded-lg p-4 mb-4">

<div className="flex items-center justify-between mb-3">

<div className="flex items-center gap-3">

<FileText className="w-6 h-6 text-indigo-600" />

<div>

<p className="font-medium text-gray-900">Orçamento Disponível</p>

<p className="text-sm text-gray-600">{formData.metadata.budgetReference}</p>

</div>

</div>

<span className="px-3 py-1 bg-green-100 text-green-700 rounded-full text-sm font-medium">

Aprovado

</span>

</div>

<div className="grid grid-cols-2 gap-4 mt-4 text-sm">

<div>

<p className="text-gray-600">Empresa:</p>

<p className="font-medium">{approvedBudget.company}</p>

</div>

<div>

<p className="text-gray-600">Exercício:</p>

<p className="font-medium">{approvedBudget.year}</p>

</div>

<div>

<p className="text-gray-600">Aprovado por:</p>

<p className="font-medium">{approvedBudget.approvedBy}</p>

</div>

<div>

<p className="text-gray-600">Data de Aprovação:</p>

<p className="font-medium">{approvedBudget.approvalDate}</p>

</div>

</div>

</div>

<div className="bg-yellow-50 border border-yellow-200 rounded-lg p-4 mb-4">

<div className="flex items-start gap-3">

<AlertCircle className="w-5 h-5 text-yellow-600 mt-0.5" />

<div className="flex-1">

<p className="text-sm font-medium text-yellow-900">O que será importado:</p>

<ul className="text-sm text-yellow-700 mt-2 space-y-1">

<li>• Valores trimestrais convertidos para o mês selecionado</li>

<li>• Custos fixos mensais (salários, energia, etc.)</li>

<li>• Mapeamento PGC-AO consistente com o orçamento</li>

<li>• Prioridades automáticas baseadas no tipo de despesa</li>

</ul>

</div>

</div>

</div>

<button

onClick={importFromBudget}

className="w-full px-6 py-3 bg-gradient-to-r from-indigo-600 to-purple-600 text-white rounded-lg hover:opacity-90 transition-opacity flex items-center justify-center gap-2"

>

<Download className="w-5 h-5" />

Importar Dados do Orçamento

</button>

</>

) : (

<div className="bg-green-50 border border-green-200 rounded-lg p-4">

<div className="flex items-center gap-3">

<CheckCircle className="w-6 h-6 text-green-600" />

<div>

<p className="font-medium text-green-900">Dados Importados com Sucesso!</p>

<p className="text-sm text-green-700 mt-1">

{formData.inflows.length} entradas e {formData.outflows.length} saídas foram importadas

</p>

<p className="text-xs text-green-600 mt-2">

Você pode ajustar os valores nas próximas etapas se necessário

</p>

</div>

</div>

</div>

)}

</div>

{formData.importedFromBudget && (

<div className="bg-blue-50 border border-blue-200 rounded-lg p-4">

<div className="flex items-start gap-3">

<AlertCircle className="w-5 h-5 text-blue-600 mt-0.5" />

<div>

<p className="text-sm font-medium text-blue-900">Próximos Passos</p>

<p className="text-sm text-blue-700 mt-1">

Revise e ajuste os dados importados conforme necessário. Você pode adicionar

entradas e saídas extras ou modificar as existentes.

</p>

</div>

</div>

</div>

)}

</div>

);

case 1: *// Informações Gerais*

return (

<div className="space-y-6">

<h3 className="text-lg font-semibold text-gray-900">Informações do Plano</h3>

{formData.importedFromBudget && (

<div className="bg-green-50 border border-green-200 rounded-lg p-3 mb-4">

<p className="text-sm text-green-700">

✓ Dados importados do {formData.metadata.budgetReference}

</p>

</div>

)}

<div className="grid grid-cols-2 gap-4">

<div>

<label className="block text-sm font-medium text-gray-700 mb-2">

Mês de Referência

</label>

<select

className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500"

value={formData.metadata.month}

onChange={(e) => setFormData(prev => ({

...prev,

metadata: { ...prev.metadata, month: parseInt(e.target.value) }

}))}

>

{[

'Janeiro', 'Fevereiro', 'Março', 'Abril', 'Maio', 'Junho',

'Julho', 'Agosto', 'Setembro', 'Outubro', 'Novembro', 'Dezembro'

].map((month, index) => (

<option key={index} value={index + 1}>{month}</option>

))}

</select>

</div>

<div>

<label className="block text-sm font-medium text-gray-700 mb-2">

Ano

</label>

<select

className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500"

value={formData.metadata.year}

onChange={(e) => setFormData(prev => ({

...prev,

metadata: { ...prev.metadata, year: parseInt(e.target.value) }

}))}

>

<option value="2024">2024</option>

<option value="2025">2025</option>

</select>

</div>

</div>

<div>

<label className="block text-sm font-medium text-gray-700 mb-2">

Empresa

</label>

<input

type="text"

className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500 bg-gray-50"

value={formData.metadata.company}

readOnly

/>

</div>

<div className="grid grid-cols-2 gap-4">

<div>

<label className="block text-sm font-medium text-gray-700 mb-2">

Elaborado por

</label>

<input

type="text"

className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500"

placeholder="Nome do responsável"

value={formData.metadata.preparedBy}

onChange={(e) => setFormData(prev => ({

...prev,

metadata: { ...prev.metadata, preparedBy: e.target.value }

}))}

/>

</div>

<div>

<label className="block text-sm font-medium text-gray-700 mb-2">

Aprovado por

</label>

<input

type="text"

className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500"

placeholder="Nome do aprovador"

value={formData.metadata.approvedBy}

onChange={(e) => setFormData(prev => ({

...prev,

metadata: { ...prev.metadata, approvedBy: e.target.value }

}))}

/>

</div>

</div>

<div>

<label className="block text-sm font-medium text-gray-700 mb-2">

Referência do Orçamento

</label>

<input

type="text"

className="w-full px-4 py-2 border border-gray-300 rounded-lg bg-gray-50"

value={formData.metadata.budgetReference}

readOnly

/>

<p className="text-xs text-gray-500 mt-1">

Documento de origem dos dados importados

</p>

</div>

</div>

);

case 2: *// Saldo Inicial*

return (

<div className="space-y-6">

<h3 className="text-lg font-semibold text-gray-900">Saldo Inicial de Tesouraria</h3>

<div className="bg-blue-50 border border-blue-200 rounded-lg p-4">

<div className="flex items-start gap-3">

<AlertCircle className="w-5 h-5 text-blue-600 mt-0.5" />

<div>

<p className="text-sm font-medium text-blue-900">Informação Importante</p>

<p className="text-sm text-blue-700 mt-1">

O saldo inicial deve refletir o valor disponível em caixa e bancos no primeiro dia do mês.

</p>

</div>

</div>

</div>

<div>

<label className="block text-sm font-medium text-gray-700 mb-2">

Saldo Inicial (Kz)

</label>

<div className="relative">

<DollarSign className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400 w-5 h-5" />

<input

type="number"

className="w-full pl-10 pr-4 py-3 text-lg border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500"

placeholder="0,00"

value={formData.initialBalance}

onChange={(e) => setFormData(prev => ({

...prev,

initialBalance: parseFloat(e.target.value) || 0

}))}

/>

</div>

{formData.initialBalance > 0 && (

<p className="mt-2 text-sm text-gray-600">

Valor formatado: {formatCurrency(formData.initialBalance)}

</p>

)}

</div>

</div>

);

case 3: *// Entradas Previstas*

return (

<div className="space-y-6">

<div className="flex justify-between items-center">

<h3 className="text-lg font-semibold text-gray-900">Entradas Previstas (Classe 7 - Proveitos)</h3>

<button

onClick={addInflow}

className="flex items-center gap-2 px-4 py-2 bg-green-600 text-white rounded-lg hover:bg-green-700 transition-colors"

>

<Plus className="w-4 h-4" />

Adicionar Entrada Extra

</button>

</div>

{formData.importedFromBudget && (

<div className="bg-blue-50 border border-blue-200 rounded-lg p-3">

<p className="text-sm text-blue-700">

ℹ️ Entradas marcadas com 📥 foram importadas do orçamento

</p>

</div>

)}

{formData.inflows.length === 0 ? (

<div className="text-center py-12 bg-gray-50 rounded-lg">

<TrendingUp className="w-12 h-12 text-gray-300 mx-auto mb-3" />

<p className="text-gray-500">Nenhuma entrada prevista</p>

<p className="text-sm text-gray-400 mt-1">Importe do orçamento ou adicione manualmente</p>

</div>

) : (

<div className="space-y-4">

{formData.inflows.map((inflow, index) => (

<div key={inflow.id} className={`border rounded-lg p-4 hover:shadow-md transition-shadow ${

inflow.fromBudget ? 'bg-green-50 border-green-200' : 'bg-white border-gray-200'

}`}>

<div className="flex justify-between items-start mb-3">

<h4 className="font-medium text-gray-900 flex items-center gap-2">

Entrada #{index + 1}

{inflow.fromBudget && <span title="Importado do orçamento">📥</span>}

</h4>

<button

onClick={() => removeInflow(inflow.id)}

className="text-red-600 hover:bg-red-50 p-1 rounded"

>

<Trash2 className="w-4 h-4" />

</button>

</div>

<div className="grid grid-cols-1 md:grid-cols-2 gap-4">

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Conta PGC

</label>

<select

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

value={inflow.pgcCode}

onChange={(e) => updateInflow(inflow.id, 'pgcCode', e.target.value)}

>

{Object.entries(pgcMapping.revenues).map(([code, info]) => (

<option key={code} value={code}>

{code} - {info.name}

</option>

))}

</select>

</div>

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Descrição

</label>

<input

type="text"

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

placeholder="Detalhe da entrada"

value={inflow.description}

onChange={(e) => updateInflow(inflow.id, 'description', e.target.value)}

/>

</div>

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Valor (Kz)

</label>

<input

type="number"

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

placeholder="0,00"

value={inflow.amount}

onChange={(e) => updateInflow(inflow.id, 'amount', e.target.value)}

/>

</div>

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Data Prevista

</label>

<input

type="date"

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

value={inflow.expectedDate}

onChange={(e) => updateInflow(inflow.id, 'expectedDate', e.target.value)}

/>

</div>

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Probabilidade (%)

</label>

<select

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

value={inflow.probability}

onChange={(e) => updateInflow(inflow.id, 'probability', parseInt(e.target.value))}

>

<option value="100">100% - Confirmado</option>

<option value="90">90% - Muito Provável</option>

<option value="75">75% - Provável</option>

<option value="50">50% - Incerto</option>

</select>

</div>

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Origem/Cliente

</label>

<input

type="text"

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

placeholder="Nome do cliente/origem"

value={inflow.source}

onChange={(e) => updateInflow(inflow.id, 'source', e.target.value)}

/>

</div>

</div>

{inflow.fromBudget && inflow.budgetAnnual && (

<div className="mt-3 pt-3 border-t border-gray-200">

<p className="text-xs text-gray-600">

Orçamento Anual: {formatCurrency(inflow.budgetAnnual)} |

Trimestre: {formatCurrency(inflow.budgetQuarter || 0)}

</p>

</div>

)}

{inflow.amount > 0 && (

<div className="mt-3 pt-3 border-t">

<p className="text-sm text-gray-600">

Valor esperado: {formatCurrency(inflow.amount \* (inflow.probability / 100))}

</p>

</div>

)}

</div>

))}

</div>

)}

</div>

);

case 4: *// Saídas Previstas*

return (

<div className="space-y-6">

<div className="flex justify-between items-center">

<h3 className="text-lg font-semibold text-gray-900">Saídas Previstas (Classe 6 - Custos)</h3>

<button

onClick={addOutflow}

className="flex items-center gap-2 px-4 py-2 bg-red-600 text-white rounded-lg hover:bg-red-700 transition-colors"

>

<Plus className="w-4 h-4" />

Adicionar Saída Extra

</button>

</div>

{formData.importedFromBudget && (

<div className="bg-blue-50 border border-blue-200 rounded-lg p-3">

<p className="text-sm text-blue-700">

ℹ️ Saídas marcadas com 📥 foram importadas do orçamento

</p>

</div>

)}

{formData.outflows.length === 0 ? (

<div className="text-center py-12 bg-gray-50 rounded-lg">

<TrendingDown className="w-12 h-12 text-gray-300 mx-auto mb-3" />

<p className="text-gray-500">Nenhuma saída prevista</p>

<p className="text-sm text-gray-400 mt-1">Importe do orçamento ou adicione manualmente</p>

</div>

) : (

<div className="space-y-4">

{formData.outflows.map((outflow, index) => (

<div key={outflow.id} className={`border rounded-lg p-4 hover:shadow-md transition-shadow ${

outflow.fromBudget ? 'bg-red-50 border-red-200' : 'bg-white border-gray-200'

}`}>

<div className="flex justify-between items-start mb-3">

<h4 className="font-medium text-gray-900 flex items-center gap-2">

Saída #{index + 1}

{outflow.fromBudget && <span title="Importado do orçamento">📥</span>}

{outflow.priority === 1 && <span className="text-xs bg-red-100 text-red-700 px-2 py-1 rounded">Crítica</span>}

</h4>

<button

onClick={() => removeOutflow(outflow.id)}

className="text-red-600 hover:bg-red-50 p-1 rounded"

>

<Trash2 className="w-4 h-4" />

</button>

</div>

<div className="grid grid-cols-1 md:grid-cols-2 gap-4">

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Conta PGC

</label>

<select

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

value={outflow.pgcCode}

onChange={(e) => updateOutflow(outflow.id, 'pgcCode', e.target.value)}

>

{Object.entries(pgcMapping.costs).map(([code, info]) => (

<option key={code} value={code}>

{code} - {info.name}

</option>

))}

</select>

</div>

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Descrição

</label>

<input

type="text"

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

placeholder="Detalhe da saída"

value={outflow.description}

onChange={(e) => updateOutflow(outflow.id, 'description', e.target.value)}

/>

</div>

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Valor (Kz)

</label>

<input

type="number"

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

placeholder="0,00"

value={outflow.amount}

onChange={(e) => updateOutflow(outflow.id, 'amount', e.target.value)}

/>

</div>

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Data Programada

</label>

<input

type="date"

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

value={outflow.scheduledDate}

onChange={(e) => updateOutflow(outflow.id, 'scheduledDate', e.target.value)}

/>

</div>

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Prioridade

</label>

<select

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

value={outflow.priority}

onChange={(e) => updateOutflow(outflow.id, 'priority', parseInt(e.target.value))}

>

<option value="1">1 - Crítica (Folha, INSS)</option>

<option value="2">2 - Alta (Juros, Energia)</option>

<option value="3">3 - Média (Fornecedores)</option>

<option value="4">4 - Baixa (Adiável)</option>

</select>

</div>

<div>

<label className="block text-xs font-medium text-gray-600 mb-1">

Fornecedor

</label>

<input

type="text"

className="w-full px-3 py-2 border border-gray-300 rounded-md text-sm"

placeholder="Nome do fornecedor"

value={outflow.supplier}

onChange={(e) => updateOutflow(outflow.id, 'supplier', e.target.value)}

/>

</div>

</div>

{outflow.fromBudget && outflow.budgetAnnual && (

<div className="mt-3 pt-3 border-t border-gray-200">

<p className="text-xs text-gray-600">

Orçamento Anual: {formatCurrency(outflow.budgetAnnual)} |

Tipo: {outflow.frequency}

</p>

</div>

)}

</div>

))}

</div>

)}

</div>

);

case 5: *// Financiamento*

return (

<div className="space-y-6">

<h3 className="text-lg font-semibold text-gray-900">Necessidades de Financiamento</h3>

<div className="bg-yellow-50 border border-yellow-200 rounded-lg p-4">

<div className="flex items-start gap-3">

<AlertCircle className="w-5 h-5 text-yellow-600 mt-0.5" />

<div>

<p className="text-sm font-medium text-yellow-900">Análise Preliminar</p>

<p className="text-sm text-yellow-700 mt-1">

Com base nos dados inseridos, o sistema calculará automaticamente a necessidade de financiamento.

</p>

</div>

</div>

</div>

<div className="space-y-4">

<div>

<label className="block text-sm font-medium text-gray-700 mb-2">

Suprimentos ENDIAMA E.P. (Kz)

</label>

<input

type="number"

className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500"

placeholder="0,00"

value={formData.financing.endiama}

onChange={(e) => setFormData(prev => ({

...prev,

financing: { ...prev.financing, endiama: parseFloat(e.target.value) || 0 }

}))}

/>

<p className="text-xs text-gray-500 mt-1">Conta PGC: 80 - Suprimentos/Empréstimos</p>

</div>

<div>

<label className="block text-sm font-medium text-gray-700 mb-2">

Financiamento Bancário (Kz)

</label>

<input

type="number"

className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500"

placeholder="0,00"

value={formData.financing.bank}

onChange={(e) => setFormData(prev => ({

...prev,

financing: { ...prev.financing, bank: parseFloat(e.target.value) || 0 }

}))}

/>

</div>

<div>

<label className="block text-sm font-medium text-gray-700 mb-2">

Outras Fontes (Kz)

</label>

<input

type="number"

className="w-full px-4 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500"

placeholder="0,00"

value={formData.financing.other}

onChange={(e) => setFormData(prev => ({

...prev,

financing: { ...prev.financing, other: parseFloat(e.target.value) || 0 }

}))}

/>

</div>

</div>

</div>

);

case 6: *// Resumo e Validação*

const totals = calculateTotals();

return (

<div className="space-y-6">

<h3 className="text-lg font-semibold text-gray-900">Resumo do Plano de Tesouraria</h3>

{*/\* Referência ao Orçamento \*/*}

{formData.importedFromBudget && (

<div className="bg-indigo-50 border border-indigo-200 rounded-lg p-4">

<div className="flex items-center gap-3">

<Database className="w-5 h-5 text-indigo-600" />

<div>

<p className="text-sm font-medium text-indigo-900">Baseado no Orçamento Aprovado</p>

<p className="text-xs text-indigo-700 mt-1">

Documento: {formData.metadata.budgetReference} |

Importado em: {new Date(formData.metadata.budgetImportDate).toLocaleDateString('pt-AO')}

</p>

</div>

</div>

</div>

)}

{*/\* Cards de Resumo \*/*}

<div className="grid grid-cols-1 md:grid-cols-2 gap-4">

<div className="bg-white border border-gray-200 rounded-lg p-4">

<p className="text-sm text-gray-600 mb-2">Saldo Inicial</p>

<p className="text-2xl font-bold text-gray-900">{formatCurrency(formData.initialBalance)}</p>

</div>

<div className="bg-green-50 border border-green-200 rounded-lg p-4">

<p className="text-sm text-green-600 mb-2">Total de Entradas</p>

<p className="text-2xl font-bold text-green-700">{formatCurrency(totals.totalInflows)}</p>

<p className="text-xs text-green-600 mt-1">

{formData.inflows.length} lançamentos

</p>

</div>

<div className="bg-red-50 border border-red-200 rounded-lg p-4">

<p className="text-sm text-red-600 mb-2">Total de Saídas</p>

<p className="text-2xl font-bold text-red-700">{formatCurrency(totals.totalOutflows)}</p>

<p className="text-xs text-red-600 mt-1">

{formData.outflows.length} lançamentos

</p>

</div>

<div className="bg-blue-50 border border-blue-200 rounded-lg p-4">

<p className="text-sm text-blue-600 mb-2">Financiamento</p>

<p className="text-2xl font-bold text-blue-700">{formatCurrency(totals.totalFinancing)}</p>

</div>

</div>

{*/\* Resultado Final \*/*}

<div className={`rounded-lg p-6 ${

totals.finalBalance >= 0

? 'bg-gradient-to-r from-green-500 to-green-600'

: 'bg-gradient-to-r from-red-500 to-red-600'

} text-white`}>

<div className="flex justify-between items-center">

<div>

<p className="text-sm opacity-90">Fluxo Líquido do Período</p>

<p className="text-3xl font-bold mt-1">{formatCurrency(totals.netFlow)}</p>

</div>

<div className="text-right">

<p className="text-sm opacity-90">Saldo Final Previsto</p>

<p className="text-3xl font-bold mt-1">{formatCurrency(totals.finalBalance)}</p>

</div>

</div>

</div>

{*/\* Análise de Liquidez \*/*}

<div className="bg-gray-50 rounded-lg p-4">

<h4 className="font-medium text-gray-900 mb-3">Análise de Liquidez</h4>

<div className="space-y-2 text-sm">

<div className="flex justify-between">

<span className="text-gray-600">Pagamentos Críticos (Prioridade 1):</span>

<span className="font-medium text-red-700">{formatCurrency(totals.criticalPayments)}</span>

</div>

<div className="flex justify-between">

<span className="text-gray-600">Índice de Liquidez:</span>

<span className={`font-medium ${totals.liquidityRatio >= 1 ? 'text-green-700' : 'text-red-700'}`}>

{totals.liquidityRatio.toFixed(2)}

</span>

</div>

</div>

</div>

{*/\* Alertas \*/*}

{totals.finalBalance < 0 && (

<div className="bg-red-50 border border-red-200 rounded-lg p-4">

<div className="flex items-start gap-3">

<AlertCircle className="w-5 h-5 text-red-600 mt-0.5" />

<div>

<p className="text-sm font-medium text-red-900">Alerta: Saldo Negativo Previsto</p>

<p className="text-sm text-red-700 mt-1">

O plano indica um déficit de {formatCurrency(Math.abs(totals.finalBalance))}.

Considere aumentar o financiamento ou reduzir despesas não prioritárias.

</p>

</div>

</div>

</div>

)}

{*/\* Ações Finais \*/*}

<div className="flex gap-3">

<button className="flex-1 px-6 py-3 bg-gray-200 text-gray-700 rounded-lg hover:bg-gray-300 transition-colors">

Guardar Rascunho

</button>

<button className="flex-1 px-6 py-3 bg-gradient-to-r from-indigo-600 to-purple-600 text-white rounded-lg hover:opacity-90 transition-opacity flex items-center justify-center gap-2">

<Save className="w-5 h-5" />

Gerar Plano de Tesouraria

</button>

</div>

</div>

);

default:

return null;

}

};

return (

<div className="min-h-screen bg-gray-50 p-6">

<div className="max-w-5xl mx-auto">

<div className="bg-white rounded-2xl shadow-xl overflow-hidden">

{*/\* Header com Steps \*/*}

<div className="bg-gradient-to-r from-indigo-600 to-purple-600 px-8 py-6">

<h1 className="text-2xl font-bold text-white mb-2">

Formulário de Plano de Tesouraria Mensal

</h1>

<p className="text-indigo-100">Sistema Integrado com Orçamento Aprovado - Mapeamento PGC-AO</p>

{*/\* Progress Steps \*/*}

<div className="mt-6 flex items-center justify-between">

{steps.map((step, index) => (

<div key={step.id} className="flex items-center">

<div

className={`flex items-center justify-center w-10 h-10 rounded-full transition-all ${

currentStep === index

? 'bg-white text-indigo-600 scale-110 shadow-lg'

: currentStep > index

? 'bg-green-400 text-white'

: 'bg-indigo-400 text-white opacity-60'

}`}

>

<span className="text-lg">{step.icon}</span>

</div>

{index < steps.length - 1 && (

<div

className={`h-1 transition-all ${

currentStep > index ? 'bg-green-400' : 'bg-indigo-400 opacity-30'

}`}

style={{ width: '70px' }}

/>

)}

</div>

))}

</div>

<div className="mt-3 flex justify-between text-white text-xs">

{steps.map((step) => (

<div

key={step.id}

className={`text-center ${currentStep === step.id ? 'font-semibold' : 'opacity-70'}`}

style={{ width: '100px' }}

>

{step.title}

</div>

))}

</div>

</div>

{*/\* Content \*/*}

<div className="p-8">

{renderStepContent()}

</div>

{*/\* Navigation \*/*}

<div className="flex justify-between items-center px-8 py-4 bg-gray-50 border-t">

<button

onClick={() => setCurrentStep(Math.max(0, currentStep - 1))}

disabled={currentStep === 0}

className={`px-6 py-2 rounded-lg transition-colors ${

currentStep === 0

? 'bg-gray-200 text-gray-400 cursor-not-allowed'

: 'bg-gray-300 text-gray-700 hover:bg-gray-400'

}`}

>

Anterior

</button>

<div className="flex gap-2">

{steps.map((step) => (

<button

key={step.id}

onClick={() => setCurrentStep(step.id)}

className={`w-2 h-2 rounded-full transition-all ${

currentStep === step.id ? 'w-8 bg-indigo-600' : 'bg-gray-300'

}`}

/>

))}

</div>

{currentStep < steps.length - 1 ? (

<button

onClick={() => setCurrentStep(Math.min(steps.length - 1, currentStep + 1))}

className="px-6 py-2 bg-indigo-600 text-white rounded-lg hover:bg-indigo-700 transition-colors flex items-center gap-2"

>

Próximo

<ChevronRight className="w-4 h-4" />

</button>

) : (

<button

className="px-6 py-2 bg-green-600 text-white rounded-lg hover:bg-green-700 transition-colors flex items-center gap-2"

>

<FileText className="w-4 h-4" />

Finalizar

</button>

)}

</div>

</div>

</div>

</div>

);

};

export default TreasuryPlanForm;