MLEX Repositório Pessoal de Jogos

Bernardo Hummes Flores
Henrique Chaves Pacheco
Maria Cecília Matos Correa

A visão sistêmica para o Repositório Pessoal de Jogos MLEX sumariza um conjunto de entidades baseadas em estruturas de classes e interações entre classes do paradigma de Programação Orientada a Objetos.

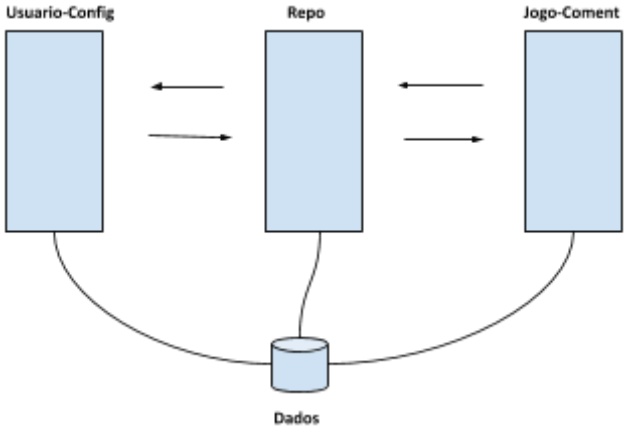
A arquitetura da solução em software para o MLEX é realizada pela troca de mensagens entre três camadas de sistema: UsuarioCommand-PlataformaConfiguracao, Repositorio e Jogo-Comentario.

Cada camada da arquitetura define um conjunto de classes que interage com uma classe abstrata FileHandler, devido a necessidade de manipulação de arquivos, a fim de garantir persistência de dados relativos às configurações do usuário, categorias de jogos e dados dos jogos, além da seção de comentários de cada jogo.

- UsuarioCommand-PlataformaConfiguracao: responsável pelas interações com o usuário e manipulação de arquivos de configuração do usuário.

- Repositorio: responsável pela organização de jogos em categorias, arquivamento de categorias e manipulação de listas.

- Jogo-Comentário: responsável pelos dados integrais de cada jogo, arquivamento de dados dos jogos e respectivos comentários.



Funcionalidades e soluções do sistema MLEX

1) Adicionar jogos ao repositório

A adição de jogos ao repositório é realizada pela troca de mensagens entre UsuarioCommand, Repositorio e Jogo.
Para que um usuário consiga adicionar um jogo ao Repositório é necessário que a classe Repositório possua/crie um identificador do jogo. O jogo será, então, adicionado ao arquivo de categorias, manipulado pela classe repositório. Finalmente, o jogo será construído na classe Jogo e armazenado em arquivo.

2) Remover jogos do repositório

A remoção de jogos do repositório é realizada pela troca de mensagens entre UsuarioCommand, Repositorio e Jogo.
Para tanto, o Repositório solicita a verificação de senha à UsuarioCommand. Após verificada a senha e encontrado o arquivo do repositório contendo os dados do jogo, os dados do jogo é removido do arquivamento do repositório. Finalmente, o arquivo do jogo é buscado na classe Jogo, deletado e o objeto correspondente é destruído do sistema.

3) Atualizar informações do jogo pelo usuário

A atualização de informações do jogo pelo usuário é realizada pela troca de mensagens entre UsuarioCommand, Repositorio e Jogo.
Para tanto, é necessário que o usuário forneça as novas informações do jogo para a classe UsuarioCommand. Então, a classe Repositório recebe dados para atualização, altera dados nos seus arquivos e provoca uma atualização de versão nos dados do jogo correspondente na classe Jogo.

4) Adicionar comentários sobre o jogo

A adição de comentário sobre o jogo é realizada pela troca de mensagens entre UsuarioCommand, Repositorio, Jogo e Comentário.
Para tanto, é necessário que o usuário forneça um comentário, do tipo string, e uma nota, do tipo inteiro, para a classe UsuarioCommand. Os dados são repassados para Repositório e Jogo, a classe Jogo aciona o construtor de Comentário e atualiza o arquivo de comentário do Jogo correspondente.

5) Filtrar jogos por características

A filtragem de jogos por características é realizada pela troca de mensagens entre UsuarioCommand e PlataformaConfiguração.
Para tanto, o usuário define quais serão os critérios de exibição para a classe PlataformaConfiguração. A classe PlataformaConfiguração atualiza o seu critério de exibição em arquivos locais e aciona a exibição do conteúdo do repositório pela classe UsuarioCommand.

6) Criar categorias
A criação de categorias é realizada pela troca de mensagens entre UsuarioCommand e Repositorio.
Para tanto, a partir do nome da nova categoria, fornecido pelo usuário, a classe UsuarioCommand aciona a criação da categoria na classe Repositório. Além disso, a classe repositório as informações necessários em seus arquivos locais.

7) Verificação de integridade
A verificação de integridade é realizada pela troca de mensagens entre UsuarioCommand, Repositorio e Jogo.
Para tanto, o usuário envia a requisição de verificação de integridade para a classe UsuarioCommand e o identificador do jogo é obtido na classe Repositorio.
Assim, o método de verificar integridade é acionado na classe Jogo e seus arquivos são atualizados.

8) Alterar configurações do funcionamento
A alteração de configurações de funcionamento é realizada pelo acionamento dos métodos correspondentes na classe PlataformaConfiguracao.
Para tanto, o usuário informa os dados da sua conta e senha, bem como as opções a serem alteradas no arquivo de configuração na classe PlataformaConfiguracao.

9) Recomendação de jogo
A recomendação de jogos é realizada pela troca de mensagem das classes UsuarioCommand, Repositorio, Jogo e Comentario.
Para tanto, o usuário aciona o método de recomendação de jogo em UsuarioCommand, fornecendo também o email, para o qual serão enviados dados sobre determinado jogo. As informações sobre o jogo são adquiridas em uma lista, fornecida pela classe Jogo. Além dos dados do jogo, o repositório também inclui na recomendação os comentários do jogo, salvos pelo usuário préviamente.

10) Segurança dos arquivos de configuração do usuário
A segurança dos arquivos de configuração do usuário é garantida através do uso de criptografia no armazenamento dos arquivos de configuração usuário.

Tecnologias e Método de desenvolvimento

O sistema utilizará bibliotecas padrão da linguagem Java para prover funções de manipulação de arquivos (java.io) e outras facilidades.
Além disso, o processo de desenvolvimento do projeto será baseado no modelo em cascata, no qual cada etapa de desenvolvimento do projeto ocorre sequencialmente.
Durante a etapa de implementação, o grupo utilizará as ferramentas GitHub e Eclipse.