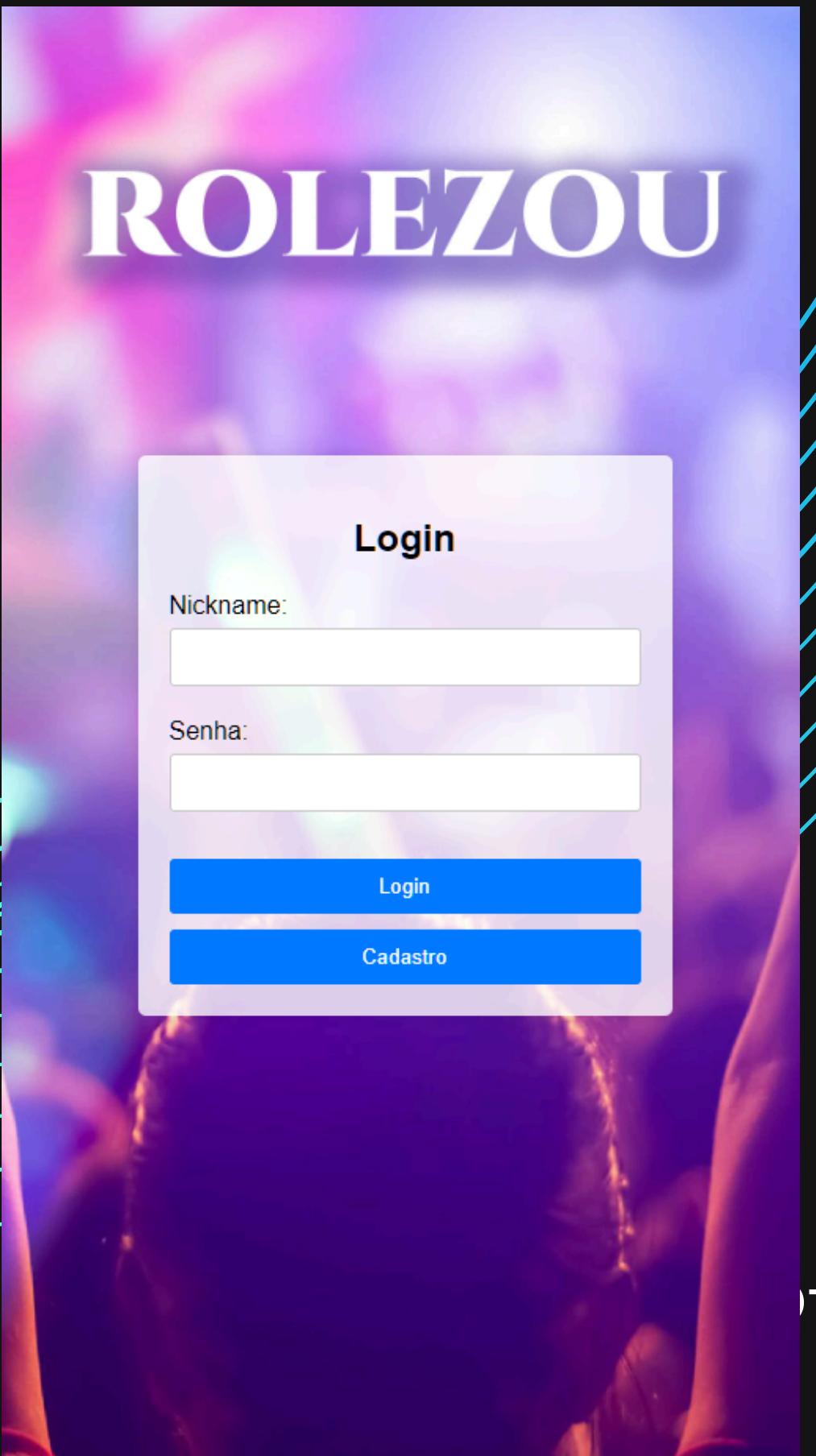


Projeto: Rolezou

SISTEMA DE DIVULGAÇÃO DE BALADAS E ATIVIDADES
UNIVERSITÁRIAS

CDIA MA3

Anigreice Marcolino
Bárbara Melo
Henrique Félix
Juliana Soares
Rebecca Campos



Objetivos:

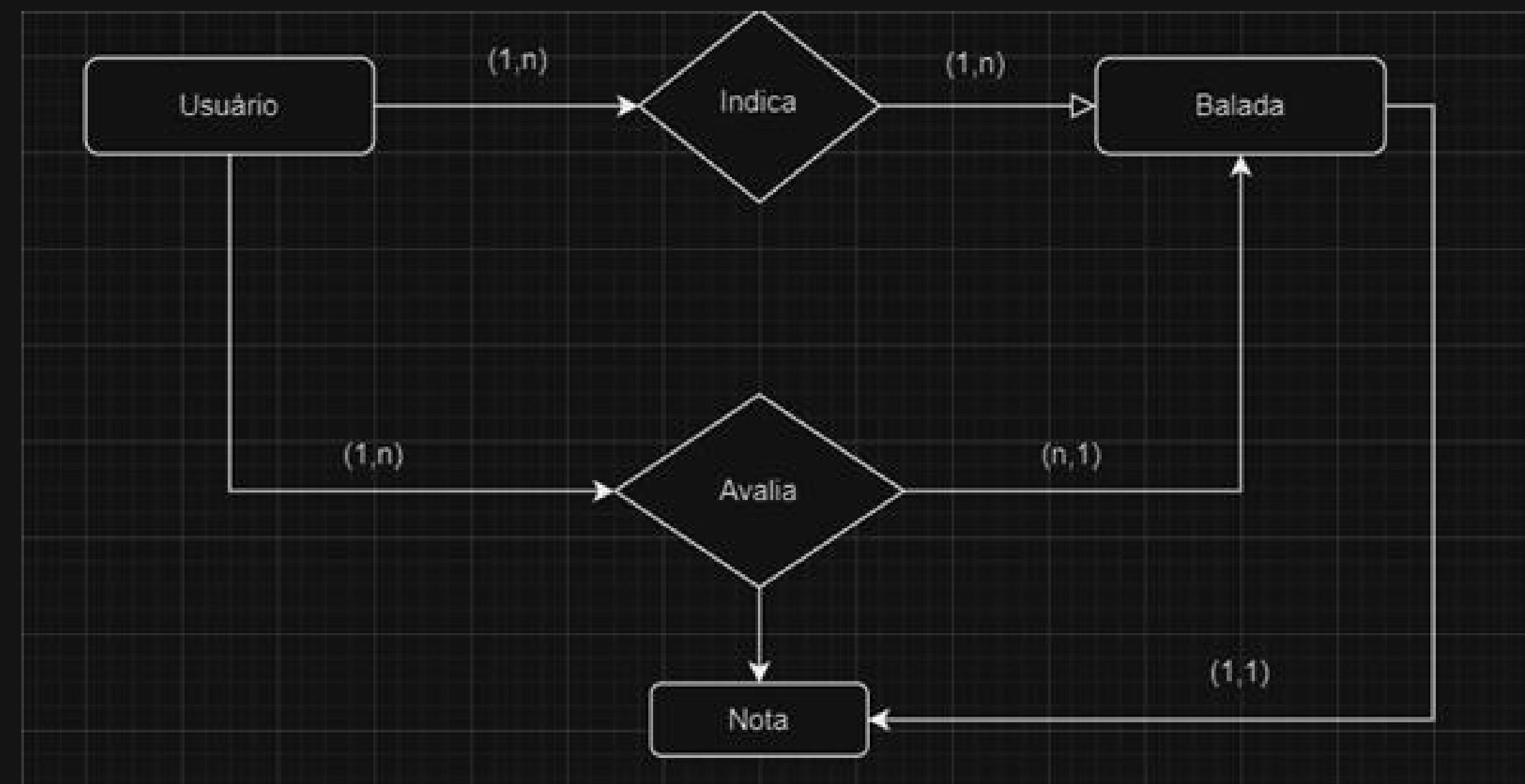
INDICAÇÃO DE EVENTO

O usuário deve ser capaz de indicar um evento com nome, local, preço e data.

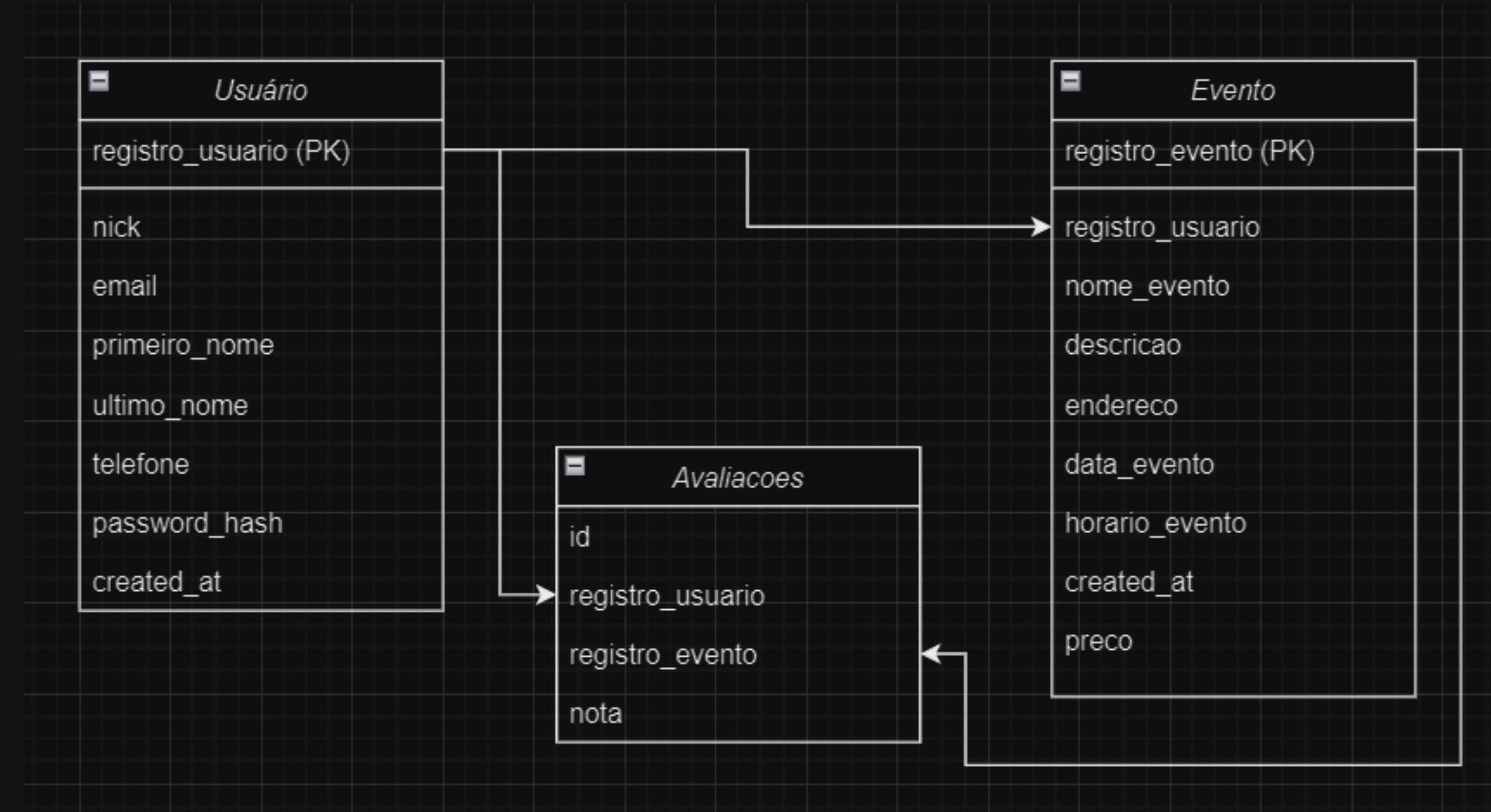
VISUALIZAÇÃO DE EVENTOS

O usuário deve ser capaz de pesquisar os eventos que irão acontecer e filtrar de acordo com a sua necessidade.

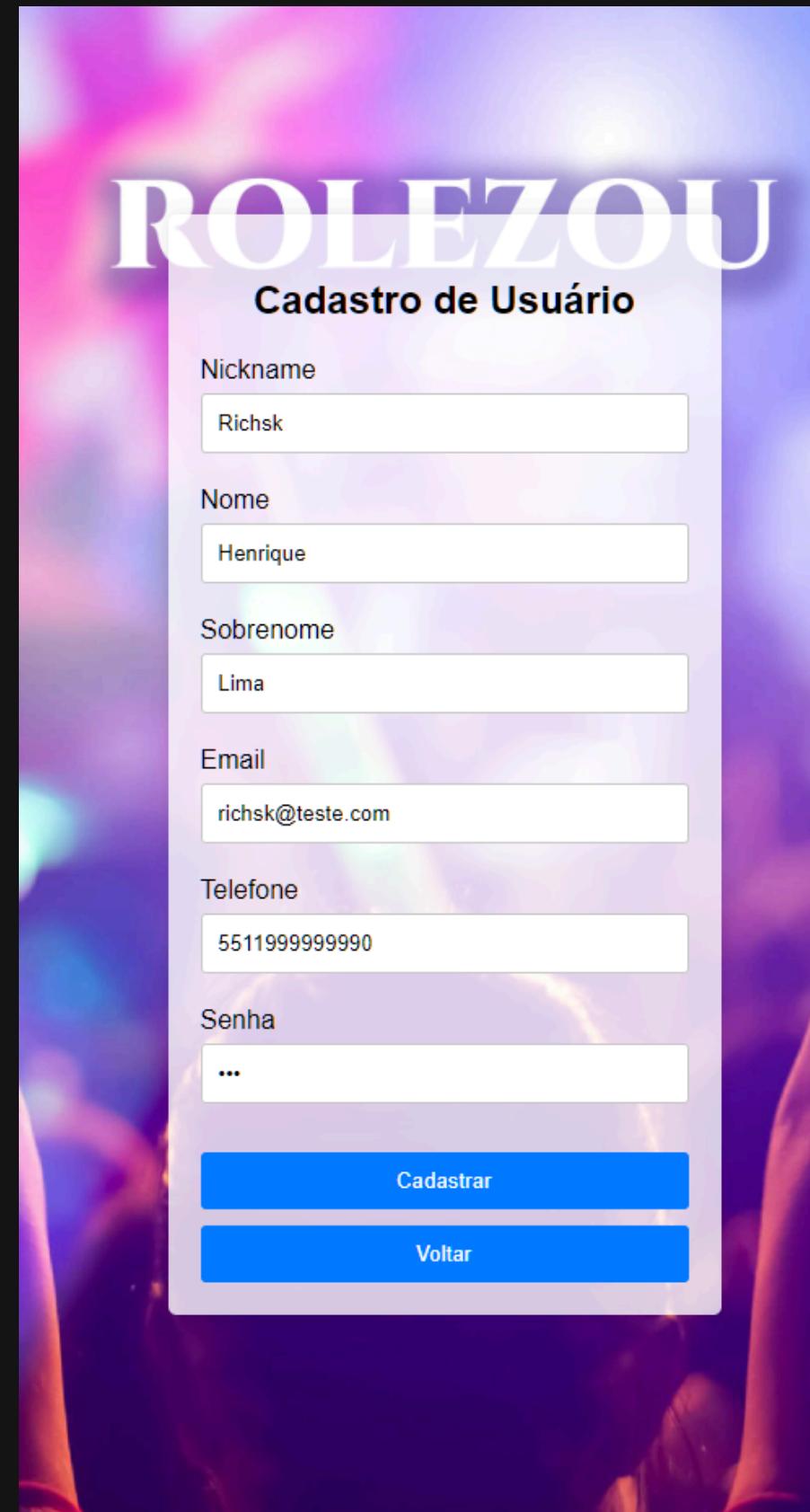
Modelo conceitual



Modelo lógico



Cadastro



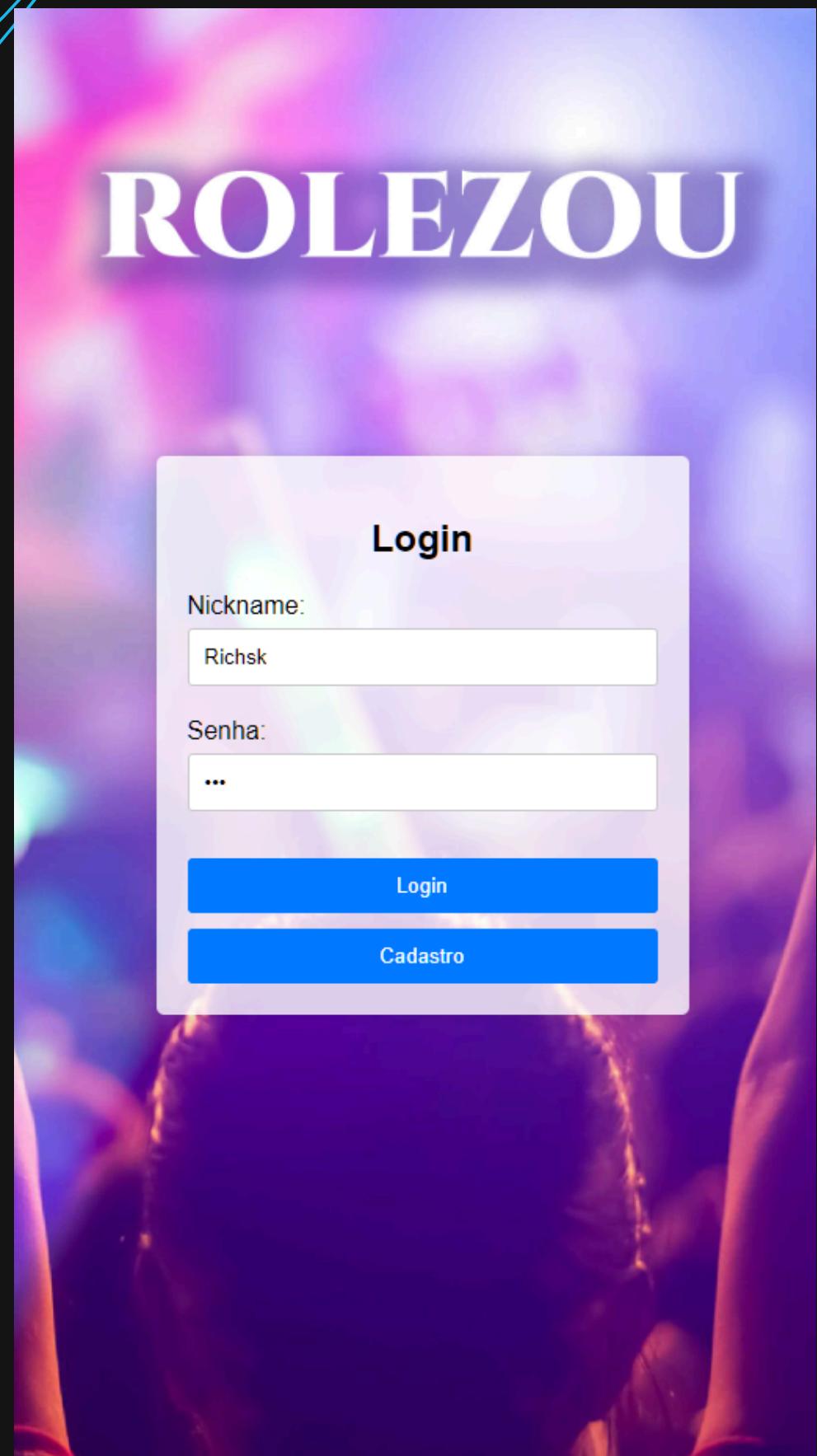
```
# Rota para a página de cadastro
@app.route('/cadastro', methods=['GET', 'POST'])
def cadastro():
    if request.method == 'POST':
        nick = request.form['nickname']
        primeiro_nome = request.form['nome']
        ultimo_nome = request.form['sobrenome']
        email = request.form['email']
        telefone = request.form['telefone']
        senha = request.form['senha']

        if nick and primeiro_nome and ultimo_nome and email and telefone and senha:
            try:
                conn = get_db_connection()
                cursor = conn.cursor()

                # Gerar hash da senha antes de armazenar no banco de dados
                password_hash = generate_password_hash(senha)

                cursor.execute("INSERT INTO usuarios (nick, primeiro_nome, ultimo_nome, email, telefone, password_hash) VALUES (%s, %s, %s, %s, %s, %s)", (nick, primeiro_nome, ultimo_nome, email, telefone, password_hash))
```

Login

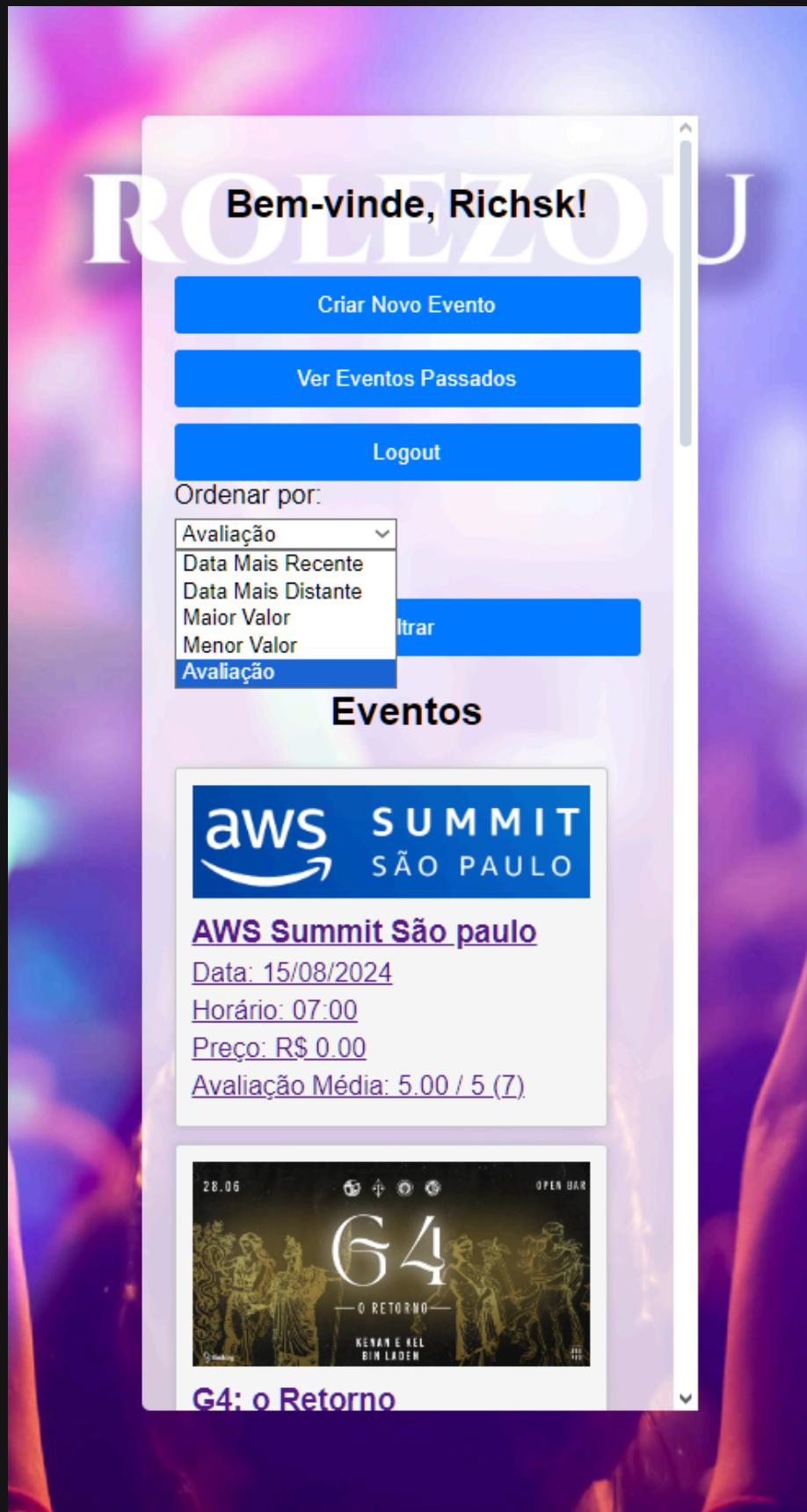


```
def verificar_credenciais(nick, password):
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT password_hash FROM usuarios WHERE nick = %s", (nick,))
    result = cursor.fetchone()

    if result is None:
        cursor.close()
        conn.close()
        return False

    stored_password_hash = result[0]
    cursor.close()
    conn.close()
    return check_password_hash(stored_password_hash, password)
```

Página inicial e filtros



```
conn = get_db_connection()
cursor = conn.cursor(dictionary=True)

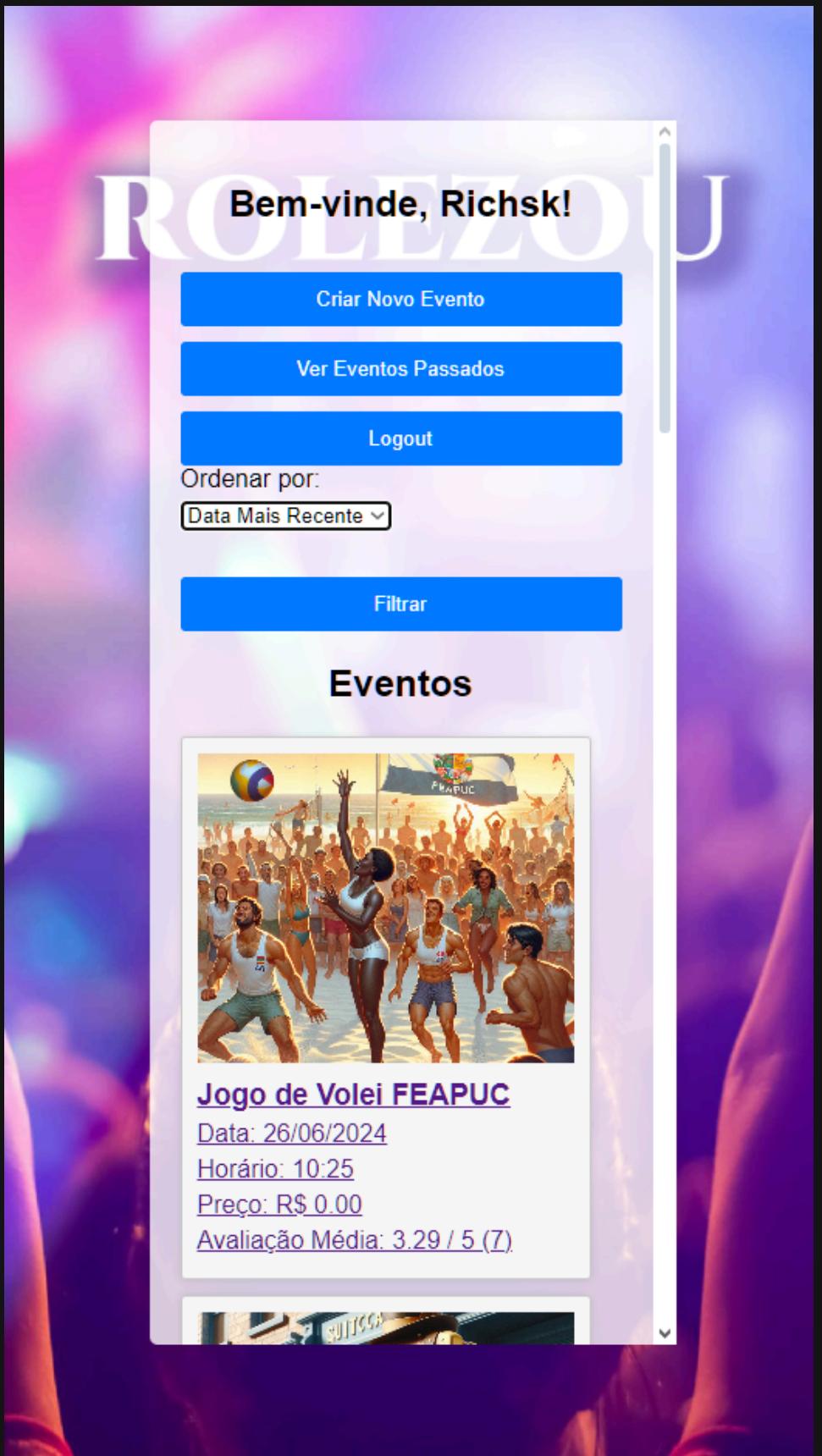
query = """
    SELECT e.registro_evento, e.nome_evento, e.descricao, e.endereco,
           DATE_FORMAT(e.data_evento, '%d/%m/%Y') AS data_formatada,
           e.horario_evento, e.preco, e.imagem, e.criado_por,
           COALESCE(AVG(a.notas), 0) as media_avaliacao,
           COUNT(a.notas) as num_avaliacoes
      FROM eventos e
     LEFT JOIN avaliacoes a ON e.registro_evento = a.registro_evento
   GROUP BY e.registro_evento, e.nome_evento, e.descricao, e.endereco,
           e.data_evento, e.horario_evento, e.preco, e.imagem, e.criado_por
"""

if ordenar_por == 'data_recente':
    query += " ORDER BY e.data_evento ASC"
elif ordenar_por == 'data_distante':
    query += " ORDER BY e.data_evento DESC"
elif ordenar_por == 'maior_valor':
    query += " ORDER BY e.preco DESC"
elif ordenar_por == 'menor_valor':
    query += " ORDER BY e.preco ASC"
elif ordenar_por == 'avaliacao':
    query += " ORDER BY media_avaliacao DESC"

cursor.execute(query)
eventos = cursor.fetchall()
cursor.close()
conn.close()

return render_template('index.html', eventos=eventos, ordenar_por=ordenar_por)
```

Página inicial e filtros



```
conn = get_db_connection()
cursor = conn.cursor(dictionary=True)

query = """
    SELECT e.registro_evento, e.nome_evento, e.descricao, e.endereco,
           DATE_FORMAT(e.data_evento, '%d/%m/%Y') AS data_formatada,
           e.horario_evento, e.preco, e.imagem, e.criado_por,
           COALESCE(AVG(a.notas), 0) as media_avaliacao,
           COUNT(a.notas) as num_avaliacoes
      FROM eventos e
     LEFT JOIN avaliacoes a ON e.registro_evento = a.registro_evento
    GROUP BY e.registro_evento, e.nome_evento, e.descricao, e.endereco,
             e.data_evento, e.horario_evento, e.preco, e.imagem, e.criado_por
    .....

if ordenar_por == 'data_recente':
    query += " ORDER BY e.data_evento ASC"
elif ordenar_por == 'data_distante':
    query += " ORDER BY e.data_evento DESC"
elif ordenar_por == 'maior_valor':
    query += " ORDER BY e.preco DESC"
elif ordenar_por == 'menor_valor':
    query += " ORDER BY e.preco ASC"
elif ordenar_por == 'avaliacao':
    query += " ORDER BY media_avaliacao DESC"

cursor.execute(query)
eventos = cursor.fetchall()
cursor.close()
conn.close()

return render_template('index.html', eventos=eventos, ordenar_por=ordenar_por)
```

Criando um novo evento

Criar Novo Evento

Título:

Descrição:

Endereço:

CEP:

Data:

Horário:

Imagen do Evento:

Escolher arquivo Nenhum arquivo escolhido

Preço:

Criar Evento

```
if titulo and descricao and endereco and cep and data and horario and file and preco and allowed_file(file.filename):  
  
    filename = secure_filename(file.filename)  
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)  
    file.save(filepath)  
    print(f'Saving file to: {filepath}')  
    file.save(filepath)  
    print('File saved successfully.')  
  
    try:  
        conn = get_db_connection()  
        cursor = conn.cursor()  
        cursor.execute("INSERT INTO eventos (registro_usuario, nome_evento, descricao, endereco, cep, data_evento, horario_evento, imagem, preco,  
        criador_id, titulo, descricao, endereco, cep, data, horario, filename, preco, criador)  
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)",  
        (criador_id, titulo, descricao, endereco, cep, data, horario, filename, preco, criador))  
        conn.commit()  
        flash('Evento criado com sucesso!', 'success')  
        return redirect(url_for('index'))  
    except mysql.connector.Error as err:  
        flash(f'Erro ao criar evento: {err}', 'danger')  
    finally:  
        cursor.close()  
        conn.close()  
  
else:  
    flash('Por favor, preencha todos os campos!', 'warning')
```

Pagina do evento



```
@app.route('/evento/<int:registro_evento>')
def evento(registro_evento):
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("SELECT registro_evento, nome_evento, descricao, endereco, DATE_FORMAT(data_evento, '%d/%m/%Y') AS data_formatada, horario_evento, preco, imagem FROM eventos WHERE registro_evento = %s", (registro_evento,))
    evento = cursor.fetchone()

    cursor.execute("SELECT AVG(nota) as media_avaliacao FROM avaliacoes WHERE registro_evento = %s", (registro_evento,))
    media_avaliacao_result = cursor.fetchone()

    cursor.close()
    conn.close()

    if evento:
        media_avaliacao = media_avaliacao_result['media_avaliacao'] if media_avaliacao_result and media_avaliacao_result['media_avaliacao'] is not None else 0
        return render_template('evento.html', evento=evento, media_avaliacao=media_avaliacao)
    else:
        flash('Evento não encontrado.', 'danger')
        return redirect(url_for('index'))
```

	registro_evento	registro_usuario	nome_evento	descricao	endereco	data_evento	horario_evento	created_at	preco	imagem
▶	10	10	Sunset - OPENBAR	Sábado (15h - 21:00h) - estamos de volta para ...	Rua Alagoas, 836, Higienópolis, São Paulo, SP	2024-06-29	15:00	2024-06-24 17:06:49	80.00	d110478a6e2840a7a3457f5cb860a4d2.jpg
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Sistema de avaliação



```
@app.route('/avaliar/<int:registro_evento>', methods=['POST'])
def avaliar(registro_evento):
    if not esta_autenticado():
        flash('Você precisa fazer login para avaliar um evento.', 'warning')
        return redirect(url_for('login'))

    nota = request.form['nota']
    usuario = session['usuario']

    try:
        conn = get_db_connection()
        cursor = conn.cursor()

        # Obter o ID do usuário
        cursor.execute("SELECT registro_usuario FROM usuarios WHERE nick = %s", (usuario,))
        registro_usuario = cursor.fetchone()[0]

        # Inserir ou atualizar a avaliação
        cursor.execute("""
            INSERT INTO avaliacoes (registro_usuario, registro_evento, nota)
            VALUES (%s, %s, %s)
            ON DUPLICATE KEY UPDATE nota = VALUES(nota)
        """, (registro_usuario, registro_evento, nota))

        conn.commit()
        flash('Avaliação salva com sucesso!', 'success')
    except mysql.connector.Error as err:
        flash(f'Erro ao salvar avaliação: {err}', 'danger')
    finally:
        cursor.close()
        conn.close()

    return redirect(url_for('evento', registro_evento=registro_evento))
```

id	registro_usuario	registro_evento	nota
1	4	1	5
15	4	2	1
16	4	3	5
17	4	4	3
18	2	1	5
19	2	2	3
21	2	3	5
22	2	4	1
30	5	1	4
31	5	2	5
32	5	3	5
33	5	4	1
35	11	9	4

Eventos passados

Eventos Passados



Pool Party no 5º Andar da PUC

Data: 2024-06-22
Horário: 17:37:00
Preço: R\$ 0.00
Criado por: Hello_Kitty



Set Brat

Data: 2024-06-22
Horário: 0:12:00
Preço: R\$ 600.00
Criado por: Hello_Kitty

```
# Verificar eventos passados
today = date.today()
cursor.execute("SELECT * FROM eventos WHERE data_evento < %s", (today,))
eventos_passados = cursor.fetchall()

# Mover eventos para a tabela de eventos passados
for evento in eventos_passados:
    try:
        cursor.execute("INSERT INTO eventos_passados (nome_evento, descricao, endereco, data_evento, horario_evento, preco, imagem, criado_por) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)", (evento['nome_evento'], evento['descricao'], evento['endereco'], evento['data_evento'], evento['horario_evento'], evento['preco'], evento['imagem'], evento['criado_por']))
        cursor.execute("DELETE FROM avaliacoes WHERE registro_evento = %s", (evento['registro_evento'],))
        cursor.execute("DELETE FROM eventos WHERE registro_evento = %s", (evento['registro_evento'],))
        conn.commit()
    except mysql.connector.Error as err:
        flash(f'Erro ao mover evento para eventos passados: {err}', 'danger')

# Fechar conexão com o banco de dados
cursor.close()
conn.close()
```

	registro_evento	nome_evento	descricao	endereco	data_evento	horario_evento	preco	imagem	criado_por	criado_em
▶	3	Pool Party no 5º Andar da PUC	Venha de biquíni pra puc vai ser super revolucionário!	Puc Perdizes	2024-06-22	17:37:00	0.00	campus_perdizes.jpg	Hello_Kitty	2024-06-23
▶	4	Set Brat	Charli XCX tocando seu dj set na Zig Club	R. Álvaro de Carvalho, 190 - Centro Histórico d...	2024-06-22	00:12:00	600.00	images.png	Hello_Kitty	2024-06-23
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL