



# Aplicação para compra e venda de ingressos

Versão 1.3

Trabalho referente à disciplina de  
Construção de Software do curso  
de Engenharia de Software.

GOIÂNIA  
2024.1

Arthur Moura Bernardo  
Joyce Beatriz Ferreira da Costa Silva  
Kaio Ribeiro Sanchez  
Pietro Niero Roque

# Aplicação para compra e venda de ingressos

Versão 1.3

Trabalho referente à disciplina de Construção de Software, ministrada pelo professor Elias Batista, do curso de Engenharia de Software.

GOIÂNIA  
2024.1

# Histórico de Alterações

Data	Versão	Descrição
30/05/2024	1.0	Estruturação do documento
03/05/2024	1.0	História de usuários e detalhamento das classes
24/06/2024	1.1	Casos de teste
25/06/2024	1.1	Revisão casos de teste
15/07/2024	1.2	Revisão casos de uso
21/07/2024	1.2	Revisão casos de teste
22/07/2024	1.2	Requisitos funcionais/não funcionais
23/07/2024	1.3	Estratégia de controle de versão
23/07/2024	1.3	Revisão da documentação

# Conteúdo

1 Introdução	5
2 Requisitos Funcionais / Não Funcionais	5
Requisitos Funcionais	5
Requisitos Não Funcionais	6
3 Casos de Uso/ Historia de Usuários	7
5 Testes	10
6 Estratégia de controle de versão	14
7 Repositórios e Recursos	15

# 1 Introdução

Desenvolvido em Java 8, com framework Spring, este sistema tem como objetivo principal oferecer uma solução robusta e escalável para a venda de ingressos em diversos eventos, sejam eles culturais, esportivos ou de entretenimento. A aplicação permitirá aos seus usuários criar eventos, vender os ingressos desses eventos e, posteriormente, revender os ingressos comprados para outros usuários.

Utilizando o banco de dados PostgreSQL, o software garantirá uma estrutura sólida para armazenamento e gerenciamento de dados, permitindo uma manipulação eficiente de informações relacionadas a eventos, clientes, transações e relatórios. O uso do Maven como ferramenta de gerenciamento de dependências e construção do projeto garante uma abordagem organizada e padronizada no desenvolvimento, facilitando tanto a manutenção quanto a escalabilidade do sistema.

Com uma arquitetura baseada em tecnologias amplamente adotadas e uma abordagem orientada a objetos, o software promete oferecer uma experiência de venda de ingressos intuitiva e fluida tanto para os usuários finais quanto para os administradores do sistema, contribuindo para o sucesso e a eficiência na gestão de eventos.

## 2 Requisitos Funcionais / Não Funcionais

### Requisitos Funcionais

#### 1. Criação de Evento (HU001)

- O sistema deve permitir que um organizador de eventos crie um evento.
- O sistema deve solicitar informações básicas do evento, como nome, data, local e descrição.

#### 2. Definição de Quantidade de Ingressos (HU002)

- O sistema deve permitir que o organizador de eventos defina a quantidade de ingressos disponíveis para o evento.
- O sistema deve verificar e assegurar que a quantidade de ingressos não exceda a capacidade máxima definida pelo organizador.

#### 3. Definição de Preço de Ingressos (HU003)

- O sistema deve permitir que o organizador de eventos defina o preço dos ingressos.
- O sistema deve suportar diferentes tipos de ingressos com preços variados.

#### 4. Busca de Eventos (HU004)

- O sistema deve permitir que compradores busquem eventos disponíveis para compra.
- O sistema deve oferecer filtros de busca por data, local, tipo de evento, etc.

## **5. Compra de Ingressos (HU005)**

- O sistema deve permitir que compradores comprem ingressos para eventos.
- O sistema deve processar pagamentos de forma segura.
- O sistema deve diminuir a quantidade de ingressos disponíveis após a compra.

## **6. Visualização de Compras (HU006)**

- O sistema deve permitir que compradores visualizem os ingressos que compraram.
- O sistema deve fornecer detalhes do evento e dos ingressos comprados.

## **7. Visualização de Detalhes do Evento (HU007)**

- O sistema deve permitir que compradores visualizem detalhes do evento, como data, local, descrição e ingressos disponíveis.
- O sistema deve exibir avaliações e comentários sobre o evento, se houver.

## **8. Cancelamento de Evento (HU008)**

- O sistema deve permitir que organizadores de eventos cancelem um evento.
- O sistema deve notificar os compradores sobre o cancelamento e gerenciar o reembolso dos ingressos comprados, se necessário.

## **Requisitos Não Funcionais**

### **1. Desempenho**

- O sistema deve ser capaz de atender a múltiplos usuários simultaneamente sem degradação de desempenho.
- Consultas ao banco de dados devem ser otimizadas para tempos de resposta rápidos.

### **2. Escalabilidade**

- O sistema deve ser escalável para suportar um aumento no número de usuários e eventos sem perda de desempenho.

### **3. Segurança**

- O sistema deve garantir a segurança dos dados dos usuários.
- O sistema deve usar HTTPS para todas as comunicações entre cliente e servidor.

### **4. Usabilidade**

- A interface do usuário deve ser intuitiva e fácil de usar, tanto para organizadores quanto para compradores.
- O sistema deve ser acessível e compatível com diferentes dispositivos e navegadores.

## 5. Confiabilidade

- O sistema deve ter alta disponibilidade.

## 6. Manutenibilidade

- O código do sistema deve ser bem documentado e estruturado para facilitar a manutenção e futuras atualizações.
- O sistema deve seguir boas práticas de desenvolvimento e padrões de projetos.

## 7. Compatibilidade

- O sistema deve suportar integração com APIs externas para processamento de pagamentos e outras funcionalidades adicionais

# 3 Casos de Uso/ Historia de Usuários

História de Usuários é um recurso usado no desenvolvimento de software para capturar requisitos e funcionalidades do sistema de uma maneira mais compreensível para clientes, analistas e projetistas.

Neste projeto, foi optado pelo padrão COMO, QUERO, PARA na criação das histórias de usuários. Além disso, cada história possui um identificador (Id) para facilitar a navegação pelo projeto.

Id	HU001
COMO	usuário (organizador de eventos)
QUERO	poder criar um evento no sistema
PARA	anunciar os ingressos a venda

Id	HU002
COMO	usuário (organizador de eventos)
QUERO	poder definir a quantidade ingressos disponíveis no meu evento
PARA	controlar a capacidade do evento

Id	HU003
COMO	usuário (organizador de eventos)

QUERO	poder definir o preço dos ingressos do meu evento
PARA	para estabelecer um valor de venda

<b>Id</b>	<b>HU004</b>
COMO	usuário (comprador)
QUERO	poder buscar eventos disponíveis para compra
PARA	encontrar eventos de interesse

<b>Id</b>	<b>HU005</b>
COMO	usuário (comprador)
QUERO	poder comprar ingressos para um evento
PARA	para garantir minha participação

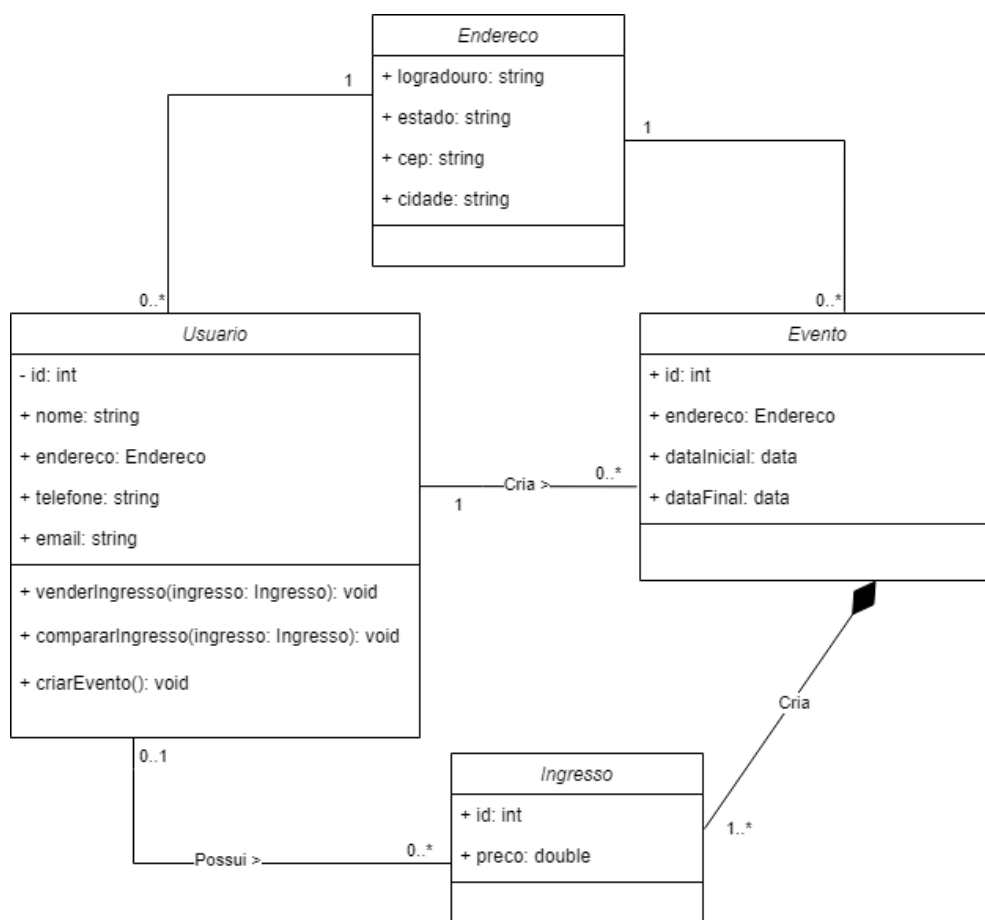
<b>Id</b>	<b>HU006</b>
COMO	usuário (comprador)
QUERO	poder visualizar os ingressos que comprei
PARA	para acompanhar minhas compras

<b>Id</b>	<b>HU007</b>
COMO	usuário (comprador)
QUERO	poder visualizar detalhes de um evento
PARA	definir se vou ou não comprar um ingresso para esse evento



<b>Id</b>	<b>HU008</b>
COMO	usuário (organizador de eventos)
QUERO	poder cancelar um evento
PARA	para retirá-lo do sistema e reembolsar os compradores, se necessário

## 4 Diagrama de classes



### Detalhamento das classes:

**Usuário:** Classe criada para representar o usuário do sistema. Possui como atributos todas aquelas características que um usuário do sistema deve ter (nome, endereço, telefone, email, etc...). Além disso, possui os métodos para comprar e vender ingressos e criar eventos.

**Evento:** Classe criada para representar o evento criado pelo usuário. Contém os atributos de um evento (endereço, dataInicial, dataFinal), incluindo uma lista com os ingressos disponíveis.

**Ingresso:** Classe criada para representar o ingresso. Contém como atributos o preço e o evento a qual está associado.

**Endereço:** Classe criada para representar os endereços, tanto dos usuários quanto dos eventos. Contém os atributos esperados para essa classe (logradouro, estado, cep, cidade).

## 5 Testes

Casos de teste (CTs) são descrições detalhadas para verificar se funcionalidades específicas de um software funcionam corretamente, garantindo qualidade e identificando problemas. Eles são cruciais para a confiabilidade do software e a satisfação do cliente.

Neste projeto, os testes irão representar os casos de uso descritos no tópico 2. Serão detalhados com as PRÉ-CONDIÇÕES, PASSOS e RESULTADO ESPERADO.

Cada caso de teste terá um identificador (Id) e uma pequena descrição, para facilitar a navegação, além da HU da qual ele representa.

Id	CT001 - Criar um evento (HU001)
PRÉ-CONDIÇÕES	O usuário (organizador) está acessando o sistema.
PASSOS	1) Navegar para a página de criação de eventos. 2) Preencher os campos obrigatórios. 3) Clicar no botão "Criar Evento".
RES. ESPERADO	Evento é criado com sucesso e uma mensagem de confirmação é exibida.

Id	CT002 - Criar um evento com dados inválidos (HU001)
PRÉ-CONDIÇÕES	O usuário (organizador) está acessando o sistema.
PASSOS	1) Navegar para a página de criação de eventos. 2) Deixar algum campo obrigatório em branco. 3) Clicar no botão "Criar Evento".
RES. ESPERADO	O sistema exibe uma mensagem de erro indicando que todos os campos obrigatórios devem ser preenchidos.

<b>Id</b>	<b>CT003 - Definição de Qtd. de ingressos válida (HU002)</b>
PRÉ-CONDIÇÕES	O usuário (organizador) está acessando o sistema e tem um evento criado.
PASSOS	1) Navegar para a página de edição do evento. 2) Inserir a quantidade de ingressos disponíveis. 3) Clicar no botão "Salvar".
RES. ESPERADO	Quantidade de ingressos é salva com sucesso e uma mensagem de confirmação é exibida.

<b>Id</b>	<b>CT004 - Definição de Qtd. de ingressos inválida (HU002)</b>
PRÉ-CONDIÇÕES	O usuário (organizador) está acessando o sistema e tem um evento criado.
PASSOS	1) Navegar para a página de edição do evento. 2) Inserir um valor negativo na quantidade de ingressos. 3) Clicar no botão "Salvar".
RES. ESPERADO	O sistema exibe uma mensagem de erro indicando que a quantidade de ingressos deve ser um valor positivo.

<b>Id</b>	<b>CT005 - Definição de preço de ingressos válido (HU003)</b>
PRÉ-CONDIÇÕES	O usuário (organizador) está acessando o sistema e tem um evento criado.
PASSOS	1) Navegar para a página de edição do evento. 2) Inserir o preço dos ingressos. 3) Clicar no botão "Salvar".
RES. ESPERADO	Preço dos ingressos é salvo com sucesso e uma mensagem de confirmação é exibida.

<b>Id</b>	<b>CT006 - Definição de preço de ingressos inválido (HU003)</b>
PRÉ-CONDIÇÕES	O usuário (organizador) está acessando o sistema e tem um

	evento criado.
PASSOS	1) Navegar para a página de edição do evento. 2) Inserir um valor negativo no preço dos ingressos. 3) Clicar no botão "Salvar".
RES. ESPERADO	O sistema exibe uma mensagem de erro indicando que o preço dos ingressos deve ser um valor positivo.

<b>Id</b>	<b>CT007 - Busca de eventos com filtros válidos (HU004)</b>
PRÉ-CONDIÇÕES	O usuário (comprador) está acessando o sistema.
PASSOS	1) Navegar para a página de busca de eventos. 2) Inserir critérios de busca (ex.: data, localização, tipo). 3) Clicar no botão "Buscar".
RES. ESPERADO	O sistema exibe uma lista de eventos que correspondem aos critérios de busca.

<b>Id</b>	<b>CT008 - Busca de eventos sem critérios de busca (HU004)</b>
PRÉ-CONDIÇÕES	O usuário (comprador) está acessando o sistema.
PASSOS	1) Navegar para a página de busca de eventos. 2) Não inserir nenhum critério de busca. 3) Clicar no botão "Buscar".
RES. ESPERADO	O sistema exibe uma lista de todos os eventos disponíveis.

<b>Id</b>	<b>CT009 - Compra de ingresso com dados válidos (HU005)</b>
PRÉ-CONDIÇÕES	O usuário (comprador) está acessando o sistema e há ingressos disponíveis para o evento.
PASSOS	1) Navegar para a página do evento. 2) Selecionar a quantidade de ingressos desejada. 3) Clicar no botão "Comprar". 4) Inserir informações de pagamento. 5) Confirmar a compra.
RES. ESPERADO	Compra é realizada com sucesso.

<b>Id</b>	<b>CT010 - Compra de ingresso sem ingresso disp. (HU005)</b>
PRÉ-CONDIÇÕES	O usuário (comprador) está acessando o sistema e não há ingressos disponíveis para o evento.
PASSOS	1) Navegar para a página do evento. 2) Selecionar a quantidade de ingressos desejada. 3) Clicar no botão "Comprar".
RES. ESPERADO	O sistema exibe uma mensagem indicando que não há ingressos disponíveis para o evento.

<b>Id</b>	<b>CT018 - Visualização dos ingressos comprados sem nenhum ingresso (HU006)</b>
PRÉ-CONDIÇÕES	O usuário (comprador) está acessando o sistema e não tem ingressos comprados.
PASSOS	1) Navegar para a página de ingressos comprados.
RES. ESPERADO	O sistema exibe uma mensagem indicando que não há ingressos comprados.

<b>Id</b>	<b>CT019 - Visualização de detalhes de um evento existente (HU007)</b>
PRÉ-CONDIÇÕES	O usuário (comprador) está acessando o sistema e há eventos disponíveis.
PASSOS	1) Navegar para a página de um evento. 2) Clicar no evento desejado.
RES. ESPERADO	O sistema exibe os detalhes do evento selecionado.

<b>Id</b>	<b>CT020 - Visualização de detalhes de um evento inexistente (HU007)</b>
PRÉ-CONDIÇÕES	O usuário (comprador) está acessando o sistema.

PASSOS	1) Navegar para a página de um evento inexistente (ex.: evento excluído ou ID inválido).
RES. ESPERADO	O sistema exibe uma mensagem indicando que o evento não foi encontrado.

<b>Id</b>	<b>CT021 - Cancelamento de evento com sucesso (HU008)</b>
PRÉ-CONDIÇÕES	O usuário (organizador) está acessando o sistema e tem um evento criado.
PASSOS	1) Navegar para a página de gerenciamento do evento. 2) Selecionar o evento a ser cancelado. 3) Clicar no botão "Cancelar Evento". 4) Confirmar o cancelamento.
RES. ESPERADO	Evento é cancelado com sucesso, removido do sistema, e os compradores são notificados.

<b>Id</b>	<b>CT022 - Tentativa de cancelamento de evento inexistente (HU008)</b>
PRÉ-CONDIÇÕES	O usuário (organizador de eventos) está acessando o sistema.
PASSOS	1) Navegar para a página de gerenciamento de eventos. 2) Tentar cancelar um evento que não existe.
RES. ESPERADO	O sistema exibe uma mensagem indicando que o evento não foi encontrado.

## 6 Estratégia de controle de versão

Para o gerenciamento de versões do nosso projeto, adotamos uma estratégia simples, mas eficaz, utilizando Git e a plataforma Github como sistema de controle de versão. Aqui estão os principais pontos da nossa abordagem:

### 6.1. Branch Principal (master)

Temos uma branch principal chamada *master* que contém a versão estável e pronta para produção do nosso código. Todas as funcionalidades são eventualmente mescladas nesta branch após passarem por revisões e testes adequados.

## 6.2. Branches de Desenvolvimento

Embora nossa intenção inicial fosse utilizar *branches* separadas para novas funcionalidades (*feature*) e correções de bugs (*hotfix*), todas as modificações acabaram sendo realizadas diretamente na master devido à ausência de bugs, simplicidade do projeto e equipe reduzida.

Nossa estratégia reflete a simplicidade e o foco na entrega rápida de uma solução funcional, evitando complicações adicionais no fluxo de trabalho.

## 7 Repositórios e Recursos

Nome do Repositório: [sistema-eventos](#)

Descrição: Repositório utilizado para o desenvolvimento do sistema de eventos. Contém o código-fonte e a documentação relacionados ao projeto.

Data de Acesso: 23 de julho de 2024, às 10:11

Diagrama de Classes: [Diagrama de Classes](#)

Descrição: Diagrama que representa a estrutura e os relacionamentos das classes no sistema de eventos.