Relatório do PBL: Problema 3
EXA-863 - MI Programação

João Marcelo N. Fernandes[1]

[1]Universidade Estadual de Feira de Santana (UEFS) - Feira de Santana, BA - Brasil

joaomarcelo642@gmail.com

Abstract. *This paper presents the development of a system for event management and ticket purchasing using Java and JavaFX. The system aims to simplify event organization, user and administrator registration, as well as ticket sales and feedback management. It includes functionalities such as event creation, ticket purchase with different payment methods, and event evaluation. The development stages, key technical decisions, and obtained results are detailed in the following sections.*

Resumo. *Este artigo apresenta o desenvolvimento de um sistema para gerenciamento de eventos e compra de ingressos utilizando a linguagem Java e a biblioteca JavaFX. O objetivo é facilitar a organização de eventos, o cadastro de usuários e administradores, além de gerenciar vendas de ingressos e feedbacks. O sistema implementa funcionalidades como cadastro de eventos, compra de ingressos com diferentes formas de pagamento e avaliação de eventos. As etapas de desenvolvimento, as principais decisões técnicas e os resultados obtidos são detalhados nas seções seguintes.*

1. Introdução

Na disciplina de MI - Programação do semestre 2024.2, foi proposto a implementação de melhorias relativas ao código desenvolvido anteriormente. Um problema no qual era solicitada um sistema eficiente de venda de ingressos. O presente trabalho descreve o desenvolvimento de uma interface gráfica para o programa desenvolvido anteriormente, utilizando a ferramenta JavaFX.

O objetivo do projeto é criar um sistema integrado para organizar eventos, permitindo que administradores cadastrem eventos e que usuários comprem ingressos de forma prática. Tendo em vista o desenvolvimento de uma GUI (Graphical User Interface) afim de conectar organizadores e participantes de eventos, otimizando processos como controle de ingressos e feedbacks.

A solução foi implementada em Java, utilizando JavaFX para a interface gráfica, garantindo portabilidade e flexibilidade no desenvolvimento. O sistema oferece recursos como:

- Cadastro de eventos por administradores.
- Compra de ingressos com opções de pagamento via cartão de crédito ou boleto.
- Feedbacks de eventos para melhoria contínua.

Na seção de fundamentação teórica, serão explicitados os conceitos usados como base para resolução do problema. Na seção de metodologia, serão descritos os processos

no desenvolvimento do software, como as escolhas de abordagem, as definições dos requisitos e funcionalidades e a ordem de codificação, bem como as tecnologias utilizadas. Na seção resultados e discussões serão apresentados os resultados apresentados pelo desenvolvimento, bem como testes e instruções de utilização. Na seção conclusão serão informadas brevemente as conclusões e os objetivos cumpridos, e na seção referências, serão listadas as referências utilizadas para a elaboração da resolução.

2. Fundamentação Teórica

Nessa seção, seram apresentados e fundamentados os conceitos que serviram como base para o desenvolvimento do problema. A implementação deste sistema está ancorada principalmente na utilização da ferramenta JavaFX, para a implementação da interface gráfica, além da continuidade em relação à POO.

2.1. Programação Orientada a Objetos (POO)

A POO [Deitel and Deitel 2017] é uma abordagem amplamente utilizada em engenharia de software que organiza o código em torno de ”objetos”, que representam entidades do mundo real. O sistema anterior foi desenvolvido com o intuito de ser um sistema de venda de ingressos completo, dando diversas possibilidades a quem estiver usando, como cadastrar eventos, usuario além de realizar operações de compra e publicações de feedbacks. A versão atual mantém essa estrutura, mas a expande ao integrar todo o funcionamento do programa a uma interface gráfica, desse modo, tornando mais acessível.

2.2. JavaFX

Para a implementação de uma interface gráfica, foi utilizada a ferramenta JavaFX [Oracle 2024], que oferece a possiblidade de contruir essa interface através de arquivos *fxml*, que representam as telas e que são devidamente programadas e integradas ao sistema atraves de classes controladoras.

3. Metodologia

Nesta seção serão apresentados os requisitos do sistema, os processos de desenvolvimento, o fluxo básico do programa e as ferramentas utilizadas para a implementação.

3.1. Definição de requisitos

Os principais requisitos do sistema incluem:

- Telas de Login e Cadastro
- Tela de Listagem de Eventos.
- Tela de Compra de Ingresso.
- Confirmação e Resumo de Compra.
- Opção Para Avaliar Evento.

3.2. Processo de Desenvolvimento

O desenvolvimento seguiu as seguintes etapas:

- Planejamento inicial: Definição dos requisitos e funcionalidades com base na análise do problema.
- Modelagem do sistema: Implementação das classes principais (Evento, Usuario, Ingresso, Controller) para gerenciar dados e lógica.
- Interface gráfica: Desenvolvimento das telas com JavaFX para interação do usuário.

3.2.1. Alterações no Programa Anterior

Foram feitas alterações e adições necessárias no programa, como a criação de métodos afim de melhorar e facilitar a integração com o front-end, além da criação de classes *Adapter*, afim de melhorar a serialização dos dados e objetos gerados.

3.3. Fluxograma

Abaixo, o fluxo básico do sistema:

- Login/Cadastro do usuário.
- Usuário acessa a lista de eventos.
- Seleção de um evento para exibir detalhes.
- Compra de ingresso ou envio de feedback.

3.4. Ambiente de Desenvolvimento

O sistema operacional utilizado para o desenvolvimento foi o *Windows 10*, utilizando a linguagem *JAVA 21.0.4* juntamento com a biblioteca *JavaFX 21*, por meio do IDE (ambiente de desenvolvimento integrado) *IntelliJ IDEA 2024.2.4 (Ultimate Edition)*.

3.5. Estrutura do Programa

O programa foi desenvolvido seguindo um padrão organizacional [Gamma et al. 1994], dividindo as classes de back-end, os controladores do front-end, os arquivos das telas e a aplicação pela *main*.

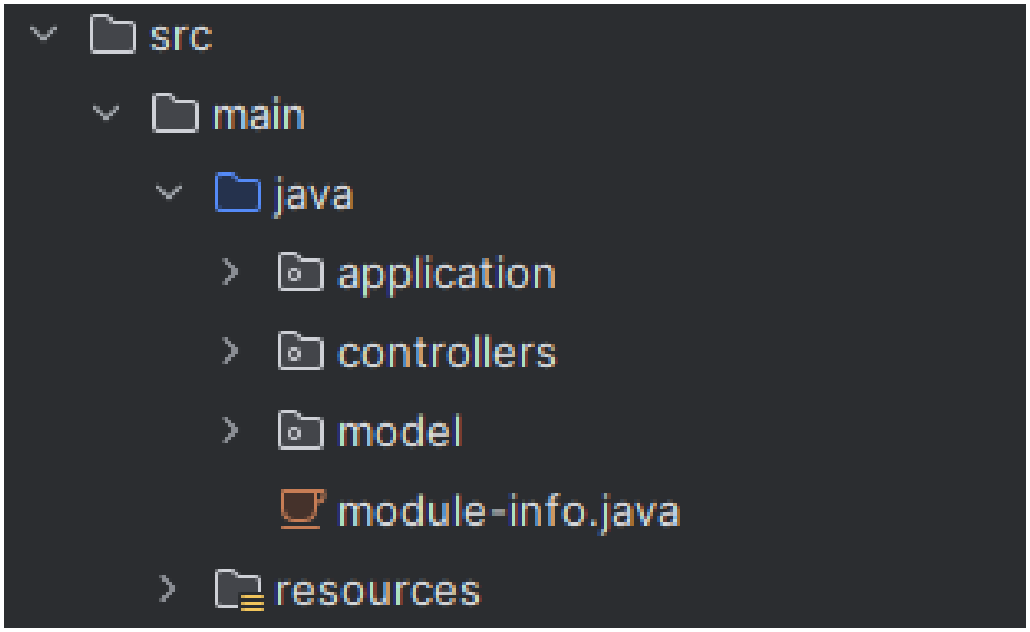

Figure 1. Estrutura do programa

4. Resultados e Discussões

4.1. Solução desenvolvida

O sistema foi implementado de forma modular, tendo uma classe *SceneManager* responsável pela troca de telas, com cada funcionalidade principal associada a uma classe controladora:

- **LoginController**: Permite que usuários e administradores acessem o sistema.
- **CadastroController**: Permite o registro de novos usuários e administradores.
- **AdminHomeController**: Permite que administradores registrem novos eventos.
- **ListagemEventosController**: Lista os eventos disponíveis para o usuário.
- **CadastrarCartaoController**: Permite o usuário cadastrar um cartão de crédito.
- **EditarUsuarioController**: Permie o usuário editar os dados da conta.
- **IngressosCompradosController**: Permite o usuário visualizar os seus ingressos comprados.
- **DetalhesEventoController**: Exibe informações detalhadas de eventos, como descrição, data e assentos disponíveis.
- **CompraIngressoController**: Gerencia a compra de ingressos, com validação de pagamento.
- **EnviarFeedbackController**: Permite o usuário enviar um feedback sobre o evento.

Cada uma dessas classes controladoras se associa com um arquivo *fxml*, que é responsável pelo design da tela.

4.2. Dados de Entrada e Saída

- Entrada: Informações fornecidas pelo usuário, como dados de login, descrição de eventos, assentos selecionados e forma de pagamento.
- Saída: Mensagens de sucesso/erro, confirmação de compra e boletos gerados.

4.3. Testes e resultados

Os testes realizados incluíram:

- Cadastro de usuários e administradores.
- Criação de eventos com múltiplos assentos.
- Cadastro de cartão.
- Simulação de compra de ingressos via cartão e boleto.
- Publicação de feedback sobre eventos.

4.3.1. Resultados Obtidos

- O sistema funcionou corretamente na maioria dos cenários testados.
- O processo de compra foi validado com diferentes formas de pagamento.
- Os feedbacks registrados foram exibidos corretamente para os eventos associados.

4.3.2. Limitações Identificadas

- Não foi implementada a tradução de todas as telas do sistema, apenas da tela de login.
- Não foi implementada validação completa de CPF e cartões.
- Não há suporte para cancelamento de compras.

4.4. Manual de Uso Para Administrador

1. Faça login ou cadastre-se como administrador.
2. Preencha os campos com os dados requeridos do evento.
3. Os assentos devem ser adicionados separados por vírgula (ex: A1,B1,C1).
4. Clique no botão para cadastrar o evento.

4.5. Manual de Uso Para Usuário

1. Faça login ou cadastre-se como usuário.
2. Acesse a lista de eventos disponíveis.
3. Cadastre um cartão de crédito (opcional).
4. Selecione um evento para visualizar detalhes.
5. Escolha um assento e prossiga com a compra.
6. Selecione a forma de pagamento e confirme a compra (para selecionar o cartão, é necessário já ter cadastrado anteriormente).

4.6. Cadastros Feitos

4.6.1. Usuários Cadastrados

- Login: joao, Senha: 123, Nome: Joao Marcelo, CPF: 1234567890, Email: joao@, usuário comum.
- Login: Gabriel, Senha: 123, Nome: Gabriel Barbosa, CPF: 7894561230, Email: gabi@.com, usuário comum.
- Login: admin, Senha: 1234, Nome: Administrador, CPF: 987654321, Email: adm@.com, Administrador.

4.6.2. Eventos Cadastrados

- Nome: Gladiador II, Descricao: Continuação do épico Gladiador de Ridley Scott, Data: 30/01/2025.
- Nome: Festival de Verão , Descricao: O maior festival da Bahia, Data: 27/01/2025.
- Nome: Batalha da Aldeia , Descricao: A maior batalha de rap do Brasil, Data: 01/06/2025

5. Conclusão

O sistema desenvolvido atingiu os principais objetivos propostos, proporcionando uma solução funcional para o gerenciamento de eventos e compra de ingressos, contudo a tradução automática das telas não foi implementada em sua totalidade, apenas na 1° tela, sendo esse o unico requisito não alcançado. Nesse projeto foi possível ampliar conhecimentos sobre o desenvolvimento e utilização de GUI, além da integração entre *front-end* e *back-end*. No entanto, há espaço para melhorias, como:

- Tradução automática totalmente implementada.
- Integração com sistemas de pagamento online.
- Implementação de validação de dados mais robusta.
- Adição de funcionalidades como cancelamento de ingressos.

Essas melhorias podem tornar o sistema mais completo e alinhado às necessidades dos usuários.

References

Deitel, P. J. and Deitel, H. M. (2017). *Java: How to Program*. Pearson, 11ª edição edition.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Oracle (2024). Javafx documentation.