**formulário para entrada de dados PARA O ORCAMENTO ANUAL**

**Formulário de Entrada de Dados (React)**

**Um wizard multi-passos que guia o usuário através de:**

* **Informações da empresa e setores**
* **Entrada de receitas/produtos**
* **Custos com materiais, serviços e pessoal**
* **Cadastro de ativos**
* **Configuração de sazonalidade**
* **Resumo final com métricas calculadas**

**veja:** [**Formulário de Entrada de Dados Orçamentários | Claude | Claude**](https://claude.ai/public/artifacts/f7a5ed1f-b4e8-47ff-b77d-986f5326b177)

**codigO:**

import React, { useState } from 'react';

import { ChevronRight, Plus, Trash2, Save, AlertCircle, CheckCircle } from 'lucide-react';

const BudgetInputForm = () => {

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

const [formData, setFormData] = useState({

companyInfo: {

name: '',

type: '',

sectors: []

},

revenues: [],

costs: {

materials: [],

services: [],

personnel: [],

fixed: []

},

assets: [],

seasonality: {

hasSeasonality: false,

months: []

}

});

const steps = [

{ id: 0, title: 'Informações da Empresa', icon: '🏢' },

{ id: 1, title: 'Receitas', icon: '💰' },

{ id: 2, title: 'Custos com Materiais', icon: '📦' },

{ id: 3, title: 'Custos com Serviços', icon: '🔧' },

{ id: 4, title: 'Custos com Pessoal', icon: '👥' },

{ id: 5, title: 'Ativos', icon: '🏭' },

{ id: 6, title: 'Sazonalidade', icon: '📊' },

{ id: 7, title: 'Revisão e Envio', icon: '✅' }

];

const addRevenue = () => {

setFormData(prev => ({

...prev,

revenues: [...prev.revenues, { description: '', quantity: 0, unitPrice: 0, total: 0 }]

}));

};

const updateRevenue = (index, field, value) => {

const newRevenues = [...formData.revenues];

newRevenues[index][field] = value;

if (field === 'quantity' || field === 'unitPrice') {

newRevenues[index].total = newRevenues[index].quantity \* newRevenues[index].unitPrice;

}

setFormData(prev => ({ ...prev, revenues: newRevenues }));

};

const removeRevenue = (index) => {

setFormData(prev => ({

...prev,

revenues: prev.revenues.filter((\_, i) => i !== index)

}));

};

const addCost = (type) => {

const newCost = type === 'personnel'

? { role: '', quantity: 1, salary: 0, type: 'Permanente', total: 0 }

: { description: '', quantity: 0, unitCost: 0, frequency: 'Mensal', total: 0 };

setFormData(prev => ({

...prev,

costs: {

...prev.costs,

[type]: [...prev.costs[type], newCost]

}

}));

};

const calculateTotals = () => {

const revenueTotal = formData.revenues.reduce((sum, r) => sum + (r.total || 0), 0);

const materialsCosts = formData.costs.materials.reduce((sum, c) => sum + (c.total || 0), 0);

const servicesCosts = formData.costs.services.reduce((sum, c) => sum + (c.total || 0), 0);

const personnelCosts = formData.costs.personnel.reduce((sum, c) => sum + (c.total || 0), 0);

const fixedCosts = formData.costs.fixed.reduce((sum, c) => sum + (c.total || 0), 0);

const totalCosts = materialsCosts + servicesCosts + personnelCosts + fixedCosts;

const netResult = revenueTotal - totalCosts;

const margin = revenueTotal > 0 ? (netResult / revenueTotal \* 100).toFixed(1) : 0;

return { revenueTotal, totalCosts, netResult, margin };

};

const renderStepContent = () => {

switch(currentStep) {

case 0:

return (

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

<div>

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

Nome da 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 focus:border-transparent"

placeholder="Digite o nome da empresa"

value={formData.companyInfo.name}

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

...prev,

companyInfo: { ...prev.companyInfo, name: e.target.value }

}))}

/>

</div>

<div>

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

Descrição do Negócio

</label>

<textarea

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

rows="4"

placeholder="Descreva a atividade principal da empresa..."

value={formData.companyInfo.type}

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

...prev,

companyInfo: { ...prev.companyInfo, type: e.target.value }

}))}

/>

</div>

<div>

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

Setores de Atuação

</label>

<div className="grid grid-cols-2 gap-3">

{['Agricultura', 'Pecuária', 'Indústria', 'Comércio', 'Serviços', 'Tecnologia'].map(sector => (

<label key={sector} className="flex items-center space-x-2">

<input

type="checkbox"

className="rounded text-indigo-600 focus:ring-indigo-500"

checked={formData.companyInfo.sectors.includes(sector)}

onChange={(e) => {

const sectors = e.target.checked

? [...formData.companyInfo.sectors, sector]

: formData.companyInfo.sectors.filter(s => s !== sector);

setFormData(prev => ({

...prev,

companyInfo: { ...prev.companyInfo, sectors }

}));

}}

/>

<span className="text-sm">{sector}</span>

</label>

))}

</div>

</div>

</div>

);

case 1:

return (

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

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

<h3 className="text-lg font-semibold">Produtos/Serviços e Receitas</h3>

<button

onClick={addRevenue}

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

>

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

Adicionar Produto

</button>

</div>

{formData.revenues.map((revenue, index) => (

<div key={index} className="p-4 border border-gray-200 rounded-lg space-y-3">

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

<div className="flex-1 grid grid-cols-1 md:grid-cols-4 gap-3">

<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="Nome do produto"

value={revenue.description}

onChange={(e) => updateRevenue(index, 'description', e.target.value)}

/>

</div>

<div>

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

Quantidade/Ano

</label>

<input

type="number"

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

placeholder="0"

value={revenue.quantity}

onChange={(e) => updateRevenue(index, 'quantity', parseFloat(e.target.value) || 0)}

/>

</div>

<div>

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

Preço Unitário (Kz)

</label>

<input

type="number"

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

placeholder="0"

value={revenue.unitPrice}

onChange={(e) => updateRevenue(index, 'unitPrice', parseFloat(e.target.value) || 0)}

/>

</div>

<div>

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

Total (Kz)

</label>

<div className="px-3 py-2 bg-gray-100 rounded-md text-sm font-semibold">

{revenue.total.toLocaleString('pt-AO')}

</div>

</div>

</div>

<button

onClick={() => removeRevenue(index)}

className="ml-3 p-2 text-red-600 hover:bg-red-50 rounded-lg transition-colors"

>

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

</button>

</div>

</div>

))}

{formData.revenues.length === 0 && (

<div className="text-center py-8 text-gray-500">

<p>Nenhum produto adicionado ainda.</p>

<p className="text-sm mt-2">Clique em "Adicionar Produto" para começar.</p>

</div>

)}

</div>

);

case 2:

case 3:

case 4:

case 5:

const costType = currentStep === 2 ? 'materials' :

currentStep === 3 ? 'services' :

currentStep === 4 ? 'personnel' : 'fixed';

const costTitle = currentStep === 2 ? 'Materiais e Matérias-Primas' :

currentStep === 3 ? 'Serviços Externos' :

currentStep === 4 ? 'Pessoal' : 'Ativos Fixos';

return (

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

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

<h3 className="text-lg font-semibold">{costTitle}</h3>

<button

onClick={() => addCost(costType)}

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

>

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

Adicionar Item

</button>

</div>

{costType === 'personnel' ? (

formData.costs.personnel.map((person, index) => (

<div key={index} className="p-4 border border-gray-200 rounded-lg">

<div className="grid grid-cols-1 md:grid-cols-4 gap-3">

<input

type="text"

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

placeholder="Cargo/Função"

value={person.role}

/>

<input

type="number"

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

placeholder="Quantidade"

value={person.quantity}

/>

<input

type="number"

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

placeholder="Salário Mensal (Kz)"

value={person.salary}

/>

<select className="px-3 py-2 border border-gray-300 rounded-md text-sm">

<option>Permanente</option>

<option>Temporário</option>

<option>Freelancer</option>

</select>

</div>

</div>

))

) : (

<div className="text-center py-8 text-gray-500">

<p>Configuração para {costTitle}</p>

</div>

)}

</div>

);

case 6:

return (

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

<h3 className="text-lg font-semibold">Sazonalidade do Negócio</h3>

<div className="flex items-center space-x-3">

<input

type="checkbox"

className="rounded text-indigo-600"

checked={formData.seasonality.hasSeasonality}

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

...prev,

seasonality: { ...prev.seasonality, hasSeasonality: e.target.checked }

}))}

/>

<label className="text-sm font-medium">

O negócio possui variações sazonais significativas

</label>

</div>

{formData.seasonality.hasSeasonality && (

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

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

Indique a distribuição percentual esperada de receitas por mês:

</p>

<div className="grid grid-cols-3 md:grid-cols-4 gap-3">

{['Jan', 'Fev', 'Mar', 'Abr', 'Mai', 'Jun', 'Jul', 'Ago', 'Set', 'Out', 'Nov', 'Dez'].map((month, index) => (

<div key={month}>

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

<input

type="number"

className="w-full px-2 py-1 border border-gray-300 rounded text-sm"

placeholder="8.33%"

min="0"

max="100"

/>

</div>

))}

</div>

</div>

)}

</div>

);

case 7:

const totals = calculateTotals();

return (

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

<h3 className="text-lg font-semibold">Resumo do Orçamento</h3>

<div className="bg-gradient-to-r from-indigo-500 to-purple-600 rounded-xl p-6 text-white">

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

<div>

<p className="text-sm opacity-90">Receita Total</p>

<p className="text-2xl font-bold">{totals.revenueTotal.toLocaleString('pt-AO')} Kz</p>

</div>

<div>

<p className="text-sm opacity-90">Custos Totais</p>

<p className="text-2xl font-bold">{totals.totalCosts.toLocaleString('pt-AO')} Kz</p>

</div>

<div>

<p className="text-sm opacity-90">Resultado Líquido</p>

<p className="text-2xl font-bold">{totals.netResult.toLocaleString('pt-AO')} Kz</p>

</div>

<div>

<p className="text-sm opacity-90">Margem</p>

<p className="text-2xl font-bold">{totals.margin}%</p>

</div>

</div>

</div>

<div className="space-y-3">

<div className="flex items-start gap-3 p-4 bg-green-50 border border-green-200 rounded-lg">

<CheckCircle className="w-5 h-5 text-green-600 mt-0.5" />

<div>

<p className="font-medium text-green-900">Dados prontos para processamento</p>

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

O sistema irá mapear automaticamente para o PGC-AO e gerar relatórios completos

</p>

</div>

</div>

<div className="flex items-start gap-3 p-4 bg-blue-50 border border-blue-200 rounded-lg">

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

<div>

<p className="font-medium text-blue-900">Mapeamento inteligente</p>

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

Contas com baixa confiança serão sinalizadas para revisão manual

</p>

</div>

</div>

</div>

<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" />

Processar e Gerar Relatórios

</button>

</div>

</div>

);

default:

return null;

}

};

return (

<div className="min-h-screen bg-gradient-to-br from-indigo-50 to-purple-50 p-6">

<div className="max-w-6xl 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-6">

Sistema de Entrada de Dados Orçamentários

</h1>

<div className="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'

: 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={`w-full h-1 mx-2 transition-all ${

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

}`}

style={{ width: '60px' }}

/>

)}

</div>

))}

</div>

<div className="mt-4 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>

<button

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

disabled={currentStep === steps.length - 1}

className={`px-6 py-2 rounded-lg transition-colors flex items-center gap-2 ${

currentStep === steps.length - 1

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

: 'bg-indigo-600 text-white hover:bg-indigo-700'

}`}

>

Próximo

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

</button>

</div>

</div>

</div>

</div>

);

};

export default BudgetInputForm;