```
import { useState, useEffect } from "react";
import QRCode from "react-qr-code";
import { loadStripe } from "@stripe/stripe-js";
import { Elements, PaymentElement, useStripe, useElements, Payimport { useState,
useEffect } from "react";
import QRCode from "react-qr-code";
import { loadStripe } from "@stripe/stripe-js";
import { Elements, PaymentElement, useStripe, useElements,
PaymentRequestButtonElement } from "@stripe/react-stripe-js";
import { useToast } from "@/hooks/use-toast";
import PatrocinadoresCarousel from "@/components/PatrocinadoresCarousel";
import logoClube from "@assets/LOGO_CLUBE-05_1752081350082.png";

const stripePromise = loadStripe(import.meta.env.VITE_STRIPE_PUBLIC_KEY ||
"pk_test_51RdaS1Qlsea8vAKZC1WmSHcCGXNGGTxJuLZ3iq90MUpeCxq5CUhj5C2Qwm
HWO008hWIMSaZ0yh75EzrSUpXyvTs6002cYD8L9l");

// Componente de Carrossel usando o componente unificado
function BannerCarousel({ compact = false, showTitle = false }: { compact?: boolean;
showTitle?: boolean }) {
  return <PatrocinadoresCarousel showTitle={showTitle} className={compact ? "mt-6 mb-4"
: "mb-6"} />;
}

// Componente de pagamento Stripe
function StripePaymentModal({ clientSecret, onSuccess, onCancel, nomeComprador,
telefoneComprador, quantidade, valorTotal }: {
  clientSecret: string;
  onSuccess: () => void;
  onCancel: () => void;
  nomeComprador: string;
  telefoneComprador: string;
  quantidade: number;
  valorTotal: number;
}) {
  const stripe = useStripe();
  const elements = useElements();
  const [isProcessing, setIsProcessing] = useState(false);
  const [paymentRequest, setPaymentRequest] = useState<any>(null);

  // Configurar Apple Pay / Google Pay
  useEffect(() => {
    if (!stripe) return;

    const pr = stripe.paymentRequest({
      country: 'BR',
      currency: 'brl',
      total: {
```

```
      label: `Ingresso${quantidade > 1 ? 's' : ''} Premium - Instituto O Grito`,
      amount: valorTotal, // Valor total em centavos
    },
    requestPayerName: true,
    requestPayerEmail: true,
  });

  pr.canMakePayment().then(result => {
    if (result) {
      setPaymentRequest(pr);
    }
  });

  pr.on('paymentmethod', async (ev) => {
    setIsProcessing(true);
    try {
      const { error } = await stripe.confirmCardPayment(clientSecret, {
        payment_method: ev.paymentMethod.id
      });

      if (error) {
        ev.complete('fail');
        alert('Erro no pagamento: ' + error.message);
      } else {
        ev.complete('success');
        onSuccess();
      }
    } catch (error) {
      ev.complete('fail');
      alert('Erro no pagamento');
    }
    setIsProcessing(false);
  });
}, [stripe, clientSecret, onSuccess]);

const handleSubmit = async (event: React.FormEvent) => {
  event.preventDefault();

  if (!stripe || !elements) {
    return;
  }

  setIsProcessing(true);

  try {
    const { error } = await stripe.confirmPayment({
      elements,
      confirmParams: {
```

```
          return_url: window.location.origin,
        },
        redirect: 'if_required'
      });

      if (error) {
        alert('Erro no pagamento: ' + error.message);
      } else {
        onSuccess();
      }
    } catch (error) {
      alert('Erro no pagamento');
    }

    setIsProcessing(false);
  };

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50
p-4">
      <div className="bg-white rounded-xl p-6 max-w-md w-full max-h-[90vh]
overflow-y-auto">
        <div className="flex justify-between items-center mb-4">
          <h2 className="text-xl font-bold">Pagamento com Cartão</h2>
          <button
            onClick={onCancel}
            className="text-gray-500 hover:text-gray-700 text-2xl"
            data-testid="button-fechar-modal"
          >
            ×
          </button>
        </div>

        <div className="mb-4 p-3 bg-gray-50 rounded-lg">
          <p className="text-sm text-gray-600">Ingresso para: {nomeComprador}</p>
          <p className="text-sm text-gray-600">Telefone: {telefoneComprador}</p>
          <p className="text-sm text-gray-600">Quantidade: {quantidade}
ingresso{quantidade > 1 ? 's' : ''}</p>
          <p className="font-bold">Valor Total: R$ {(valorTotal / 100).toLocaleString('pt-BR', {
minimumFractionDigits: 2, maximumFractionDigits: 2 })}</p>
        </div>

        {/* Apple Pay / Google Pay */}
        {paymentRequest && (
          <div className="mb-4">
            <PaymentRequestButtonElement
              options={{ paymentRequest }}
              className="stripe-payment-request-button"
```

```
          />
          <div className="text-center my-4 text-gray-500">ou</div>
        </div>
      )}

      {/* Formulário de cartão */}
      <form onSubmit={handleSubmit}>
        <div className="mb-4">
          <PaymentElement />
        </div>

        <button
          type="submit"
          disabled={isProcessing || !stripe || !elements}
          className="w-full bg-yellow-500 text-black py-3 rounded-xl hover:bg-yellow-600
transition font-semibold disabled:bg-gray-300"
          data-testid="button-confirmar-pagamento"
        >
          {isProcessing ? 'Processando...' : `Confirmar Pagamento R$ ${(valorTotal /
100).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}`}
        </button>
      </form>
    </div>
  </div>
 );
}

export default function PagamentoIngresso() {
 const { toast } = useToast();
 const [nome, setNome] = useState("");
 const [telefone, setTelefone] = useState("");
 const [email, setEmail] = useState("");
 const [etapa, setEtapa] = useState<'card-inicial' | 'opcoes-pagamento' | 'formulario-pix' |
'qual-empresa' | 'opcoes-resgate' | 'form-dados-convite' | 'qr-code' | 'formulario-cartao' |
'escolher-metodo-pagamento' | 'pagamento-cartao' | 'pagamento-rede' |
'verificando-pagamento' | 'associar-empresa' | 'cadastrar-usuario' | 'confirmar-vinculo' |
'formulario-rede' | 'formulario-cielo'>('card-inicial');
 const [clientSecret, setClientSecret] = useState("");
 const [isCreatingPayment, setIsCreatingPayment] = useState(false);
 const [showPixFallback, setShowPixFallback] = useState(false);
 const [quantidadeIngressos, setQuantidadeIngressos] = useState(1);

 // Estados para formulário Rede
 const [redeQty, setRedeQty] = useState(1);
 const [redeInstallments, setRedeInstallments] = useState(1);
 const [redeCardNumber, setRedeCardNumber] = useState("");
 const [redeCardName, setRedeCardName] = useState("");
 const [redeCardMonth, setRedeCardMonth] = useState("");
```

```
  const [redeCardYear, setRedeCardYear] = useState("");
  const [redeCardCvv, setRedeCardCvv] = useState("");

  // Estados para formulário Cielo
  const [cieloQty, setCieloQty] = useState(1);
  const [cieloInstallments, setCieloInstallments] = useState(1);
  const [cieloCardNumber, setCieloCardNumber] = useState("");
  const [cieloCardName, setCieloCardName] = useState("");
  const [cieloCardMonth, setCieloCardMonth] = useState("");
  const [cieloCardYear, setCieloCardYear] = useState("");
  const [cieloCardCvv, setCieloCardCvv] = useState("");

  // Timer e verificação de pagamento
  const [timerSegundos, setTimerSegundos] = useState(60);
  const [timerIniciado, setTimerIniciado] = useState(false);
  const [paymentIntentId, setPaymentIntentId] = useState("");

  // Estados para sistema de cotas empresariais
  const [nomeEmpresa, setNomeEmpresa] = useState("");
  const [emailEmpresa, setEmailEmpresa] = useState("");
  const [cotaEmpresa, setCotaEmpresa] = useState<any>(null);
  const [isValidatingEmpresa, setIsValidatingEmpresa] = useState(false);
  const [isResgatandoIngresso, setIsResgatandoIngresso] = useState(false);
  const [temEmpresa, setTemEmpresa] = useState<boolean | null>(null);

  // Dados do convite (para "Colocar Nome no Convite")
  const [nomeConvite, setNomeConvite] = useState("");
  const [emailConvite, setEmailConvite] = useState("");
  const [telefoneConvite, setTelefoneConvite] = useState("");

  // Estado para TXID PIX (identificador único do pagamento)
  const [pixTxid, setPixTxid] = useState("");

  // Payload Pix oficial (CNPJ: 28.790.664/0001-10 | Valor: R$ 1.000,00)
  const pixPayload =

"00020126360014BR.GOV.BCB.PIX011428790664000110520400005303986540610000.005
802BR5917INSTITUTO O GRITO6008RIBEIRAO62070503***63045B06";

  // Timer de 1 minuto para verificar pagamento PIX
  useEffect(() => {
    let interval: NodeJS.Timeout;

    if (timerIniciado && timerSegundos > 0) {
      interval = setInterval(() => {
        setTimerSegundos((prev) => prev - 1);
      }, 1000);
    }
```

```
  return () => clearInterval(interval);
}, [timerIniciado, timerSegundos]);

const verificarPagamento = async () => {
  setEtapa('verificando-pagamento');

  try {
    const response = await fetch('/api/verificar-pagamento-stripe', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        nome,
        telefone,
        paymentIntentId
      }),
    });

    const resultado = await response.json();

    if (resultado.pago) {
      // Pagamento confirmado, ir para tela de cadastro
      setEtapa('cadastrar-usuario');
    } else {
      toast({
        title: "Pagamento não detectado",
        description: "Por favor, tente novamente ou entre em contato.",
        variant: "destructive",
      });
      setEtapa('qr-code');
      setTimerSegundos(60);
      setTimerIniciado(false);
    }
  } catch (error) {
    console.error('Erro ao verificar pagamento:', error);
    toast({
      title: "Erro ao verificar pagamento",
      description: "Por favor, tente novamente.",
      variant: "destructive",
    });
    setEtapa('qr-code');
    setTimerSegundos(60);
    setTimerIniciado(false);
  }
};

const handlePixFallback = () => {
  setShowPixFallback(true);
```

```
  // Copiar automaticamente quando mostra fallback
  navigator.clipboard.writeText(pixPayload);
};

const mostrarOpcoesPagamento = () => {
  setEtapa('opcoes-pagamento');
};

const handlePixClick = async () => {
  // Verificar se temos os dados necessários
  if (!nome || !telefone) {
    setEtapa('formulario-pix');
    return;
  }

  setIsCreatingPayment(true);

  try {
    // Criar registro PIX pendente
    const response = await fetch('/api/ingressos/pix/iniciar', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        nome,
        telefone,
        email: email || undefined,
        quantidade: 1 // PIX sempre 1 ingresso
      }),
    });

    if (!response.ok) {
      throw new Error('Erro ao criar registro PIX');
    }

    const data = await response.json();

    // Salvar TXID e ir para tela de QR code
    setPixTxid(data.txid);
    setQuantidadeIngressos(1);
    setEtapa('qr-code');

    console.log(`✅ Registro PIX criado com TXID: ${data.txid}`);

  } catch (error) {
    console.error('❌ Erro ao criar registro PIX:', error);
    toast({
      title: "Erro ao gerar PIX",
      description: "Tente novamente ou escolha outro método de pagamento.",
```

```
      variant: "destructive",
    });
  } finally {
    setIsCreatingPayment(false);
  }
};

const handleValidarEmpresa = async (e: React.FormEvent) => {
  e.preventDefault();

  if (!nomeEmpresa.trim() || !emailEmpresa.trim()) {
    alert('Por favor, preencha nome e e-mail da empresa');
    return;
  }

  setIsValidatingEmpresa(true);

  try {
    const response = await fetch('/api/cotas/validar-empresa', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        nomeEmpresa: nomeEmpresa.trim(),
        email: emailEmpresa.trim()
      }),
    });

    const resultado = await response.json();

    if (resultado.valida && resultado.cota) {
      setCotaEmpresa(resultado.cota);
      setEtapa('opcoes-resgate');
    } else {
      alert(resultado.mensagem || 'Dados incorretos ou empresa sem convites disponíveis');
    }
  } catch (error) {
    console.error('Erro ao validar empresa:', error);
    alert('Erro ao buscar empresa. Tente novamente.');
  } finally {
    setIsValidatingEmpresa(false);
  }
};

const handleRetirarAgora = async () => {
  if (!cotaEmpresa) return;

  setIsResgatandoIngresso(true);
```

```
    try {
      const response = await fetch('/api/cotas/resgatar-ingresso', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          idCota: cotaEmpresa.id
        }),
      });

      const resultado = await response.json();

      if (resultado.success && resultado.ingresso) {
        // Salvar dados para navegação de volta
        localStorage.setItem('ultimoResgateCotaId', cotaEmpresa.id.toString());
        localStorage.setItem('ingressoResgatadoNumero', resultado.ingresso.numero);

        console.log('🎯 [RETIRAR-AGORA] Redirecionando para:',
`/ingresso/visualizar/${resultado.ingresso.numero}`);

        // Redirecionar para visualização do ingresso usando o número
        window.location.href = `/ingresso/visualizar/${resultado.ingresso.numero}`;
      } else {
        alert(resultado.error || 'Erro ao resgatar ingresso');
      }
    } catch (error) {
      console.error('Erro ao resgatar ingresso:', error);
      alert('Erro ao resgatar ingresso. Tente novamente.');
    } finally {
      setIsResgatandoIngresso(false);
    }
  };

  const handleColocarNome = () => {
    setEtapa('form-dados-convite');
  };

  const handleSubmitDadosConvite = async (e: React.FormEvent) => {
    e.preventDefault();

    if (!cotaEmpresa) return;

    setIsResgatandoIngresso(true);

    try {
      const response = await fetch('/api/cotas/resgatar-ingresso', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
```

```
          idCota: cotaEmpresa.id,
          nomeComprador: nomeConvite.trim(),
          emailComprador: emailConvite.trim(),
          telefoneComprador: telefoneConvite.trim()
        }),
      });

      const resultado = await response.json();

      if (resultado.success && resultado.ingresso) {
        // Salvar dados para navegação de volta
        localStorage.setItem('ultimoResgateCotaId', cotaEmpresa.id.toString());
        localStorage.setItem('ingressoResgatadoNumero', resultado.ingresso.numero);

        console.log('🎯 [PAGAMENTO-INGRESSO] Redirecionando para:',
`/ingresso/visualizar/${resultado.ingresso.numero}`);

        // Redirecionar para visualização do ingresso usando o número
        window.location.href = `/ingresso/visualizar/${resultado.ingresso.numero}`;
      } else {
        alert(resultado.error || 'Erro ao resgatar ingresso');
      }
    } catch (error) {
      console.error('Erro ao resgatar ingresso:', error);
      alert('Erro ao resgatar ingresso. Tente novamente.');
    } finally {
      setIsResgatandoIngresso(false);
    }
  };

  const handleFormSubmit = async (e: React.FormEvent) => {
    e.preventDefault();

    try {
      // Enviar dados para o backend
      const response = await fetch('/api/patrocinador-2026', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ nome, email, telefone }),
      });

      if (!response.ok) {
        throw new Error('Erro ao salvar dados');
      }

      const data = await response.json();
```

```
    console.log('✅ Patrocinador 2026 cadastrado:', data);

    // Prosseguir para o QR code
    setEtapa('qr-code');
  } catch (error) {
    console.error('❌ Erro ao cadastrar patrocinador:', error);
    // Ainda assim mostrar o QR code (não bloquear pagamento)
    setEtapa('qr-code');
  }
};

const handleCartaoClick = async () => {
  // Primeiro, coletar dados do comprador
  if (!nome || !telefone) {
    // Se não temos dados, ir para formulário de cartão primeiro
    setEtapa('formulario-cartao');
    return;
  }

  setIsCreatingPayment(true);

  try {
    // Salvar dados do patrocinador 2026
    await fetch('/api/patrocinador-2026', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ nome, email, telefone }),
    });

    // Criar PaymentIntent para o ingresso (valor unitário R$ 1.000,00 * quantidade)
    const valorUnitario = 100000; // R$ 1.000,00 em centavos
    const valorTotal = valorUnitario * quantidadeIngressos;

    const response = await fetch('/api/ingresso/create-payment', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        nome,
        telefone,
        amount: valorTotal,
        quantidade: quantidadeIngressos
      }),
    });

    if (!response.ok) {
      throw new Error('Erro ao criar pagamento');
    }
```

```
    const { clientSecret } = await response.json();
    setClientSecret(clientSecret);
    setEtapa('pagamento-cartao');

  } catch (error) {
    console.error('❌ Erro ao criar pagamento:', error);
    toast({
      title: "Erro ao iniciar pagamento",
      description: "Por favor, tente novamente.",
      variant: "destructive",
    });
  } finally {
    setIsCreatingPayment(false);
  }
};

const handleRedeClick = async () => {
  // Primeiro, coletar dados do comprador
  if (!nome || !telefone) {
    // Se não temos dados, ir para formulário de cartão primeiro
    setEtapa('formulario-cartao');
    return;
  }

  // Ir para formulário de Rede (novo fluxo)
  setEtapa('formulario-rede');
};

const handleRedeSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setIsCreatingPayment(true);

  try {
    // Salvar dados do patrocinador 2026
    await fetch('/api/patrocinador-2026', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ nome, email, telefone }),
    });

    // Processar pagamento com Rede
    const response = await fetch('/api/pagamentos/rede/checkout', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        qty: redeQty,
        installments: redeInstallments,
        card: {
```

```
        cardNumber: redeCardNumber.replace(/\s/g, ''),
        cardholderName: redeCardName,
        expirationMonth: redeCardMonth,
        expirationYear: redeCardYear,
        securityCode: redeCardCvv
      },
      nome,
      telefone
    }),
  });

  const result = await response.json();

  if (result.ok) {
    // Verificar se há 3DS (redirectUrl para autenticação)
    if (result.data?.threeDSecure?.url) {
      toast({
        title: "Autenticação necessária",
        description: "Redirecionando para autenticação 3DS...",
      });

      // Redirecionar para 3DS
      setTimeout(() => {
        window.location.href = result.data.threeDSecure.url;
      }, 500);
    } else {
      // Pagamento direto aprovado
      toast({
        title: "Pagamento processado!",
        description: `Pedido ${result.orderId} criado com sucesso.`,
      });

      setTimeout(() => {
        window.location.href = `/ingresso/sucesso?orderId=${result.orderId}`;
      }, 500);
    }
  } else {
    throw new Error(result.error || 'Erro ao processar pagamento');
  }
} catch (error: any) {
  console.error('❌ Erro ao processar Rede:', error);
  toast({
    title: "Erro ao processar pagamento",
    description: error.message || "Tente novamente.",
    variant: "destructive",
  });
} finally {
  setIsCreatingPayment(false);
```

```
  }
};

const handleCieloClick = async () => {
  // Primeiro, coletar dados do comprador
  if (!nome || !telefone || !email) {
    // Se não temos dados, ir para formulário de cartão primeiro
    setEtapa('formulario-cartao');
    return;
  }

  setIsCreatingPayment(true);

  try {
    // Criar pedido no backend antes de abrir o Checkout Cielo
    const response = await fetch('/api/pagamentos/cielo-checkout/criar', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        qty: quantidadeIngressos,
        nome,
        email,
        telefone
      })
    });

    const data = await response.json();

    if (data.ok && data.checkoutUrl) {
      console.log('🔵 [CIELO CHECKOUT] Abrindo checkout:', data.checkoutUrl);

      // Resetar estado de carregamento ANTES de redirecionar
      setIsCreatingPayment(false);

      // Redirecionar para Checkout Cielo na mesma página
      window.location.href = data.checkoutUrl;
    } else {
      setIsCreatingPayment(false);
      toast({
        title: "Erro ao processar pagamento",
        description: data.error || "Tente novamente.",
        variant: "destructive",
      });
    }
  } catch (error: any) {
    setIsCreatingPayment(false);
    console.error('❌ [CIELO CHECKOUT] Erro:', error);
    toast({
```

```
      title: "Erro ao processar pagamento",
      description: error.message || "Tente novamente.",
      variant: "destructive",
    });
  }
};

const handleCieloSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setIsCreatingPayment(true);

  try {
    // Salvar dados do patrocinador 2026
    await fetch('/api/patrocinador-2026', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ nome, email, telefone }),
    });

    // Processar pagamento com Cielo
    const response = await fetch('/api/pagamentos/cielo/checkout', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        qty: cieloQty,
        installments: cieloInstallments,
        card: {
          number: cieloCardNumber.replace(/\s/g, ''),
          holder: cieloCardName,
          expirationMonth: cieloCardMonth,
          expirationYear: cieloCardYear,
          securityCode: cieloCardCvv
        },
        nome,
        telefone
      }),
    });

    const result = await response.json();

    if (result.ok) {
      // Verificar se há 3DS (returnUrl para autenticação)
      if (result.returnUrl) {
        toast({
          title: "Autenticação necessária",
          description: "Redirecionando para autenticação 3DS...",
        });
```

```
        // Redirecionar para 3DS
        setTimeout(() => {
          window.location.href = result.returnUrl;
        }, 500);
      } else {
        // Pagamento direto aprovado
        toast({
          title: "Pagamento processado!",
          description: `Pedido ${result.orderId} criado com sucesso.`,
        });

        setTimeout(() => {
          window.location.href = `/ingresso/sucesso?orderId=${result.orderId}`;
        }, 500);
      }
    } else {
      throw new Error(result.message || 'Erro ao processar pagamento');
    }
  } catch (error: any) {
    console.error('❌ Erro ao processar Cielo:', error);
    toast({
      title: "Erro ao processar pagamento",
      description: error.message || "Tente novamente.",
      variant: "destructive",
    });
  } finally {
    setIsCreatingPayment(false);
  }
};

const voltarInicial = () => {
  setEtapa('card-inicial');
  setNome("");
  setTelefone("");
  setEmail("");
  setClientSecret(""); // Limpar clientSecret do Stripe
  setIsCreatingPayment(false); // Resetar estado de carregamento
};

const voltarOpcoes = () => {
  setEtapa('opcoes-pagamento');
};

return (
  <div className="min-h-screen bg-white" style={{ fontFamily: 'Inter, system-ui, sans-serif'
}}>
    <div className="container mx-auto max-w-md px-4 py-8">
```

```
      {/* Header com Logo */}
      <div className="text-center mb-6">
        <img
          src={logoClube}
          alt="Clube do Grito"
          className="h-32 mx-auto mb-4"
          data-testid="logo-clube"
        />

        {/* Carrossel sem título "Patrocinadores" */}
        {etapa === 'card-inicial' && <BannerCarousel showTitle={false} />}
      </div>

      {/* Card Unificado - Apoie o Grito + Benefícios do Ingresso */}
      {etapa === 'card-inicial' && (
        <div className="bg-black border-t-4 border-[#FFCC00] rounded-xl p-8 text-white">
          {/* Seção "Apoie todo mês" */}
          <div className="mb-8">
            <h2 className="text-xl font-semibold text-white text-center mb-4"
data-testid="titulo-apoie-todo-mes">
              Seu Grito, nossa VOZ!
            </h2>
            <p className="text-white text-justify leading-relaxed text-sm sm:text-base mb-6"
data-testid="texto-apoie-todo-mes">
              Ao realizar a compra desse convite, você contribui mensalmente com R$100,00
reais durante 10 meses para manter de pé nossas ações de acesso a cultura, esporte,
inclusão produtiva, socioemocional, educação e muito mais.
            </p>
          </div>

          {/* Linha divisória sutil */}
          <div className="border-t border-[#FFCC00] opacity-40 mb-6"></div>

          {/* Seção sobre o evento */}
          <div className="mt-4 mb-6">
            <h3 className="text-lg font-semibold text-white mb-2"
data-testid="titulo-sobre-encontro">
              Sobre o Encontro do Clube do Grito
            </h3>
            <p className="text-white text-justify leading-relaxed text-sm sm:text-base mb-4"
data-testid="texto-sobre-encontro">
              Uma noite para celebrar <span className="font-bold">quem apoia a
transformação</span>. Um encontro para se aproximar do nosso trabalho, o impacto
gerado, conectar parceiros e <span className="font-bold">arrecadar recursos</span> para
manter as oficinas o ano inteiro em 2026.
            </p>
          </div>
```

```
          {/* Botões de pagamento - movidos para cima */}
          <div className="space-y-4 mb-8">
            {/* Botão Patrocinadores */}
            <button
              onClick={() => setEtapa('qual-empresa')}
              className="w-full bg-black text-white py-4 rounded-xl hover:bg-gray-800
transition font-semibold text-lg border-2 border-white"
              data-testid="button-patrocinadores"
            >
              Patrocinadores
            </button>

            {/* Botão Ingresso Avulso */}
            <button
              onClick={() => {
                setQuantidadeIngressos(2);
                setEtapa('formulario-cartao'); // Ir direto para formulário (sem criar Stripe)
              }}
              disabled={isCreatingPayment}
              className="w-full bg-yellow-500 text-black py-4 rounded-xl hover:bg-yellow-600
transition font-semibold text-lg disabled:bg-gray-300"
              data-testid="button-pagar-cartao"
            >
              {isCreatingPayment ? 'Preparando pagamento...' : 'Comprar Ingresso Avulso'}
            </button>

          </div>

          {/* Seção "O que está incluso na experiência:" */}
          <div className="mt-8">
            <h3 className="text-lg font-semibold text-white mb-3"
data-testid="titulo-beneficios">
              O que está incluso na experiência:
            </h3>
            <ul className="space-y-3 text-white text-sm sm:text-base">
              <li className="flex items-start">
                <span className="text-[#FFCC00] mr-2 flex-shrink-0">✓</span>
                <span>Jantar exclusivo com menu especial</span>
              </li>
              <li className="flex items-start">
                <span className="text-[#FFCC00] mr-2 flex-shrink-0">✓</span>
                <span>Networking com convidados especiais</span>
              </li>
              <li className="flex items-start">
                <span className="text-[#FFCC00] mr-2 flex-shrink-0">✓</span>
                <span>Experiência musical e muita alegria</span>
              </li>
            </ul>
```

```
          </div>
        </div>
      )}

      {/* Tela: Qual empresa? */}
      {etapa === 'qual-empresa' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-qual-empresa">
            Qual empresa?
          </h2>

          <form onSubmit={handleValidarEmpresa} className="space-y-4">
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Nome da
Empresa</label>
              <input
                type="text"
                value={nomeEmpresa}
                onChange={(e) => setNomeEmpresa(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="Digite o nome da empresa"
                data-testid="input-nome-empresa"
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Insira e-mail
cadastrado</label>
              <input
                type="email"
                value={emailEmpresa}
                onChange={(e) => setEmailEmpresa(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="Digite o e-mail cadastrado"
                data-testid="input-email-empresa"
              />
            </div>

            <div className="space-y-3 pt-4">
              <button
                type="submit"
                disabled={isValidatingEmpresa}
```

```
              className="w-full bg-yellow-500 text-black py-4 rounded-xl hover:bg-yellow-600
transition font-semibold text-lg disabled:bg-gray-300"
              data-testid="button-buscar-empresa"
            >
              {isValidatingEmpresa ? 'Buscando...' : 'Buscar Empresa'}
            </button>

            <button
              type="button"
              onClick={voltarInicial}
              className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
              data-testid="button-voltar"
            >
              Voltar
            </button>
          </div>
        </form>

        {/* Carrossel compacto de patrocinadores */}
        <BannerCarousel compact={true} />
      </div>
    )}

    {/* Tela: Opções de Resgate */}
    {etapa === 'opcoes-resgate' && cotaEmpresa && (
      <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
        <h2 className="text-2xl font-bold text-center mb-4"
data-testid="titulo-opcoes-resgate">
          {cotaEmpresa.nomeEmpresa}
        </h2>

        <div className="bg-gray-50 rounded-lg p-4 mb-6">
          <div className="flex justify-between items-center mb-2">
            <span className="text-gray-700 font-medium">Total de Convites:</span>
            <span className="text-2xl font-bold
text-gray-900">{cotaEmpresa.quantidadeTotal}</span>
          </div>
          <div className="flex justify-between items-center mb-2">
            <span className="text-gray-700 font-medium">Já Retirados:</span>
            <span className="text-2xl font-bold
text-yellow-600">{cotaEmpresa.quantidadeUsada}</span>
          </div>
          <div className="flex justify-between items-center pt-2 border-t border-gray-300">
            <span className="text-gray-700 font-medium">Disponíveis:</span>
            <span className="text-2xl font-bold
text-green-600">{cotaEmpresa.quantidadeTotal - cotaEmpresa.quantidadeUsada}</span>
          </div>
```

```
          </div>

          <div className="space-y-3">
            <button
              onClick={handleRetirarAgora}
              disabled={isResgatandoIngresso}
              className="w-full bg-black text-white py-4 rounded-xl hover:bg-gray-800
transition font-semibold text-lg border-2 border-black disabled:bg-gray-400"
              data-testid="button-retirar-agora"
            >
              {isResgatandoIngresso ? 'Resgatando...' : 'Retirar Agora'}
            </button>

            <button
              onClick={handleColocarNome}
              disabled={isResgatandoIngresso}
              className="w-full bg-yellow-500 text-black py-4 rounded-xl hover:bg-yellow-600
transition font-semibold text-lg disabled:bg-gray-300"
              data-testid="button-colocar-nome"
            >
              Colocar Nome no Convite
            </button>

            <button
              type="button"
              onClick={() => {
                setCotaEmpresa(null);
                setNomeEmpresa("");
                setEtapa('qual-empresa');
              }}
              className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
              data-testid="button-voltar-empresa"
            >
              Voltar
            </button>
          </div>

          {/* Carrossel compacto de patrocinadores */}
          <BannerCarousel compact={true} />
        </div>
      )}

      {/* Tela: Formulário de Dados do Convite */}
      {etapa === 'form-dados-convite' && cotaEmpresa && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-4"
data-testid="titulo-dados-convite">
```

```
            Dados do Convite
          </h2>

          <p className="text-center text-gray-600 mb-6">
            {cotaEmpresa.nomeEmpresa} - Preencha os dados de quem irá usar este convite
          </p>

          <form onSubmit={handleSubmitDadosConvite} className="space-y-4">
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Nome
Completo</label>
              <input
                type="text"
                value={nomeConvite}
                onChange={(e) => setNomeConvite(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="Digite o nome"
                data-testid="input-nome-convite"
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700
mb-2">E-mail</label>
              <input
                type="email"
                value={emailConvite}
                onChange={(e) => setEmailConvite(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="Digite o e-mail"
                data-testid="input-email-convite"
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Telefone
(WhatsApp)</label>
              <input
                type="tel"
                value={telefoneConvite}
                onChange={(e) => setTelefoneConvite(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
```

```
                placeholder="(31) 99999-9999"
                data-testid="input-telefone-convite"
              />
            </div>

            <div className="space-y-3 pt-4">
              <button
                type="submit"
                disabled={isResgatandoIngresso}
                className="w-full bg-yellow-500 text-black py-4 rounded-xl hover:bg-yellow-600
transition font-semibold text-lg disabled:bg-gray-300"
                data-testid="button-confirmar-dados-convite"
              >
                {isResgatandoIngresso ? 'Gerando Convite...' : 'Confirmar e Gerar Convite'}
              </button>

              <button
                type="button"
                onClick={() => setEtapa('opcoes-resgate')}
                className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
                data-testid="button-voltar-opcoes"
              >
                Voltar
              </button>
            </div>
          </form>

          {/* Carrossel compacto de patrocinadores */}
          <BannerCarousel compact={true} />
        </div>
      )}

      {/* Formulário para Pix - TELA ÚNICA DE DADOS + ESCOLHA DE MÉTODO */}
      {etapa === 'formulario-pix' && (
        <div className="bg-black border-4 border-yellow-400 rounded-3xl p-8">
          <button
            onClick={voltarInicial}
            className="text-yellow-400 hover:text-yellow-300 mb-6 flex items-center gap-2
transition-colors"
            data-testid="button-voltar-top"
          >
            <span>← Voltar</span>
          </button>

          <h2 className="text-white text-3xl font-bold text-center mb-4"
data-testid="titulo-formulario-pix">
            Resgate de Ingresso Avulso
```

```
          </h2>

          <p className="text-gray-300 text-center mb-8">
            Preencha seus dados para continuar
          </p>

          <form onSubmit={handleFormSubmit} className="space-y-6">
            {/* Nome */}
            <div>
              <label className="text-white font-semibold mb-2 block">Nome Completo
*</label>
              <input
                type="text"
                value={nome}
                onChange={(e) => setNome(e.target.value)}
                required
                className="w-full h-14 bg-white text-black text-lg rounded-xl px-4
placeholder:text-gray-400"
                placeholder="Digite seu nome completo"
                data-testid="input-nome"
              />
            </div>

            {/* Email */}
            <div>
              <label className="text-white font-semibold mb-2 block">E-mail *</label>
              <input
                type="email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                required
                className="w-full h-14 bg-white text-black text-lg rounded-xl px-4
placeholder:text-gray-400"
                placeholder="seu@email.com"
                data-testid="input-email"
              />
            </div>

            {/* Telefone */}
            <div>
              <label className="text-white font-semibold mb-2 block">Telefone (WhatsApp)
*</label>
              <input
                type="tel"
                value={telefone}
                onChange={(e) => setTelefone(e.target.value)}
                required
```

```
              className="w-full h-14 bg-white text-black text-lg rounded-xl px-4
placeholder:text-gray-400"
              placeholder="(31) 99999-9999"
              data-testid="input-telefone"
            />
          </div>

          {/* Botão Confirmar */}
          <button
            type="submit"
            className="w-full h-14 bg-gray-300 hover:bg-gray-200 text-black text-lg
font-semibold rounded-xl transition-colors"
            data-testid="button-confirmar-pix"
          >
            Confirmar e ver QR Code Pix
          </button>
        </form>

        {/* Carrossel compacto de patrocinadores */}
        <div className="mt-8">
          <BannerCarousel compact={true} />
        </div>
      </div>
    )}

    {/* QR Code Pix */}
    {etapa === 'qr-code' && (
      <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
        <h2 className="text-2xl font-bold text-center mb-4" data-testid="titulo-qr-code">
          Pix Copia e Cola
        </h2>

        {/* Código de Identificação */}
        {pixTxid && (
          <div className="bg-yellow-50 border-2 border-yellow-400 rounded-lg p-4 mb-4">
            <p className="text-sm font-semibold text-yellow-800 mb-1">📋 Código de
Identificação:</p>
            <p className="text-2xl font-bold text-yellow-900 text-center">{pixTxid}</p>
            <p className="text-xs text-yellow-700 mt-2 text-center">
              Guarde este código para acompanhar seu pagamento
            </p>
          </div>
        )}

        <div className="bg-gray-50 p-6 rounded-xl mb-6">
          <div className="flex justify-center mb-4">
            <div className="bg-white p-4 rounded-xl">
              <QRCode value={pixPayload} size={200} data-testid="qr-code" />
```

```
            </div>
          </div>

          <p className="text-center text-gray-600 mb-4 leading-relaxed">
            Escaneie o QR Code acima<br />
            com o app do seu banco
          </p>

          <div className="border-t border-gray-200 pt-4">
            <p className="text-sm font-semibold text-gray-700 mb-2">Ou copie o código
Pix:</p>
            <div className="bg-white border border-gray-300 rounded-lg p-3 mb-3 break-all
text-xs font-mono text-gray-700">
              {pixPayload}
            </div>
            <button
              onClick={() => {
                navigator.clipboard.writeText(pixPayload);
                toast({
                  title: "Código Pix copiado!",
                  description: "Faça o pagamento no app do seu banco",
                });
              }}
              className="w-full bg-yellow-500 text-black py-3 rounded-xl hover:bg-yellow-600
transition font-semibold"
              data-testid="button-copiar-pix"
            >
              Copiar código Pix
            </button>
          </div>
        </div>

        <div className="bg-blue-50 border border-blue-200 rounded-lg p-4 mb-4">
          <p className="text-sm text-blue-800">
            <strong>💡 Como pagar:</strong>
          </p>
          <ol className="text-sm text-blue-700 mt-2 space-y-1 list-decimal list-inside">
            <li>Abra o app do seu banco</li>
            <li>Escolha "Pagar com Pix"</li>
            <li>Escaneie o QR Code ou cole o código</li>
            <li>Confirme o pagamento de R$ 1.000,00</li>
            <li>Aguarde a confirmação manual do pagamento pela equipe</li>
          </ol>
        </div>

        <div className="bg-green-50 border border-green-200 rounded-lg p-4 mb-4">
          <p className="text-sm text-green-800">
            <strong>✅ Próximos passos:</strong>
```

```
            </p>
            <p className="text-sm text-green-700 mt-2">
              Após realizar o pagamento PIX, nossa equipe confirmará seu ingresso
manualmente.
              Você receberá uma confirmação por WhatsApp no número {telefone}.
            </p>
          </div>

          <button
            onClick={voltarInicial}
            className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition mt-4"
            data-testid="button-voltar-inicial"
          >
            Voltar ao início
          </button>
        </div>
      )}

      {/* Formulário de Cartão (sem dados de cliente) */}
      {etapa === 'formulario-cartao' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-formulario-cartao">
            Seus dados
          </h2>

          <form onSubmit={(e) => {
            e.preventDefault();
            setEtapa('escolher-metodo-pagamento');
          }} className="space-y-4">
            {/* Nome */}
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Nome
Completo</label>
              <input
                type="text"
                value={nome}
                onChange={(e) => setNome(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="Digite seu nome"
                data-testid="input-nome-cartao"
              />
            </div>

            {/* Email */}
```

```
            <div>
              <label className="block text-sm font-medium text-gray-700
mb-2">E-mail</label>
              <input
                type="email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="seu@email.com"
                data-testid="input-email-cartao"
              />
            </div>

            {/* Telefone */}
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Telefone
(WhatsApp)</label>
              <input
                type="tel"
                value={telefone}
                onChange={(e) => setTelefone(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="(31) 99999-9999"
                data-testid="input-telefone-cartao"
              />
            </div>

            {/* Botões */}
            <div className="space-y-3 pt-4">
              <button
                type="submit"
                disabled={isCreatingPayment}
                className="w-full bg-yellow-500 text-black py-4 rounded-xl hover:bg-yellow-600
transition font-semibold text-lg disabled:bg-gray-300"
                data-testid="button-continuar-cartao"
              >
                {isCreatingPayment ? 'Preparando...' : 'Continuar para pagamento'}
              </button>

              <button
                type="button"
                onClick={voltarInicial}
                className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
```

```
              data-testid="button-voltar-cartao"
            >
              Voltar
            </button>
          </div>
        </form>
      </div>
    )}

    {/* Escolher Método de Pagamento (PIX ou Cartão) */}
    {etapa === 'escolher-metodo-pagamento' && (
      <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
        <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-escolher-metodo">
          Escolha a forma de pagamento
        </h2>

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

        {/* Cartão (Cielo) com seletor de quantidade */}
        <div className="border-2 border-cyan-400 rounded-xl p-4">
          <h3 className="font-semibold text-lg mb-3 text-center">Pagar com Cartão ou
PIX</h3>

          {/* Seletor de Quantidade */}
          <div className="mb-4">
            <label className="block text-sm font-medium text-gray-700 mb-2 text-center">
              Quantos ingressos você quer comprar?
            </label>
            <div className="flex gap-2 justify-center flex-wrap">
              {[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((qtd) => (
                <button
                  key={qtd}
                  type="button"
                  onClick={() => setQuantidadeIngressos(qtd)}
                  className={`px-4 py-2 rounded-lg font-semibold transition ${
                    quantidadeIngressos === qtd
                      ? 'bg-cyan-500 text-white'
                      : 'bg-gray-200 text-gray-700 hover:bg-gray-300'
                  }`}
                  data-testid={`button-quantidade-cielo-${qtd}`}
                >
                  {qtd}
                </button>
              ))}
            </div>

            {/* Total */}
```

```
              <div className="mt-3 text-center">
                <p className="text-sm text-gray-600">
                  {quantidadeIngressos} ingresso{quantidadeIngressos > 1 ? 's' : ''} × R$
1.000,00
                </p>
                <p className="text-xl font-bold text-cyan-600">
                  Total: R$ {(quantidadeIngressos * 1000)
                    .toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits:
2 })}
                </p>
              </div>
            </div>

            {/* Botão pagar com Cielo */}
            <button
              onClick={() => handleCieloClick(quantidadeIngressos)}
              disabled={isCreatingPayment}
              className="w-full bg-cyan-500 text-white py-4 rounded-xl hover:bg-cyan-600
transition font-semibold text-lg disabled:bg-gray-300"
              data-testid="button-pagar-cartao-cielo"
            >
              {isCreatingPayment ? 'Preparando...' : 'Pagar com Cartão ou PIX'}
            </button>

            <p className="text-sm text-gray-500 text-center mt-2">
              Abre checkout Cielo em nova janela
            </p>
          </div>

          {/* Botão Voltar */}
          <button
            onClick={voltarInicial}
            className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
            data-testid="button-voltar-metodo"
          >
            Voltar
          </button>
        </div>
      </div>
    )}

    {/* Verificando Pagamento */}
    {etapa === 'verificando-pagamento' && (
      <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
        <h2 className="text-2xl font-bold text-center mb-6" data-testid="titulo-verificando">
          Verificando pagamento...
        </h2>
```

```
          <div className="flex justify-center py-8">
            <div className="animate-spin rounded-full h-16 w-16 border-b-2
border-yellow-500"></div>
          </div>
          <p className="text-center text-gray-600">Por favor, aguarde enquanto verificamos
seu pagamento.</p>
        </div>
      )}

      {/* Formulário Rede - Quantidade + Parcelas + Cartão */}
      {etapa === 'formulario-rede' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-formulario-rede">
            Pagamento com Cartão - Rede
          </h2>

          <form onSubmit={handleRedeSubmit} className="space-y-4">
            {/* Quantidade */}
            <div>
              <label className="block text-sm font-medium mb-2">Quantidade de
Ingressos</label>
              <select
                value={redeQty}
                onChange={(e) => setRedeQty(Number(e.target.value))}
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="select-qty-rede"
              >
                {[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((n) => (
                  <option key={n} value={n}>{n} ingresso{n > 1 ? 's' : ''} - R$ {(n *
1000).toLocaleString('pt-BR')}</option>
                ))}
              </select>
            </div>

            {/* Parcelas */}
            <div>
              <label className="block text-sm font-medium mb-2">Parcelas (sem
juros)</label>
              <select
                value={redeInstallments}
                onChange={(e) => setRedeInstallments(Number(e.target.value))}
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="select-installments-rede"
              >
                {[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((n) => (
                  <option key={n} value={n}>
```

```
                {n}x de R$ {((redeQty * 1000) / n).toLocaleString('pt-BR', {
minimumFractionDigits: 2 })}
                </option>
              ))}
            </select>
          </div>

          {/* Número do Cartão */}
          <div>
            <label className="block text-sm font-medium mb-2">Número do Cartão</label>
            <input
              type="text"
              value={redeCardNumber}
              onChange={(e) => {
                const val = e.target.value.replace(/\D/g, '').slice(0, 16);
                const formatted = val.match(/.{1,4}/g)?.join(' ') || val;
                setRedeCardNumber(formatted);
              }}
              placeholder="1234 5678 9012 3456"
              className="w-full border border-gray-300 rounded-xl p-3"
              data-testid="input-card-number-rede"
              required
            />
          </div>

          {/* Nome no Cartão */}
          <div>
            <label className="block text-sm font-medium mb-2">Nome no Cartão</label>
            <input
              type="text"
              value={redeCardName}
              onChange={(e) => setRedeCardName(e.target.value.toUpperCase())}
              placeholder="NOME COMPLETO"
              className="w-full border border-gray-300 rounded-xl p-3"
              data-testid="input-card-name-rede"
              required
            />
          </div>

          {/* Validade e CVV */}
          <div className="grid grid-cols-3 gap-4">
            <div>
              <label className="block text-sm font-medium mb-2">Mês</label>
              <input
                type="text"
                value={redeCardMonth}
                onChange={(e) => setRedeCardMonth(e.target.value.replace(/\D/g, '').slice(0,
2))}
```

```
                placeholder="MM"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-month-rede"
                required
              />
            </div>
            <div>
              <label className="block text-sm font-medium mb-2">Ano</label>
              <input
                type="text"
                value={redeCardYear}
                onChange={(e) => setRedeCardYear(e.target.value.replace(/\D/g, '').slice(0, 4))}
                placeholder="AAAA"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-year-rede"
                required
              />
            </div>
            <div>
              <label className="block text-sm font-medium mb-2">CVV</label>
              <input
                type="text"
                value={redeCardCvv}
                onChange={(e) => setRedeCardCvv(e.target.value.replace(/\D/g, '').slice(0, 4))}
                placeholder="123"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-cvv-rede"
                required
              />
            </div>
          </div>

          {/* Total */}
          <div className="bg-gray-50 p-4 rounded-xl">
            <div className="flex justify-between items-center">
              <span className="font-semibold">Total:</span>
              <span className="text-2xl font-bold text-blue-600">
                R$ {(redeQty * 1000).toLocaleString('pt-BR')}
              </span>
            </div>
            <div className="text-sm text-gray-600 mt-1">
              {redeInstallments}x de R$ {((redeQty * 1000) /
redeInstallments).toLocaleString('pt-BR', { minimumFractionDigits: 2 })}
            </div>
          </div>

          {/* Botões */}
          <button
```

```
              type="submit"
              disabled={isCreatingPayment}
              className="w-full bg-blue-500 text-white py-4 rounded-xl hover:bg-blue-600
transition font-semibold disabled:bg-gray-300"
              data-testid="button-submit-rede"
            >
              {isCreatingPayment ? 'Processando...' : 'Confirmar Pagamento'}
            </button>

            <button
              type="button"
              onClick={() => setEtapa('escolher-metodo-pagamento')}
              className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
              data-testid="button-voltar-rede"
            >
              Voltar
            </button>
          </form>
        </div>
      )}

      {/* Formulário Cielo - Quantidade + Parcelas + Cartão */}
      {etapa === 'formulario-cielo' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-formulario-cielo">
            Pagamento com Cartão - Cielo
          </h2>

          <form onSubmit={handleCieloSubmit} className="space-y-4">
            {/* Quantidade */}
            <div>
              <label className="block text-sm font-medium mb-2">Quantidade de
Ingressos</label>
              <select
                value={cieloQty}
                onChange={(e) => setCieloQty(Number(e.target.value))}
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="select-qty-cielo"
              >
                {[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((n) => (
                  <option key={n} value={n}>{n} ingresso{n > 1 ? 's' : ''} - R$ {(n *
1000).toLocaleString('pt-BR')}</option>
                ))}
              </select>
            </div>
```

```
            {/* Parcelas */}
            <div>
              <label className="block text-sm font-medium mb-2">Parcelas (sem
juros)</label>
              <select
                value={cieloInstallments}
                onChange={(e) => setCieloInstallments(Number(e.target.value))}
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="select-installments-cielo"
              >
                {[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((n) => (
                  <option key={n} value={n}>
                    {n}x de R$ {((cieloQty * 1000) / n).toLocaleString('pt-BR', {
minimumFractionDigits: 2 })}
                  </option>
                ))}
              </select>
            </div>

            {/* Número do Cartão */}
            <div>
              <label className="block text-sm font-medium mb-2">Número do Cartão</label>
              <input
                type="text"
                value={cieloCardNumber}
                onChange={(e) => {
                  const val = e.target.value.replace(/\D/g, '').slice(0, 16);
                  const formatted = val.match(/.{1,4}/g)?.join(' ') || val;
                  setCieloCardNumber(formatted);
                }}
                placeholder="1234 5678 9012 3456"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-number-cielo"
                required
              />
            </div>

            {/* Nome no Cartão */}
            <div>
              <label className="block text-sm font-medium mb-2">Nome no Cartão</label>
              <input
                type="text"
                value={cieloCardName}
                onChange={(e) => setCieloCardName(e.target.value.toUpperCase())}
                placeholder="NOME COMPLETO"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-name-cielo"
                required
```

```
            />
          </div>

          {/* Validade e CVV */}
          <div className="grid grid-cols-3 gap-4">
            <div>
              <label className="block text-sm font-medium mb-2">Mês</label>
              <input
                type="text"
                value={cieloCardMonth}
                onChange={(e) => setCieloCardMonth(e.target.value.replace(/\D/g, '').slice(0,
2))}
                placeholder="MM"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-month-cielo"
                required
              />
            </div>
            <div>
              <label className="block text-sm font-medium mb-2">Ano</label>
              <input
                type="text"
                value={cieloCardYear}
                onChange={(e) => setCieloCardYear(e.target.value.replace(/\D/g, '').slice(0, 4))}
                placeholder="AAAA"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-year-cielo"
                required
              />
            </div>
            <div>
              <label className="block text-sm font-medium mb-2">CVV</label>
              <input
                type="text"
                value={cieloCardCvv}
                onChange={(e) => setCieloCardCvv(e.target.value.replace(/\D/g, '').slice(0, 4))}
                placeholder="123"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-cvv-cielo"
                required
              />
            </div>
          </div>

          {/* Total */}
          <div className="bg-gray-50 p-4 rounded-xl">
            <div className="flex justify-between items-center">
              <span className="font-semibold">Total:</span>
```

```
                <span className="text-2xl font-bold text-cyan-600">
                  R$ {(cieloQty * 1000).toLocaleString('pt-BR')}
                </span>
              </div>
              <div className="text-sm text-gray-600 mt-1">
                {cieloInstallments}x de R$ {((cieloQty * 1000) /
cieloInstallments).toLocaleString('pt-BR', { minimumFractionDigits: 2 })}
              </div>
            </div>

            {/* Botões */}
            <button
              type="submit"
              disabled={isCreatingPayment}
              className="w-full bg-cyan-500 text-white py-4 rounded-xl hover:bg-cyan-600
transition font-semibold disabled:bg-gray-300"
              data-testid="button-submit-cielo"
            >
              {isCreatingPayment ? 'Processando...' : 'Confirmar Pagamento'}
            </button>

            <button
              type="button"
              onClick={() => setEtapa('escolher-metodo-pagamento')}
              className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
              data-testid="button-voltar-cielo"
            >
              Voltar
            </button>
          </form>
        </div>
      )}

      {/* Cadastrar Usuário - Tela Única */}
      {etapa === 'cadastrar-usuario' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-cadastrar-usuario">
            Usuário não encontrado. Complete seu cadastro:
          </h2>

          <form onSubmit={async (e) => {
            e.preventDefault();

            // Cadastrar usuário no banco
            try {
              const response = await fetch('/api/usuarios/cadastrar', {
```

```
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({
              nome,
              telefone,
              nomeEmpresa: temEmpresa ? nomeEmpresa : null
            }),
          });

          const resultado = await response.json();

          if (resultado.success) {
            // Cadastro feito! Agora resgatar o ingresso
            toast({
              title: "Cadastro realizado!",
              description: "Agora vamos resgatar seu ingresso...",
            });

            // Chamar endpoint de resgate
            setTimeout(async () => {
              try {
                const resgateResponse = await fetch('/api/ingresso/resgatar', {
                  method: 'POST',
                  headers: { 'Content-Type': 'application/json' },
                  body: JSON.stringify({
                    telefone,
                    nomeEmpresa: temEmpresa ? nomeEmpresa : null
                  }),
                });

                const resgateResultado = await resgateResponse.json();

                if (resgateResultado.success) {
                  // Salvar no localStorage
                  localStorage.setItem('ingressoResgatadoNumero',
resgateResultado.ingresso.numero);

                  // Redirecionar para visualização
                  window.location.href =
`/ingresso/visualizar/${resgateResultado.ingresso.numero}`;
                } else {
                  toast({
                    title: "Erro ao resgatar",
                    description: resgateResultado.error || "Erro ao resgatar ingresso",
                    variant: "destructive",
                  });
                }
              } catch (error) {
```

```
                console.error('Erro ao resgatar:', error);
                toast({
                  title: "Erro ao resgatar",
                  description: "Tente novamente.",
                  variant: "destructive",
                });
              }
            }, 1000);
          } else {
            toast({
              title: "Erro ao cadastrar",
              description: resultado.mensagem || "Ocorreu um erro ao cadastrar usuário",
              variant: "destructive",
            });
          }
        } catch (error) {
          console.error('Erro ao cadastrar usuário:', error);
          toast({
            title: "Erro ao cadastrar",
            description: "Por favor, tente novamente.",
            variant: "destructive",
          });
        }
      }} className="space-y-4">
        {/* Nome */}
        <div>
          <label className="block text-sm font-medium text-gray-700 mb-2">Nome</label>
          <input
            type="text"
            value={nome}
            disabled
            className="w-full px-4 py-3 border border-gray-300 rounded-xl bg-gray-100
text-gray-700"
            data-testid="input-nome-cadastro"
          />
        </div>

        {/* Telefone */}
        <div>
          <label className="block text-sm font-medium text-gray-700
mb-2">Telefone</label>
          <input
            type="tel"
            value={telefone}
            disabled
            className="w-full px-4 py-3 border border-gray-300 rounded-xl bg-gray-100
text-gray-700"
            data-testid="input-telefone-cadastro"
```

```
            />
          </div>

          {/* Tem Empresa? */}
          <div>
            <label className="block text-sm font-medium text-gray-700
mb-3">Empresa?</label>
            <div className="flex gap-3 mb-4">
              <button
                type="button"
                onClick={() => setTemEmpresa(true)}
                className={`flex-1 py-3 rounded-xl font-semibold transition ${
                  temEmpresa === true
                    ? 'bg-yellow-500 text-black'
                    : 'bg-gray-200 text-gray-700 hover:bg-gray-300'
                }`}
                data-testid="button-empresa-sim"
              >
                Sim
              </button>
              <button
                type="button"
                onClick={() => {
                  setTemEmpresa(false);
                  setNomeEmpresa('');
                }}
                className={`flex-1 py-3 rounded-xl font-semibold transition ${
                  temEmpresa === false
                    ? 'bg-yellow-500 text-black'
                    : 'bg-gray-200 text-gray-700 hover:bg-gray-300'
                }`}
                data-testid="button-empresa-nao"
              >
                Não
              </button>
            </div>
          </div>

          {/* Campo Empresa (se Sim) */}
          {temEmpresa === true && (
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Nome da
Empresa</label>
              <input
                type="text"
                value={nomeEmpresa}
                onChange={(e) => setNomeEmpresa(e.target.value)}
                required
```

```
                  className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                  placeholder="Digite o nome da empresa"
                  data-testid="input-nome-empresa"
                />
              </div>
            )}

            {/* Botão Continuar (aparece após escolher Sim ou Não) */}
            {temEmpresa !== null && (
              <button
                type="submit"
                className="w-full bg-green-500 text-white py-4 rounded-xl hover:bg-green-600
transition font-semibold text-lg"
                data-testid="button-continuar-cadastro"
              >
                Continuar
              </button>
            )}
          </form>
        </div>
      )}

      {/* Modal de Pagamento Stripe */}
      {etapa === 'pagamento-cartao' && clientSecret && (
        <Elements stripe={stripePromise} options={{ clientSecret }}>
          <StripePaymentModal
            clientSecret={clientSecret}
            onSuccess={async () => {
              // Verificar pagamento e ir para associar empresa
              await verificarPagamento();
            }}
            onCancel={() => {
              setEtapa('card-inicial');
              setClientSecret("");
            }}
            nomeComprador={nome}
            telefoneComprador={telefone}
            quantidade={quantidadeIngressos}
            valorTotal={100000 * quantidadeIngressos}
          />
        </Elements>
      )}

      {/* Tela de Confirmação Pagamento Rede */}
      {etapa === 'pagamento-rede' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <div className="text-center mb-6">
```

```
            <div className="bg-blue-100 rounded-full w-20 h-20 flex items-center
justify-center mx-auto mb-4">
              <span className="text-4xl">💳</span>
            </div>
            <h2 className="text-2xl font-bold mb-2" data-testid="titulo-pagamento-rede">
              Pagamento com Rede
            </h2>
            <p className="text-gray-600">
              Seu pagamento está sendo processado
            </p>
          </div>

          <div className="bg-gray-50 rounded-lg p-4 mb-6">
            <p className="text-sm text-gray-600">Comprador: {nome}</p>
            <p className="text-sm text-gray-600">Telefone: {telefone}</p>
            <p className="text-sm text-gray-600">Quantidade: 1 ingresso</p>
            <p className="font-bold text-lg mt-2">Valor: R$ 1.000,00</p>
          </div>

          <div className="flex justify-center py-8">
            <div className="animate-spin rounded-full h-16 w-16 border-b-2
border-blue-500"></div>
          </div>

          <p className="text-center text-sm text-gray-500 mb-4">
            Aguarde enquanto processamos seu pagamento...
          </p>

          <button
            onClick={() => {
              setEtapa('card-inicial');
              toast({
                title: "Pagamento cancelado",
                description: "Você pode tentar novamente quando quiser.",
              });
            }}
            className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
            data-testid="button-cancelar-rede"
          >
            Cancelar
          </button>
        </div>
      )}

      {/* Mensagem de Segurança LGPD */}
      <div className="mt-6 bg-gray-50 border border-gray-200 rounded-xl p-4">
        <p className="text-gray-700 text-xs text-center leading-relaxed">
```

```
          🔒 <span className="font-semibold">Ambiente 100% Seguro.</span> Seus dados pessoais e de pagamento estão protegidos. Seguimos rigorosamente todos os protocolos de segurança e as diretrizes da Lei Geral de Proteção de Dados (LGPD). Nenhuma informação do seu cartão é armazenada em nossos servidores.
          </p>
        </div>
      </div>
    </div>
  );
}
mentRequestButtonElement } from "@stripe/react-stripe-js";
import { useToast } from "@/hooks/use-toast";
import PatrocinadoresCarousel from "@/components/PatrocinadoresCarousel";
import logoClube from "@assets/LOGO_CLUBE-05_1752081350082.png";

const stripePromise = loadStripe(import.meta.env.VITE_STRIPE_PUBLIC_KEY ||
"pk_test_51RdaS1Qlsea8vAKZC1WmSHcCGXNGGTxJuLZ3iq90MUpeCxq5CUhj5C2Qwm
HWO008hWIMSaZ0yh75EzrSUpXyvTs6002cYD8L9l");

// Componente de Carrossel usando o componente unificado
function BannerCarousel({ compact = false, showTitle = false }: { compact?: boolean;
showTitle?: boolean }) {
  return <PatrocinadoresCarousel showTitle={showTitle} className={compact ? "mt-6 mb-4"
: "mb-6"} />;
}

// Componente de pagamento Stripe
function StripePaymentModal({ clientSecret, onSuccess, onCancel, nomeComprador,
telefoneComprador, quantidade, valorTotal }: {
  clientSecret: string;
  onSuccess: () => void;
  onCancel: () => void;
  nomeComprador: string;
  telefoneComprador: string;
  quantidade: number;
  valorTotal: number;
}) {
  const stripe = useStripe();
  const elements = useElements();
  const [isProcessing, setIsProcessing] = useState(false);
  const [paymentRequest, setPaymentRequest] = useState<any>(null);

  // Configurar Apple Pay / Google Pay
  useEffect(() => {
    if (!stripe) return;

    const pr = stripe.paymentRequest({
      country: 'BR',
```

```
      currency: 'brl',
      total: {
        label: `Ingresso${quantidade > 1 ? 's' : ''} Premium - Instituto O Grito`,
        amount: valorTotal, // Valor total em centavos
      },
      requestPayerName: true,
      requestPayerEmail: true,
    });

    pr.canMakePayment().then(result => {
      if (result) {
        setPaymentRequest(pr);
      }
    });

    pr.on('paymentmethod', async (ev) => {
      setIsProcessing(true);
      try {
        const { error } = await stripe.confirmCardPayment(clientSecret, {
          payment_method: ev.paymentMethod.id
        });

        if (error) {
          ev.complete('fail');
          alert('Erro no pagamento: ' + error.message);
        } else {
          ev.complete('success');
          onSuccess();
        }
      } catch (error) {
        ev.complete('fail');
        alert('Erro no pagamento');
      }
      setIsProcessing(false);
    });
  }, [stripe, clientSecret, onSuccess]);

  const handleSubmit = async (event: React.FormEvent) => {
    event.preventDefault();

    if (!stripe || !elements) {
      return;
    }

    setIsProcessing(true);

    try {
      const { error } = await stripe.confirmPayment({
```

```
        elements,
        confirmParams: {
          return_url: window.location.origin,
        },
        redirect: 'if_required'
      });

      if (error) {
        alert('Erro no pagamento: ' + error.message);
      } else {
        onSuccess();
      }
    } catch (error) {
      alert('Erro no pagamento');
    }

    setIsProcessing(false);
  };

  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50
p-4">
      <div className="bg-white rounded-xl p-6 max-w-md w-full max-h-[90vh]
overflow-y-auto">
        <div className="flex justify-between items-center mb-4">
          <h2 className="text-xl font-bold">Pagamento com Cartão</h2>
          <button
            onClick={onCancel}
            className="text-gray-500 hover:text-gray-700 text-2xl"
            data-testid="button-fechar-modal"
          >
            ×
          </button>
        </div>

        <div className="mb-4 p-3 bg-gray-50 rounded-lg">
          <p className="text-sm text-gray-600">Ingresso para: {nomeComprador}</p>
          <p className="text-sm text-gray-600">Telefone: {telefoneComprador}</p>
          <p className="text-sm text-gray-600">Quantidade: {quantidade}
ingresso{quantidade > 1 ? 's' : ''}</p>
          <p className="font-bold">Valor Total: R$ {(valorTotal / 100).toLocaleString('pt-BR', {
minimumFractionDigits: 2, maximumFractionDigits: 2 })}</p>
        </div>

        {/* Apple Pay / Google Pay */}
        {paymentRequest && (
          <div className="mb-4">
            <PaymentRequestButtonElement
```

```
            options={{ paymentRequest }}
            className="stripe-payment-request-button"
          />
          <div className="text-center my-4 text-gray-500">ou</div>
        </div>
      )}

      {/* Formulário de cartão */}
      <form onSubmit={handleSubmit}>
        <div className="mb-4">
          <PaymentElement />
        </div>

        <button
          type="submit"
          disabled={isProcessing || !stripe || !elements}
          className="w-full bg-yellow-500 text-black py-3 rounded-xl hover:bg-yellow-600
transition font-semibold disabled:bg-gray-300"
          data-testid="button-confirmar-pagamento"
        >
          {isProcessing ? 'Processando...' : `Confirmar Pagamento R$ ${(valorTotal /
100).toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits: 2 })}`}
        </button>
      </form>
    </div>
  </div>
 );
}

export default function PagamentoIngresso() {
 const { toast } = useToast();
 const [nome, setNome] = useState("");
 const [telefone, setTelefone] = useState("");
 const [email, setEmail] = useState("");
 const [etapa, setEtapa] = useState<'card-inicial' | 'opcoes-pagamento' | 'formulario-pix' |
'qual-empresa' | 'opcoes-resgate' | 'form-dados-convite' | 'qr-code' | 'formulario-cartao' |
'escolher-metodo-pagamento' | 'pagamento-cartao' | 'pagamento-rede' |
'verificando-pagamento' | 'associar-empresa' | 'cadastrar-usuario' | 'confirmar-vinculo' |
'formulario-rede' | 'formulario-cielo'>('card-inicial');
 const [clientSecret, setClientSecret] = useState("");
 const [isCreatingPayment, setIsCreatingPayment] = useState(false);
 const [showPixFallback, setShowPixFallback] = useState(false);
 const [quantidadeIngressos, setQuantidadeIngressos] = useState(1);

 // Estados para formulário Rede
 const [redeQty, setRedeQty] = useState(1);
 const [redeInstallments, setRedeInstallments] = useState(1);
 const [redeCardNumber, setRedeCardNumber] = useState("");
```

```
  const [redeCardName, setRedeCardName] = useState("");
  const [redeCardMonth, setRedeCardMonth] = useState("");
  const [redeCardYear, setRedeCardYear] = useState("");
  const [redeCardCvv, setRedeCardCvv] = useState("");

  // Estados para formulário Cielo
  const [cieloQty, setCieloQty] = useState(1);
  const [cieloInstallments, setCieloInstallments] = useState(1);
  const [cieloCardNumber, setCieloCardNumber] = useState("");
  const [cieloCardName, setCieloCardName] = useState("");
  const [cieloCardMonth, setCieloCardMonth] = useState("");
  const [cieloCardYear, setCieloCardYear] = useState("");
  const [cieloCardCvv, setCieloCardCvv] = useState("");

  // Timer e verificação de pagamento
  const [timerSegundos, setTimerSegundos] = useState(60);
  const [timerIniciado, setTimerIniciado] = useState(false);
  const [paymentIntentId, setPaymentIntentId] = useState("");

  // Estados para sistema de cotas empresariais
  const [nomeEmpresa, setNomeEmpresa] = useState("");
  const [emailEmpresa, setEmailEmpresa] = useState("");
  const [cotaEmpresa, setCotaEmpresa] = useState<any>(null);
  const [isValidatingEmpresa, setIsValidatingEmpresa] = useState(false);
  const [isResgatandoIngresso, setIsResgatandoIngresso] = useState(false);
  const [temEmpresa, setTemEmpresa] = useState<boolean | null>(null);

  // Dados do convite (para "Colocar Nome no Convite")
  const [nomeConvite, setNomeConvite] = useState("");
  const [emailConvite, setEmailConvite] = useState("");
  const [telefoneConvite, setTelefoneConvite] = useState("");

  // Estado para TXID PIX (identificador único do pagamento)
  const [pixTxid, setPixTxid] = useState("");

  // Payload Pix oficial (CNPJ: 28.790.664/0001-10 | Valor: R$ 1.000,00)
  const pixPayload =

"00020126360014BR.GOV.BCB.PIX011428790664000110520400005303986540610000.005
802BR5917INSTITUTO O GRITO6008RIBEIRAO62070503***63045B06";

  // Timer de 1 minuto para verificar pagamento PIX
  useEffect(() => {
    let interval: NodeJS.Timeout;

    if (timerIniciado && timerSegundos > 0) {
      interval = setInterval(() => {
        setTimerSegundos((prev) => prev - 1);
```

```
    }, 1000);
  }

  return () => clearInterval(interval);
}, [timerIniciado, timerSegundos]);

const verificarPagamento = async () => {
  setEtapa('verificando-pagamento');

  try {
    const response = await fetch('/api/verificar-pagamento-stripe', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        nome,
        telefone,
        paymentIntentId
      }),
    });

    const resultado = await response.json();

    if (resultado.pago) {
      // Pagamento confirmado, ir para tela de cadastro
      setEtapa('cadastrar-usuario');
    } else {
      toast({
        title: "Pagamento não detectado",
        description: "Por favor, tente novamente ou entre em contato.",
        variant: "destructive",
      });
      setEtapa('qr-code');
      setTimerSegundos(60);
      setTimerIniciado(false);
    }
  } catch (error) {
    console.error('Erro ao verificar pagamento:', error);
    toast({
      title: "Erro ao verificar pagamento",
      description: "Por favor, tente novamente.",
      variant: "destructive",
    });
    setEtapa('qr-code');
    setTimerSegundos(60);
    setTimerIniciado(false);
  }
};
```

```
const handlePixFallback = () => {
  setShowPixFallback(true);
  // Copiar automaticamente quando mostra fallback
  navigator.clipboard.writeText(pixPayload);
};

const mostrarOpcoesPagamento = () => {
  setEtapa('opcoes-pagamento');
};

const handlePixClick = async () => {
  // Verificar se temos os dados necessários
  if (!nome || !telefone) {
    setEtapa('formulario-pix');
    return;
  }

  setIsCreatingPayment(true);

  try {
    // Criar registro PIX pendente
    const response = await fetch('/api/ingressos/pix/iniciar', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        nome,
        telefone,
        email: email || undefined,
        quantidade: 1 // PIX sempre 1 ingresso
      }),
    });

    if (!response.ok) {
      throw new Error('Erro ao criar registro PIX');
    }

    const data = await response.json();

    // Salvar TXID e ir para tela de QR code
    setPixTxid(data.txid);
    setQuantidadeIngressos(1);
    setEtapa('qr-code');

    console.log(`✅ Registro PIX criado com TXID: ${data.txid}`);

  } catch (error) {
    console.error('❌ Erro ao criar registro PIX:', error);
    toast({
```

```
      title: "Erro ao gerar PIX",
      description: "Tente novamente ou escolha outro método de pagamento.",
      variant: "destructive",
    });
  } finally {
    setIsCreatingPayment(false);
  }
};

const handleValidarEmpresa = async (e: React.FormEvent) => {
  e.preventDefault();

  if (!nomeEmpresa.trim() || !emailEmpresa.trim()) {
    alert('Por favor, preencha nome e e-mail da empresa');
    return;
  }

  setIsValidatingEmpresa(true);

  try {
    const response = await fetch('/api/cotas/validar-empresa', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        nomeEmpresa: nomeEmpresa.trim(),
        email: emailEmpresa.trim()
      }),
    });

    const resultado = await response.json();

    if (resultado.valida && resultado.cota) {
      setCotaEmpresa(resultado.cota);
      setEtapa('opcoes-resgate');
    } else {
      alert(resultado.mensagem || 'Dados incorretos ou empresa sem convites disponíveis');
    }
  } catch (error) {
    console.error('Erro ao validar empresa:', error);
    alert('Erro ao buscar empresa. Tente novamente.');
  } finally {
    setIsValidatingEmpresa(false);
  }
};

const handleRetirarAgora = async () => {
  if (!cotaEmpresa) return;
```

```
    setIsResgatandoIngresso(true);

    try {
      const response = await fetch('/api/cotas/resgatar-ingresso', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          idCota: cotaEmpresa.id
        }),
      });

      const resultado = await response.json();

      if (resultado.success && resultado.ingresso) {
        // Salvar dados para navegação de volta
        localStorage.setItem('ultimoResgateCotaId', cotaEmpresa.id.toString());
        localStorage.setItem('ingressoResgatadoNumero', resultado.ingresso.numero);

        console.log('🎯 [RETIRAR-AGORA] Redirecionando para:',
`/ingresso/visualizar/${resultado.ingresso.numero}`);

        // Redirecionar para visualização do ingresso usando o número
        window.location.href = `/ingresso/visualizar/${resultado.ingresso.numero}`;
      } else {
        alert(resultado.error || 'Erro ao resgatar ingresso');
      }
    } catch (error) {
      console.error('Erro ao resgatar ingresso:', error);
      alert('Erro ao resgatar ingresso. Tente novamente.');
    } finally {
      setIsResgatandoIngresso(false);
    }
  };

  const handleColocarNome = () => {
    setEtapa('form-dados-convite');
  };

  const handleSubmitDadosConvite = async (e: React.FormEvent) => {
    e.preventDefault();

    if (!cotaEmpresa) return;

    setIsResgatandoIngresso(true);

    try {
      const response = await fetch('/api/cotas/resgatar-ingresso', {
        method: 'POST',
```

```
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          idCota: cotaEmpresa.id,
          nomeComprador: nomeConvite.trim(),
          emailComprador: emailConvite.trim(),
          telefoneComprador: telefoneConvite.trim()
        }),
      });

      const resultado = await response.json();

      if (resultado.success && resultado.ingresso) {
        // Salvar dados para navegação de volta
        localStorage.setItem('ultimoResgateCotaId', cotaEmpresa.id.toString());
        localStorage.setItem('ingressoResgatadoNumero', resultado.ingresso.numero);

        console.log('🎯 [PAGAMENTO-INGRESSO] Redirecionando para:',
`/ingresso/visualizar/${resultado.ingresso.numero}`);

        // Redirecionar para visualização do ingresso usando o número
        window.location.href = `/ingresso/visualizar/${resultado.ingresso.numero}`;
      } else {
        alert(resultado.error || 'Erro ao resgatar ingresso');
      }
    } catch (error) {
      console.error('Erro ao resgatar ingresso:', error);
      alert('Erro ao resgatar ingresso. Tente novamente.');
    } finally {
      setIsResgatandoIngresso(false);
    }
  };

  const handleFormSubmit = async (e: React.FormEvent) => {
    e.preventDefault();

    try {
      // Enviar dados para o backend
      const response = await fetch('/api/patrocinador-2026', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ nome, email, telefone }),
      });

      if (!response.ok) {
        throw new Error('Erro ao salvar dados');
      }
```

```
    const data = await response.json();
    console.log('✅ Patrocinador 2026 cadastrado:', data);

    // Prosseguir para o QR code
    setEtapa('qr-code');
  } catch (error) {
    console.error('❌ Erro ao cadastrar patrocinador:', error);
    // Ainda assim mostrar o QR code (não bloquear pagamento)
    setEtapa('qr-code');
  }
};

const handleCartaoClick = async () => {
  // Primeiro, coletar dados do comprador
  if (!nome || !telefone) {
    // Se não temos dados, ir para formulário de cartão primeiro
    setEtapa('formulario-cartao');
    return;
  }

  setIsCreatingPayment(true);

  try {
    // Salvar dados do patrocinador 2026
    await fetch('/api/patrocinador-2026', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ nome, email, telefone }),
    });

    // Criar PaymentIntent para o ingresso (valor unitário R$ 1.000,00 * quantidade)
    const valorUnitario = 100000; // R$ 1.000,00 em centavos
    const valorTotal = valorUnitario * quantidadeIngressos;

    const response = await fetch('/api/ingresso/create-payment', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        nome,
        telefone,
        amount: valorTotal,
        quantidade: quantidadeIngressos
      }),
    });

    if (!response.ok) {
      throw new Error('Erro ao criar pagamento');
```

```
    }

    const { clientSecret } = await response.json();
    setClientSecret(clientSecret);
    setEtapa('pagamento-cartao');

  } catch (error) {
    console.error('❌ Erro ao criar pagamento:', error);
    toast({
      title: "Erro ao iniciar pagamento",
      description: "Por favor, tente novamente.",
      variant: "destructive",
    });
  } finally {
    setIsCreatingPayment(false);
  }
};

const handleRedeClick = async () => {
  // Primeiro, coletar dados do comprador
  if (!nome || !telefone) {
    // Se não temos dados, ir para formulário de cartão primeiro
    setEtapa('formulario-cartao');
    return;
  }

  // Ir para formulário de Rede (novo fluxo)
  setEtapa('formulario-rede');
};

const handleRedeSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setIsCreatingPayment(true);

  try {
    // Salvar dados do patrocinador 2026
    await fetch('/api/patrocinador-2026', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ nome, email, telefone }),
    });

    // Processar pagamento com Rede
    const response = await fetch('/api/pagamentos/rede/checkout', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        qty: redeQty,
```

```
      installments: redeInstallments,
      card: {
        cardNumber: redeCardNumber.replace(/\s/g, ''),
        cardholderName: redeCardName,
        expirationMonth: redeCardMonth,
        expirationYear: redeCardYear,
        securityCode: redeCardCvv
      },
      nome,
      telefone
    }),
  });

  const result = await response.json();

  if (result.ok) {
    // Verificar se há 3DS (redirectUrl para autenticação)
    if (result.data?.threeDSecure?.url) {
      toast({
        title: "Autenticação necessária",
        description: "Redirecionando para autenticação 3DS...",
      });

      // Redirecionar para 3DS
      setTimeout(() => {
        window.location.href = result.data.threeDSecure.url;
      }, 500);
    } else {
      // Pagamento direto aprovado
      toast({
        title: "Pagamento processado!",
        description: `Pedido ${result.orderId} criado com sucesso.`,
      });

      setTimeout(() => {
        window.location.href = `/ingresso/sucesso?orderId=${result.orderId}`;
      }, 500);
    }
  } else {
    throw new Error(result.error || 'Erro ao processar pagamento');
  }
} catch (error: any) {
  console.error('❌ Erro ao processar Rede:', error);
  toast({
    title: "Erro ao processar pagamento",
    description: error.message || "Tente novamente.",
    variant: "destructive",
  });
```

```
  } finally {
    setIsCreatingPayment(false);
  }
};

const handleCieloClick = async () => {
  // Primeiro, coletar dados do comprador
  if (!nome || !telefone || !email) {
    // Se não temos dados, ir para formulário de cartão primeiro
    setEtapa('formulario-cartao');
    return;
  }

  setIsCreatingPayment(true);

  try {
    // Criar pedido no backend antes de abrir o Checkout Cielo
    const response = await fetch('/api/pagamentos/cielo-checkout/criar', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        qty: quantidadeIngressos,
        nome,
        email,
        telefone
      })
    });

    const data = await response.json();

    if (data.ok && data.checkoutUrl) {
      console.log('🔵 [CIELO CHECKOUT] Abrindo checkout:', data.checkoutUrl);

      // Resetar estado de carregamento ANTES de redirecionar
      setIsCreatingPayment(false);

      // Redirecionar para Checkout Cielo na mesma página
      window.location.href = data.checkoutUrl;
    } else {
      setIsCreatingPayment(false);
      toast({
        title: "Erro ao processar pagamento",
        description: data.error || "Tente novamente.",
        variant: "destructive",
      });
    }
  } catch (error: any) {
    setIsCreatingPayment(false);
```

```
    console.error('❌ [CIELO CHECKOUT] Erro:', error);
    toast({
      title: "Erro ao processar pagamento",
      description: error.message || "Tente novamente.",
      variant: "destructive",
    });
  }
};

const handleCieloSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setIsCreatingPayment(true);

  try {
    // Salvar dados do patrocinador 2026
    await fetch('/api/patrocinador-2026', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ nome, email, telefone }),
    });

    // Processar pagamento com Cielo
    const response = await fetch('/api/pagamentos/cielo/checkout', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        qty: cieloQty,
        installments: cieloInstallments,
        card: {
          number: cieloCardNumber.replace(/\s/g, ''),
          holder: cieloCardName,
          expirationMonth: cieloCardMonth,
          expirationYear: cieloCardYear,
          securityCode: cieloCardCvv
        },
        nome,
        telefone
      }),
    });

    const result = await response.json();

    if (result.ok) {
      // Verificar se há 3DS (returnUrl para autenticação)
      if (result.returnUrl) {
        toast({
          title: "Autenticação necessária",
          description: "Redirecionando para autenticação 3DS...",
```

```
        });

        // Redirecionar para 3DS
        setTimeout(() => {
          window.location.href = result.returnUrl;
        }, 500);
      } else {
        // Pagamento direto aprovado
        toast({
          title: "Pagamento processado!",
          description: `Pedido ${result.orderId} criado com sucesso.`,
        });

        setTimeout(() => {
          window.location.href = `/ingresso/sucesso?orderId=${result.orderId}`;
        }, 500);
      }
    } else {
      throw new Error(result.message || 'Erro ao processar pagamento');
    }
  } catch (error: any) {
    console.error('❌ Erro ao processar Cielo:', error);
    toast({
      title: "Erro ao processar pagamento",
      description: error.message || "Tente novamente.",
      variant: "destructive",
    });
  } finally {
    setIsCreatingPayment(false);
  }
};

const voltarInicial = () => {
  setEtapa('card-inicial');
  setNome("");
  setTelefone("");
  setEmail("");
  setClientSecret(""); // Limpar clientSecret do Stripe
  setIsCreatingPayment(false); // Resetar estado de carregamento
};

const voltarOpcoes = () => {
  setEtapa('opcoes-pagamento');
};

return (
  <div className="min-h-screen bg-white" style={{ fontFamily: 'Inter, system-ui, sans-serif'
}}>
```

```
    <div className="container mx-auto max-w-md px-4 py-8">

      {/* Header com Logo */}
      <div className="text-center mb-6">
        <img
          src={logoClube}
          alt="Clube do Grito"
          className="h-32 mx-auto mb-4"
          data-testid="logo-clube"
        />

        {/* Carrossel sem título "Patrocinadores" */}
        {etapa === 'card-inicial' && <BannerCarousel showTitle={false} />}
      </div>

      {/* Card Unificado - Apoie o Grito + Benefícios do Ingresso */}
      {etapa === 'card-inicial' && (
        <div className="bg-black border-t-4 border-[#FFCC00] rounded-xl p-8 text-white">
          {/* Seção "Apoie todo mês" */}
          <div className="mb-8">
            <h2 className="text-xl font-semibold text-white text-center mb-4"
data-testid="titulo-apoie-todo-mes">
              Seu Grito, nossa VOZ!
            </h2>
            <p className="text-white text-justify leading-relaxed text-sm sm:text-base mb-6"
data-testid="texto-apoie-todo-mes">
              Ao realizar a compra desse convite, você contribui mensalmente com R$100,00
reais durante 10 meses para manter de pé nossas ações de acesso a cultura, esporte,
inclusão produtiva, socioemocional, educação e muito mais.
            </p>
          </div>

          {/* Linha divisória sutil */}
          <div className="border-t border-[#FFCC00] opacity-40 mb-6"></div>

          {/* Seção sobre o evento */}
          <div className="mt-4 mb-6">
            <h3 className="text-lg font-semibold text-white mb-2"
data-testid="titulo-sobre-encontro">
              Sobre o Encontro do Clube do Grito
            </h3>
            <p className="text-white text-justify leading-relaxed text-sm sm:text-base mb-4"
data-testid="texto-sobre-encontro">
              Uma noite para celebrar <span className="font-bold">quem apoia a
transformação</span>. Um encontro para se aproximar do nosso trabalho, o impacto
gerado, conectar parceiros e <span className="font-bold">arrecadar recursos</span> para
manter as oficinas o ano inteiro em 2026.
            </p>
```

```
          </div>

          {/* Botões de pagamento - movidos para cima */}
          <div className="space-y-4 mb-8">
            {/* Botão Patrocinadores */}
            <button
              onClick={() => setEtapa('qual-empresa')}
              className="w-full bg-black text-white py-4 rounded-xl hover:bg-gray-800
transition font-semibold text-lg border-2 border-white"
              data-testid="button-patrocinadores"
            >
              Patrocinadores
            </button>

            {/* Botão Ingresso Avulso */}
            <button
              onClick={() => {
                setQuantidadeIngressos(2);
                setEtapa('formulario-cartao'); // Ir direto para formulário (sem criar Stripe)
              }}
              disabled={isCreatingPayment}
              className="w-full bg-yellow-500 text-black py-4 rounded-xl hover:bg-yellow-600
transition font-semibold text-lg disabled:bg-gray-300"
              data-testid="button-pagar-cartao"
            >
              {isCreatingPayment ? 'Preparando pagamento...' : 'Comprar Ingresso Avulso'}
            </button>

          </div>

          {/* Seção "O que está incluso na experiência:" */}
          <div className="mt-8">
            <h3 className="text-lg font-semibold text-white mb-3"
data-testid="titulo-beneficios">
              O que está incluso na experiência:
            </h3>
            <ul className="space-y-3 text-white text-sm sm:text-base">
              <li className="flex items-start">
                <span className="text-[#FFCC00] mr-2 flex-shrink-0">✓</span>
                <span>Jantar exclusivo com menu especial</span>
              </li>
              <li className="flex items-start">
                <span className="text-[#FFCC00] mr-2 flex-shrink-0">✓</span>
                <span>Networking com convidados especiais</span>
              </li>
              <li className="flex items-start">
                <span className="text-[#FFCC00] mr-2 flex-shrink-0">✓</span>
                <span>Experiência musical e muita alegria</span>
```

```
            </li>
          </ul>
        </div>
      </div>
    )}

    {/* Tela: Qual empresa? */}
    {etapa === 'qual-empresa' && (
      <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
        <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-qual-empresa">
          Qual empresa?
        </h2>

        <form onSubmit={handleValidarEmpresa} className="space-y-4">
          <div>
            <label className="block text-sm font-medium text-gray-700 mb-2">Nome da
Empresa</label>
            <input
              type="text"
              value={nomeEmpresa}
              onChange={(e) => setNomeEmpresa(e.target.value)}
              required
              className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
              placeholder="Digite o nome da empresa"
              data-testid="input-nome-empresa"
            />
          </div>

          <div>
            <label className="block text-sm font-medium text-gray-700 mb-2">Insira e-mail
cadastrado</label>
            <input
              type="email"
              value={emailEmpresa}
              onChange={(e) => setEmailEmpresa(e.target.value)}
              required
              className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
              placeholder="Digite o e-mail cadastrado"
              data-testid="input-email-empresa"
            />
          </div>

          <div className="space-y-3 pt-4">
            <button
              type="submit"
```

```
                disabled={isValidatingEmpresa}
                className="w-full bg-yellow-500 text-black py-4 rounded-xl hover:bg-yellow-600
transition font-semibold text-lg disabled:bg-gray-300"
                data-testid="button-buscar-empresa"
              >
                {isValidatingEmpresa ? 'Buscando...' : 'Buscar Empresa'}
              </button>

              <button
                type="button"
                onClick={voltarInicial}
                className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
                data-testid="button-voltar"
              >
                Voltar
              </button>
            </div>
          </form>

          {/* Carrossel compacto de patrocinadores */}
          <BannerCarousel compact={true} />
        </div>
      )}

      {/* Tela: Opções de Resgate */}
      {etapa === 'opcoes-resgate' && cotaEmpresa && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-4"
data-testid="titulo-opcoes-resgate">
            {cotaEmpresa.nomeEmpresa}
          </h2>

          <div className="bg-gray-50 rounded-lg p-4 mb-6">
            <div className="flex justify-between items-center mb-2">
              <span className="text-gray-700 font-medium">Total de Convites:</span>
              <span className="text-2xl font-bold
text-gray-900">{cotaEmpresa.quantidadeTotal}</span>
            </div>
            <div className="flex justify-between items-center mb-2">
              <span className="text-gray-700 font-medium">Já Retirados:</span>
              <span className="text-2xl font-bold
text-yellow-600">{cotaEmpresa.quantidadeUsada}</span>
            </div>
            <div className="flex justify-between items-center pt-2 border-t border-gray-300">
              <span className="text-gray-700 font-medium">Disponíveis:</span>
              <span className="text-2xl font-bold
text-green-600">{cotaEmpresa.quantidadeTotal - cotaEmpresa.quantidadeUsada}</span>
```

```
            </div>
          </div>

          <div className="space-y-3">
            <button
              onClick={handleRetirarAgora}
              disabled={isResgatandoIngresso}
              className="w-full bg-black text-white py-4 rounded-xl hover:bg-gray-800
transition font-semibold text-lg border-2 border-black disabled:bg-gray-400"
              data-testid="button-retirar-agora"
            >
              {isResgatandoIngresso ? 'Resgatando...' : 'Retirar Agora'}
            </button>

            <button
              onClick={handleColocarNome}
              disabled={isResgatandoIngresso}
              className="w-full bg-yellow-500 text-black py-4 rounded-xl hover:bg-yellow-600
transition font-semibold text-lg disabled:bg-gray-300"
              data-testid="button-colocar-nome"
            >
              Colocar Nome no Convite
            </button>

            <button
              type="button"
              onClick={() => {
                setCotaEmpresa(null);
                setNomeEmpresa("");
                setEtapa('qual-empresa');
              }}
              className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
              data-testid="button-voltar-empresa"
            >
              Voltar
            </button>
          </div>

          {/* Carrossel compacto de patrocinadores */}
          <BannerCarousel compact={true} />
        </div>
      )}

      {/* Tela: Formulário de Dados do Convite */}
      {etapa === 'form-dados-convite' && cotaEmpresa && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
```

```
          <h2 className="text-2xl font-bold text-center mb-4"
data-testid="titulo-dados-convite">
            Dados do Convite
          </h2>

          <p className="text-center text-gray-600 mb-6">
            {cotaEmpresa.nomeEmpresa} - Preencha os dados de quem irá usar este convite
          </p>

          <form onSubmit={handleSubmitDadosConvite} className="space-y-4">
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Nome
Completo</label>
              <input
                type="text"
                value={nomeConvite}
                onChange={(e) => setNomeConvite(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="Digite o nome"
                data-testid="input-nome-convite"
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700
mb-2">E-mail</label>
              <input
                type="email"
                value={emailConvite}
                onChange={(e) => setEmailConvite(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="Digite o e-mail"
                data-testid="input-email-convite"
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Telefone
(WhatsApp)</label>
              <input
                type="tel"
                value={telefoneConvite}
                onChange={(e) => setTelefoneConvite(e.target.value)}
                required
```

```
                  className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                  placeholder="(31) 99999-9999"
                  data-testid="input-telefone-convite"
                 />
                </div>

                <div className="space-y-3 pt-4">
                 <button
                  type="submit"
                  disabled={isResgatandoIngresso}
                  className="w-full bg-yellow-500 text-black py-4 rounded-xl hover:bg-yellow-600
transition font-semibold text-lg disabled:bg-gray-300"
                  data-testid="button-confirmar-dados-convite"
                 >
                  {isResgatandoIngresso ? 'Gerando Convite...' : 'Confirmar e Gerar Convite'}
                 </button>

                 <button
                  type="button"
                  onClick={() => setEtapa('opcoes-resgate')}
                  className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
                  data-testid="button-voltar-opcoes"
                 >
                  Voltar
                 </button>
                </div>
               </form>

               {/* Carrossel compacto de patrocinadores */}
               <BannerCarousel compact={true} />
              </div>
             )}

             {/* Formulário para Pix - TELA ÚNICA DE DADOS + ESCOLHA DE MÉTODO */}
             {etapa === 'formulario-pix' && (
              <div className="bg-black border-4 border-yellow-400 rounded-3xl p-8">
               <button
                onClick={voltarInicial}
                className="text-yellow-400 hover:text-yellow-300 mb-6 flex items-center gap-2
transition-colors"
                data-testid="button-voltar-top"
               >
                <span>← Voltar</span>
               </button>
```

```
          <h2 className="text-white text-3xl font-bold text-center mb-4"
data-testid="titulo-formulario-pix">
            Resgate de Ingresso Avulso
          </h2>

          <p className="text-gray-300 text-center mb-8">
            Preencha seus dados para continuar
          </p>

          <form onSubmit={handleFormSubmit} className="space-y-6">
            {/* Nome */}
            <div>
              <label className="text-white font-semibold mb-2 block">Nome Completo
*</label>
              <input
                type="text"
                value={nome}
                onChange={(e) => setNome(e.target.value)}
                required
                className="w-full h-14 bg-white text-black text-lg rounded-xl px-4
placeholder:text-gray-400"
                placeholder="Digite seu nome completo"
                data-testid="input-nome"
              />
            </div>

            {/* Email */}
            <div>
              <label className="text-white font-semibold mb-2 block">E-mail *</label>
              <input
                type="email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                required
                className="w-full h-14 bg-white text-black text-lg rounded-xl px-4
placeholder:text-gray-400"
                placeholder="seu@email.com"
                data-testid="input-email"
              />
            </div>

            {/* Telefone */}
            <div>
              <label className="text-white font-semibold mb-2 block">Telefone (WhatsApp)
*</label>
              <input
                type="tel"
                value={telefone}
```

```
                onChange={(e) => setTelefone(e.target.value)}
                required
                className="w-full h-14 bg-white text-black text-lg rounded-xl px-4
placeholder:text-gray-400"
                placeholder="(31) 99999-9999"
                data-testid="input-telefone"
              />
            </div>

            {/* Botão Confirmar */}
            <button
              type="submit"
              className="w-full h-14 bg-gray-300 hover:bg-gray-200 text-black text-lg
font-semibold rounded-xl transition-colors"
              data-testid="button-confirmar-pix"
            >
              Confirmar e ver QR Code Pix
            </button>
          </form>

          {/* Carrossel compacto de patrocinadores */}
          <div className="mt-8">
            <BannerCarousel compact={true} />
          </div>
        </div>
      )}

      {/* QR Code Pix */}
      {etapa === 'qr-code' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-4" data-testid="titulo-qr-code">
            Pix Copia e Cola
          </h2>

          {/* Código de Identificação */}
          {pixTxid && (
            <div className="bg-yellow-50 border-2 border-yellow-400 rounded-lg p-4 mb-4">
              <p className="text-sm font-semibold text-yellow-800 mb-1">📋 Código de
Identificação:</p>
              <p className="text-2xl font-bold text-yellow-900 text-center">{pixTxid}</p>
              <p className="text-xs text-yellow-700 mt-2 text-center">
                Guarde este código para acompanhar seu pagamento
              </p>
            </div>
          )}

          <div className="bg-gray-50 p-6 rounded-xl mb-6">
            <div className="flex justify-center mb-4">
```

```
            <div className="bg-white p-4 rounded-xl">
              <QRCode value={pixPayload} size={200} data-testid="qr-code" />
            </div>
          </div>

          <p className="text-center text-gray-600 mb-4 leading-relaxed">
            Escaneie o QR Code acima<br />
            com o app do seu banco
          </p>

          <div className="border-t border-gray-200 pt-4">
            <p className="text-sm font-semibold text-gray-700 mb-2">Ou copie o código
Pix:</p>
            <div className="bg-white border border-gray-300 rounded-lg p-3 mb-3 break-all
text-xs font-mono text-gray-700">
              {pixPayload}
            </div>
            <button
              onClick={() => {
                navigator.clipboard.writeText(pixPayload);
                toast({
                  title: "Código Pix copiado!",
                  description: "Faça o pagamento no app do seu banco",
                });
              }}
              className="w-full bg-yellow-500 text-black py-3 rounded-xl hover:bg-yellow-600
transition font-semibold"
              data-testid="button-copiar-pix"
            >
              Copiar código Pix
            </button>
          </div>
        </div>

        <div className="bg-blue-50 border border-blue-200 rounded-lg p-4 mb-4">
          <p className="text-sm text-blue-800">
            <strong>💡 Como pagar:</strong>
          </p>
          <ol className="text-sm text-blue-700 mt-2 space-y-1 list-decimal list-inside">
            <li>Abra o app do seu banco</li>
            <li>Escolha "Pagar com Pix"</li>
            <li>Escaneie o QR Code ou cole o código</li>
            <li>Confirme o pagamento de R$ 1.000,00</li>
            <li>Aguarde a confirmação manual do pagamento pela equipe</li>
          </ol>
        </div>

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

```
            <p className="text-sm text-green-800">
              <strong>✅ Próximos passos:</strong>
            </p>
            <p className="text-sm text-green-700 mt-2">
              Após realizar o pagamento PIX, nossa equipe confirmará seu ingresso
manualmente.
              Você receberá uma confirmação por WhatsApp no número {telefone}.
            </p>
          </div>

          <button
            onClick={voltarInicial}
            className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition mt-4"
            data-testid="button-voltar-inicial"
          >
            Voltar ao início
          </button>
        </div>
      )}

      {/* Formulário de Cartão (sem dados de cliente) */}
      {etapa === 'formulario-cartao' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-formulario-cartao">
            Seus dados
          </h2>

          <form onSubmit={(e) => {
            e.preventDefault();
            setEtapa('escolher-metodo-pagamento');
          }} className="space-y-4">
            {/* Nome */}
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Nome
Completo</label>
              <input
                type="text"
                value={nome}
                onChange={(e) => setNome(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="Digite seu nome"
                data-testid="input-nome-cartao"
              />
            </div>
```

```
            {/* Email */}
            <div>
              <label className="block text-sm font-medium text-gray-700
mb-2">E-mail</label>
              <input
                type="email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="seu@email.com"
                data-testid="input-email-cartao"
              />
            </div>

            {/* Telefone */}
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Telefone
(WhatsApp)</label>
              <input
                type="tel"
                value={telefone}
                onChange={(e) => setTelefone(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="(31) 99999-9999"
                data-testid="input-telefone-cartao"
              />
            </div>

            {/* Botões */}
            <div className="space-y-3 pt-4">
              <button
                type="submit"
                disabled={isCreatingPayment}
                className="w-full bg-yellow-500 text-black py-4 rounded-xl hover:bg-yellow-600
transition font-semibold text-lg disabled:bg-gray-300"
                data-testid="button-continuar-cartao"
              >
                {isCreatingPayment ? 'Preparando...' : 'Continuar para pagamento'}
              </button>

              <button
                type="button"
                onClick={voltarInicial}
```

```
              className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
              data-testid="button-voltar-cartao"
            >
              Voltar
            </button>
          </div>
        </form>
      </div>
    )}

    {/* Escolher Método de Pagamento (PIX ou Cartão) */}
    {etapa === 'escolher-metodo-pagamento' && (
      <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
        <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-escolher-metodo">
          Escolha a forma de pagamento
        </h2>

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

          {/* Cartão (Cielo) com seletor de quantidade */}
          <div className="border-2 border-cyan-400 rounded-xl p-4">
            <h3 className="font-semibold text-lg mb-3 text-center">Pagar com Cartão ou
PIX</h3>

            {/* Seletor de Quantidade */}
            <div className="mb-4">
              <label className="block text-sm font-medium text-gray-700 mb-2 text-center">
                Quantos ingressos você quer comprar?
              </label>
              <div className="flex gap-2 justify-center flex-wrap">
                {[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((qtd) => (
                  <button
                    key={qtd}
                    type="button"
                    onClick={() => setQuantidadeIngressos(qtd)}
                    className={`px-4 py-2 rounded-lg font-semibold transition ${
                      quantidadeIngressos === qtd
                        ? 'bg-cyan-500 text-white'
                        : 'bg-gray-200 text-gray-700 hover:bg-gray-300'
                    }`}
                    data-testid={`button-quantidade-cielo-${qtd}`}
                  >
                    {qtd}
                  </button>
                ))}
              </div>
```

```
              {/* Total */}
              <div className="mt-3 text-center">
                <p className="text-sm text-gray-600">
                  {quantidadeIngressos} ingresso{quantidadeIngressos > 1 ? 's' : ''} × R$
1.000,00
                </p>
                <p className="text-xl font-bold text-cyan-600">
                  Total: R$ {(quantidadeIngressos * 1000)
                    .toLocaleString('pt-BR', { minimumFractionDigits: 2, maximumFractionDigits:
2 })}
                </p>
              </div>
            </div>

            {/* Botão pagar com Cielo */}
            <button
              onClick={() => handleCieloClick(quantidadeIngressos)}
              disabled={isCreatingPayment}
              className="w-full bg-cyan-500 text-white py-4 rounded-xl hover:bg-cyan-600
transition font-semibold text-lg disabled:bg-gray-300"
              data-testid="button-pagar-cartao-cielo"
            >
              {isCreatingPayment ? 'Preparando...' : 'Pagar com Cartão ou PIX'}
            </button>

            <p className="text-sm text-gray-500 text-center mt-2">
              Abre checkout Cielo em nova janela
            </p>
          </div>

          {/* Botão Voltar */}
          <button
            onClick={voltarInicial}
            className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
            data-testid="button-voltar-metodo"
          >
            Voltar
          </button>
        </div>
      </div>
    )}

    {/* Verificando Pagamento */}
    {etapa === 'verificando-pagamento' && (
      <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
        <h2 className="text-2xl font-bold text-center mb-6" data-testid="titulo-verificando">
```

```
            Verificando pagamento...
          </h2>
          <div className="flex justify-center py-8">
            <div className="animate-spin rounded-full h-16 w-16 border-b-2
border-yellow-500"></div>
          </div>
          <p className="text-center text-gray-600">Por favor, aguarde enquanto verificamos
seu pagamento.</p>
        </div>
      )}

      {/* Formulário Rede - Quantidade + Parcelas + Cartão */}
      {etapa === 'formulario-rede' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-formulario-rede">
            Pagamento com Cartão - Rede
          </h2>

          <form onSubmit={handleRedeSubmit} className="space-y-4">
            {/* Quantidade */}
            <div>
              <label className="block text-sm font-medium mb-2">Quantidade de
Ingressos</label>
              <select
                value={redeQty}
                onChange={(e) => setRedeQty(Number(e.target.value))}
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="select-qty-rede"
              >
                {[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((n) => (
                  <option key={n} value={n}>{n} ingresso{n > 1 ? 's' : ''} - R$ {(n *
1000).toLocaleString('pt-BR')}</option>
                ))}
              </select>
            </div>

            {/* Parcelas */}
            <div>
              <label className="block text-sm font-medium mb-2">Parcelas (sem
juros)</label>
              <select
                value={redeInstallments}
                onChange={(e) => setRedeInstallments(Number(e.target.value))}
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="select-installments-rede"
              >
                {[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((n) => (
```

```
                <option key={n} value={n}>
                  {n}x de R$ {((redeQty * 1000) / n).toLocaleString('pt-BR', {
minimumFractionDigits: 2 })}
                </option>
              ))}
            </select>
          </div>

          {/* Número do Cartão */}
          <div>
            <label className="block text-sm font-medium mb-2">Número do Cartão</label>
            <input
              type="text"
              value={redeCardNumber}
              onChange={(e) => {
                const val = e.target.value.replace(/\D/g, '').slice(0, 16);
                const formatted = val.match(/.{1,4}/g)?.join(' ') || val;
                setRedeCardNumber(formatted);
              }}
              placeholder="1234 5678 9012 3456"
              className="w-full border border-gray-300 rounded-xl p-3"
              data-testid="input-card-number-rede"
              required
            />
          </div>

          {/* Nome no Cartão */}
          <div>
            <label className="block text-sm font-medium mb-2">Nome no Cartão</label>
            <input
              type="text"
              value={redeCardName}
              onChange={(e) => setRedeCardName(e.target.value.toUpperCase())}
              placeholder="NOME COMPLETO"
              className="w-full border border-gray-300 rounded-xl p-3"
              data-testid="input-card-name-rede"
              required
            />
          </div>

          {/* Validade e CVV */}
          <div className="grid grid-cols-3 gap-4">
            <div>
              <label className="block text-sm font-medium mb-2">Mês</label>
              <input
                type="text"
                value={redeCardMonth}
```

```
                onChange={(e) => setRedeCardMonth(e.target.value.replace(/\D/g, '').slice(0,
2))}
                placeholder="MM"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-month-rede"
                required
              />
            </div>
            <div>
              <label className="block text-sm font-medium mb-2">Ano</label>
              <input
                type="text"
                value={redeCardYear}
                onChange={(e) => setRedeCardYear(e.target.value.replace(/\D/g, '').slice(0, 4))}
                placeholder="AAAA"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-year-rede"
                required
              />
            </div>
            <div>
              <label className="block text-sm font-medium mb-2">CVV</label>
              <input
                type="text"
                value={redeCardCvv}
                onChange={(e) => setRedeCardCvv(e.target.value.replace(/\D/g, '').slice(0, 4))}
                placeholder="123"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-cvv-rede"
                required
              />
            </div>
          </div>

          {/* Total */}
          <div className="bg-gray-50 p-4 rounded-xl">
            <div className="flex justify-between items-center">
              <span className="font-semibold">Total:</span>
              <span className="text-2xl font-bold text-blue-600">
                R$ {(redeQty * 1000).toLocaleString('pt-BR')}
              </span>
            </div>
            <div className="text-sm text-gray-600 mt-1">
              {redeInstallments}x de R$ {((redeQty * 1000) /
redeInstallments).toLocaleString('pt-BR', { minimumFractionDigits: 2 })}
            </div>
          </div>
```

```
            {/* Botões */}
            <button
              type="submit"
              disabled={isCreatingPayment}
              className="w-full bg-blue-500 text-white py-4 rounded-xl hover:bg-blue-600
transition font-semibold disabled:bg-gray-300"
              data-testid="button-submit-rede"
            >
              {isCreatingPayment ? 'Processando...' : 'Confirmar Pagamento'}
            </button>

            <button
              type="button"
              onClick={() => setEtapa('escolher-metodo-pagamento')}
              className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
              data-testid="button-voltar-rede"
            >
              Voltar
            </button>
          </form>
        </div>
      )}

      {/* Formulário Cielo - Quantidade + Parcelas + Cartão */}
      {etapa === 'formulario-cielo' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-formulario-cielo">
            Pagamento com Cartão - Cielo
          </h2>

          <form onSubmit={handleCieloSubmit} className="space-y-4">
            {/* Quantidade */}
            <div>
              <label className="block text-sm font-medium mb-2">Quantidade de
Ingressos</label>
              <select
                value={cieloQty}
                onChange={(e) => setCieloQty(Number(e.target.value))}
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="select-qty-cielo"
              >
                {[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((n) => (
                  <option key={n} value={n}>{n} ingresso{n > 1 ? 's' : ''} - R$ {(n *
1000).toLocaleString('pt-BR')}</option>
                ))}
              </select>
```

```
          </div>

          {/* Parcelas */}
          <div>
            <label className="block text-sm font-medium mb-2">Parcelas (sem
juros)</label>
            <select
              value={cieloInstallments}
              onChange={(e) => setCieloInstallments(Number(e.target.value))}
              className="w-full border border-gray-300 rounded-xl p-3"
              data-testid="select-installments-cielo"
            >
              {[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].map((n) => (
                <option key={n} value={n}>
                  {n}x de R$ {((cieloQty * 1000) / n).toLocaleString('pt-BR', {
minimumFractionDigits: 2 })}
                </option>
              ))}
            </select>
          </div>

          {/* Número do Cartão */}
          <div>
            <label className="block text-sm font-medium mb-2">Número do Cartão</label>
            <input
              type="text"
              value={cieloCardNumber}
              onChange={(e) => {
                const val = e.target.value.replace(/\D/g, '').slice(0, 16);
                const formatted = val.match(/.{1,4}/g)?.join(' ') || val;
                setCieloCardNumber(formatted);
              }}
              placeholder="1234 5678 9012 3456"
              className="w-full border border-gray-300 rounded-xl p-3"
              data-testid="input-card-number-cielo"
              required
            />
          </div>

          {/* Nome no Cartão */}
          <div>
            <label className="block text-sm font-medium mb-2">Nome no Cartão</label>
            <input
              type="text"
              value={cieloCardName}
              onChange={(e) => setCieloCardName(e.target.value.toUpperCase())}
              placeholder="NOME COMPLETO"
              className="w-full border border-gray-300 rounded-xl p-3"
```

```
              data-testid="input-card-name-cielo"
              required
            />
          </div>

          {/* Validade e CVV */}
          <div className="grid grid-cols-3 gap-4">
            <div>
              <label className="block text-sm font-medium mb-2">Mês</label>
              <input
                type="text"
                value={cieloCardMonth}
                onChange={(e) => setCieloCardMonth(e.target.value.replace(/\D/g, '').slice(0,
2))}
                placeholder="MM"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-month-cielo"
                required
              />
            </div>
            <div>
              <label className="block text-sm font-medium mb-2">Ano</label>
              <input
                type="text"
                value={cieloCardYear}
                onChange={(e) => setCieloCardYear(e.target.value.replace(/\D/g, '').slice(0, 4))}
                placeholder="AAAA"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-year-cielo"
                required
              />
            </div>
            <div>
              <label className="block text-sm font-medium mb-2">CVV</label>
              <input
                type="text"
                value={cieloCardCvv}
                onChange={(e) => setCieloCardCvv(e.target.value.replace(/\D/g, '').slice(0, 4))}
                placeholder="123"
                className="w-full border border-gray-300 rounded-xl p-3"
                data-testid="input-card-cvv-cielo"
                required
              />
            </div>
          </div>

          {/* Total */}
          <div className="bg-gray-50 p-4 rounded-xl">
```

```
              <div className="flex justify-between items-center">
                <span className="font-semibold">Total:</span>
                <span className="text-2xl font-bold text-cyan-600">
                  R$ {(cieloQty * 1000).toLocaleString('pt-BR')}
                </span>
              </div>
              <div className="text-sm text-gray-600 mt-1">
                {cieloInstallments}x de R$ {((cieloQty * 1000) /
cieloInstallments).toLocaleString('pt-BR', { minimumFractionDigits: 2 })}
              </div>
            </div>

            {/* Botões */}
            <button
              type="submit"
              disabled={isCreatingPayment}
              className="w-full bg-cyan-500 text-white py-4 rounded-xl hover:bg-cyan-600
transition font-semibold disabled:bg-gray-300"
              data-testid="button-submit-cielo"
            >
              {isCreatingPayment ? 'Processando...' : 'Confirmar Pagamento'}
            </button>

            <button
              type="button"
              onClick={() => setEtapa('escolher-metodo-pagamento')}
              className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
              data-testid="button-voltar-cielo"
            >
              Voltar
            </button>
          </form>
        </div>
      )}

      {/* Cadastrar Usuário - Tela Única */}
      {etapa === 'cadastrar-usuario' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <h2 className="text-2xl font-bold text-center mb-6"
data-testid="titulo-cadastrar-usuario">
            Usuário não encontrado. Complete seu cadastro:
          </h2>

          <form onSubmit={async (e) => {
            e.preventDefault();

            // Cadastrar usuário no banco
```

```
        try {
          const response = await fetch('/api/usuarios/cadastrar', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({
              nome,
              telefone,
              nomeEmpresa: temEmpresa ? nomeEmpresa : null
            }),
          });

          const resultado = await response.json();

          if (resultado.success) {
            // Cadastro feito! Agora resgatar o ingresso
            toast({
              title: "Cadastro realizado!",
              description: "Agora vamos resgatar seu ingresso...",
            });

            // Chamar endpoint de resgate
            setTimeout(async () => {
              try {
                const resgateResponse = await fetch('/api/ingresso/resgatar', {
                  method: 'POST',
                  headers: { 'Content-Type': 'application/json' },
                  body: JSON.stringify({
                    telefone,
                    nomeEmpresa: temEmpresa ? nomeEmpresa : null
                  }),
                });

                const resgateResultado = await resgateResponse.json();

                if (resgateResultado.success) {
                  // Salvar no localStorage
                  localStorage.setItem('ingressoResgatadoNumero',
resgateResultado.ingresso.numero);

                  // Redirecionar para visualização
                  window.location.href =
`/ingresso/visualizar/${resgateResultado.ingresso.numero}`;
                } else {
                  toast({
                    title: "Erro ao resgatar",
                    description: resgateResultado.error || "Erro ao resgatar ingresso",
                    variant: "destructive",
                  });
```

```
                }
              } catch (error) {
                console.error('Erro ao resgatar:', error);
                toast({
                  title: "Erro ao resgatar",
                  description: "Tente novamente.",
                  variant: "destructive",
                });
              }
            }, 1000);
          } else {
            toast({
              title: "Erro ao cadastrar",
              description: resultado.mensagem || "Ocorreu um erro ao cadastrar usuário",
              variant: "destructive",
            });
          }
        } catch (error) {
          console.error('Erro ao cadastrar usuário:', error);
          toast({
            title: "Erro ao cadastrar",
            description: "Por favor, tente novamente.",
            variant: "destructive",
          });
        }
      }} className="space-y-4">
        {/* Nome */}
        <div>
          <label className="block text-sm font-medium text-gray-700 mb-2">Nome</label>
          <input
            type="text"
            value={nome}
            disabled
            className="w-full px-4 py-3 border border-gray-300 rounded-xl bg-gray-100
text-gray-700"
            data-testid="input-nome-cadastro"
          />
        </div>

        {/* Telefone */}
        <div>
          <label className="block text-sm font-medium text-gray-700
mb-2">Telefone</label>
          <input
            type="tel"
            value={telefone}
            disabled
```

```
              className="w-full px-4 py-3 border border-gray-300 rounded-xl bg-gray-100
text-gray-700"
              data-testid="input-telefone-cadastro"
             />
            </div>

            {/* Tem Empresa? */}
            <div>
             <label className="block text-sm font-medium text-gray-700
mb-3">Empresa?</label>
             <div className="flex gap-3 mb-4">
              <button
               type="button"
               onClick={() => setTemEmpresa(true)}
               className={`flex-1 py-3 rounded-xl font-semibold transition ${
                temEmpresa === true
                 ? 'bg-yellow-500 text-black'
                 : 'bg-gray-200 text-gray-700 hover:bg-gray-300'
               }`}
               data-testid="button-empresa-sim"
              >
               Sim
              </button>
              <button
               type="button"
               onClick={() => {
                setTemEmpresa(false);
                setNomeEmpresa('');
               }}
               className={`flex-1 py-3 rounded-xl font-semibold transition ${
                temEmpresa === false
                 ? 'bg-yellow-500 text-black'
                 : 'bg-gray-200 text-gray-700 hover:bg-gray-300'
               }`}
               data-testid="button-empresa-nao"
              >
               Não
              </button>
             </div>
            </div>

            {/* Campo Empresa (se Sim) */}
            {temEmpresa === true && (
             <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">Nome da
Empresa</label>
              <input
               type="text"
```

```
                value={nomeEmpresa}
                onChange={(e) => setNomeEmpresa(e.target.value)}
                required
                className="w-full px-4 py-3 border border-gray-300 rounded-xl
focus:outline-none focus:ring-2 focus:ring-yellow-500"
                placeholder="Digite o nome da empresa"
                data-testid="input-nome-empresa"
              />
            </div>
          )}

          {/* Botão Continuar (aparece após escolher Sim ou Não) */}
          {temEmpresa !== null && (
            <button
              type="submit"
              className="w-full bg-green-500 text-white py-4 rounded-xl hover:bg-green-600
transition font-semibold text-lg"
              data-testid="button-continuar-cadastro"
            >
              Continuar
            </button>
          )}
        </form>
      </div>
    )}

    {/* Modal de Pagamento Stripe */}
    {etapa === 'pagamento-cartao' && clientSecret && (
      <Elements stripe={stripePromise} options={{ clientSecret }}>
        <StripePaymentModal
          clientSecret={clientSecret}
          onSuccess={async () => {
            // Verificar pagamento e ir para associar empresa
            await verificarPagamento();
          }}
          onCancel={() => {
            setEtapa('card-inicial');
            setClientSecret("");
          }}
          nomeComprador={nome}
          telefoneComprador={telefone}
          quantidade={quantidadeIngressos}
          valorTotal={100000 * quantidadeIngressos}
        />
      </Elements>
    )}

    {/* Tela de Confirmação Pagamento Rede */}
```

```
      {etapa === 'pagamento-rede' && (
        <div className="bg-white border-2 border-gray-200 rounded-xl p-6">
          <div className="text-center mb-6">
            <div className="bg-blue-100 rounded-full w-20 h-20 flex items-center
justify-center mx-auto mb-4">
              <span className="text-4xl">💳</span>
            </div>
            <h2 className="text-2xl font-bold mb-2" data-testid="titulo-pagamento-rede">
              Pagamento com Rede
            </h2>
            <p className="text-gray-600">
              Seu pagamento está sendo processado
            </p>
          </div>

          <div className="bg-gray-50 rounded-lg p-4 mb-6">
            <p className="text-sm text-gray-600">Comprador: {nome}</p>
            <p className="text-sm text-gray-600">Telefone: {telefone}</p>
            <p className="text-sm text-gray-600">Quantidade: 1 ingresso</p>
            <p className="font-bold text-lg mt-2">Valor: R$ 1.000,00</p>
          </div>

          <div className="flex justify-center py-8">
            <div className="animate-spin rounded-full h-16 w-16 border-b-2
border-blue-500"></div>
          </div>

          <p className="text-center text-sm text-gray-500 mb-4">
            Aguarde enquanto processamos seu pagamento...
          </p>

          <button
            onClick={() => {
              setEtapa('card-inicial');
              toast({
                title: "Pagamento cancelado",
                description: "Você pode tentar novamente quando quiser.",
              });
            }}
            className="w-full bg-gray-200 text-gray-700 py-3 rounded-xl hover:bg-gray-300
transition"
            data-testid="button-cancelar-rede"
          >
            Cancelar
          </button>
        </div>
      )}
```

```
        {/* Mensagem de Segurança LGPD */}
        <div className="mt-6 bg-gray-50 border border-gray-200 rounded-xl p-4">
          <p className="text-gray-700 text-xs text-center leading-relaxed">
            🔒 <span className="font-semibold">Ambiente 100% Seguro.</span> Seus dados pessoais e de pagamento estão protegidos. Seguimos rigorosamente todos os protocolos de segurança e as diretrizes da Lei Geral de Proteção de Dados (LGPD). Nenhuma informação do seu cartão é armazenada em nossos servidores.
          </p>
        </div>
      </div>
    </div>
  );
}
```