DOCUMENTAÇÃO – SISTEMA PARA BIBLIOTECA

Programação e Desenvolvimento de Software II – UFMG 2019.1

Equipe 6: Breno Pimenta, Eduardo Paraíso, Gustavo Santos

- **INTRODUÇÃO**

  O uso de softwares de gerencialmente vem sendo empregados em empresas e entidades, isso devido ao fato de haver uma grande quantidade de informação a serem guardadas e/ou manipuladas e a dificuldade de se manter esse trabalho manualmente. Um software de gerencialmente entra como uma ferramenta capaz de realizar este trabalho de forma ágil e precisa.

  O software desenvolvido trata-se de um Sistema Gerencial para Bibliotecas. O sistema foi concebido inicialmente como um self checkout, onde o usuário interessado realiza o login para efetuar buscas, empréstimos e devoluções.

  Um sistema de self checkout não só desburocratiza a biblioteca, como também reduz custos ao proprietário. Uma vez que não é necessário um agente superior, o bibliotecário, para realização das mesmas atividades.

- **IMPLEMENTAÇÃO**
  - **Estrutura de Dados**

    Os dados foram segmentados em três partes:

    - Autenticação, onde se encontram usuários e senhas
    - Pessoas, onde se encontram os dados de cada usuário juntamente com os empréstimos arquivos. Cada pessoa possui um arquivo independente com suas informações.
    - Livros, onde se encontram em arquivo único, todos o acervo da biblioteca.

  - **Decisões Tomadas**

    De todas as decisões realizadas para este projeto, a principal e que causou maior dificuldade, foi em relação a criação dos bancos de dados. Para um gerenciador de biblioteca funcionar, são necessários ao menos três bancos de dados distintos (usuário, autenticação de acesso e livros), como nenhum componente da equipe ainda cursou a disciplina Introdução a Banco de Dados (DCC011), o monitor responsável autorizou a utilizarmos arquivos de textos.

    Foi decidido segmentar ao máximo o código, criando assim diversas bibliotecas especializadas. Por exemplo, como medida de segurança, foi decido criar uma

biblioteca dedicada ao cadastro de informações do usuário e outra biblioteca dedicada ao cadastro da senha de acesso.

O projeto foi dividido em três frentes de desenvolvimento, cada frente encabeçada por um integrante da equipe. Foram elas:
"Gestão e Menu" – Breno Pimenta
"Cadastro e Login" – Eduardo Paraíso
"Busca e Exceções" – Gustavo Santos

Inicialmente tinha-se a ideia de criar um sistema altamente abstrato, porém devido ao prazo e a grande dificuldade de um desenvolvimento tão complexo neste nível, foi decido por eliminar esta ideia, porém a classe Pessoa foi mantida abstrata, o que garante uma escalabilidade do sistema para modificações futuras.

Foi decidido a padronização na nomenclatura de métodos, variáveis e classes, sendo:
- Métodos em lowerCamelCase.
- Variáveis em snake_case.
- Classes em UpperCamelCase.

Métodos que normalmente poderiam ser criados como void, foram criados como booleanos para que fosse atendido requisitos para teste.

Foi decidido como prioridade o uso de try e catch apenas na main, enquanto o lançamento de suas respectivas exceções foram feitas dentro dos métodos.

- **Escalabilidade**
  Foi criada a classe pessoa de maneira abstrata, pois como concebido inicialmente a arquitetura teria três subclasses da mesma. Porém, para que fosse atendido o prazo de entrega, optou-se em implementar apenas uma das classes que herdaria de pessoa, no entanto, a classe pessoa foi mantida abstrata para que facilitasse futuras implementações e melhorias no sistema.

  OBS.: A classe Pessoa e Usuário atendem ao Princípio da Substituição de Liskov, ou seja, não há perda de funcionalidade quando se substitui Usuário em Pessoa.

- **Principais Métodos**
  acesso( )
  Continuação do método de autenticação, caso todos os dados inseridos sejam autenticados, este método imprime o texto "Acesso autorizado" e modifica uma flag booleana para true. Caso contrário, imprime o texto "Acesso negado" e mantem a flag booleana em false.

autenticação( )

Este método trabalha em duas etapas, em um primeiro momento imprime na tela o comando para inserção do identificador do usuário (Cadastro de Pessoa Física), a informação inserida é comparada com as informações presentes no arquivo .txt de autenticação. Em um segundo momento, caso exista o identificador no arquivo, é enviado o comando para que seja inserida a senha de acesso, a informação inserida é comparada com a linha seguinte ao identificador no arquivo.

buscarLivro(string)

Permite ao usuário realizar uma busca dentro da biblioteca através do nome dos livros.

cadastrarPessoa( )

Método criado para auxiliar o auto cadastro do usuário, imprime comandos ao utilizador, onde as informações inseridas são capturadas por métodos de set e posteriormente salvas em um arquivo .txt dedicado nomeado com o Cadastro de Pessoa Física do usuário.

cadastraSenha( )

Assim como o método de cadastramento do usuário, este método imprime comandos na tela e os dados inseridos são capturados por dois métodos de set, que são comparados por um submétodo (descrito abaixo), onde em caso positivo de comparação, a senha inserida é enviada a um arquivo .txt, único, contendo login e senha de todos os usuários cadastrados, denominado autenticação.

comparaSenha( )

Submétodo do método cadastraSenha( ), tem a finalidade de comparar a senha e a contraSenha inserida pelo utilizador durante o cadastramento.

devolverLivro( )

Encerra o empréstimo do usuário, devolvendo um de seus livros.

printBusca( )

Retorna na tela, todos os resultados encontrados no acervo de livros cadastrados e que coincidem com o termo utilizado no método de busca.

- **Sets e Gets**

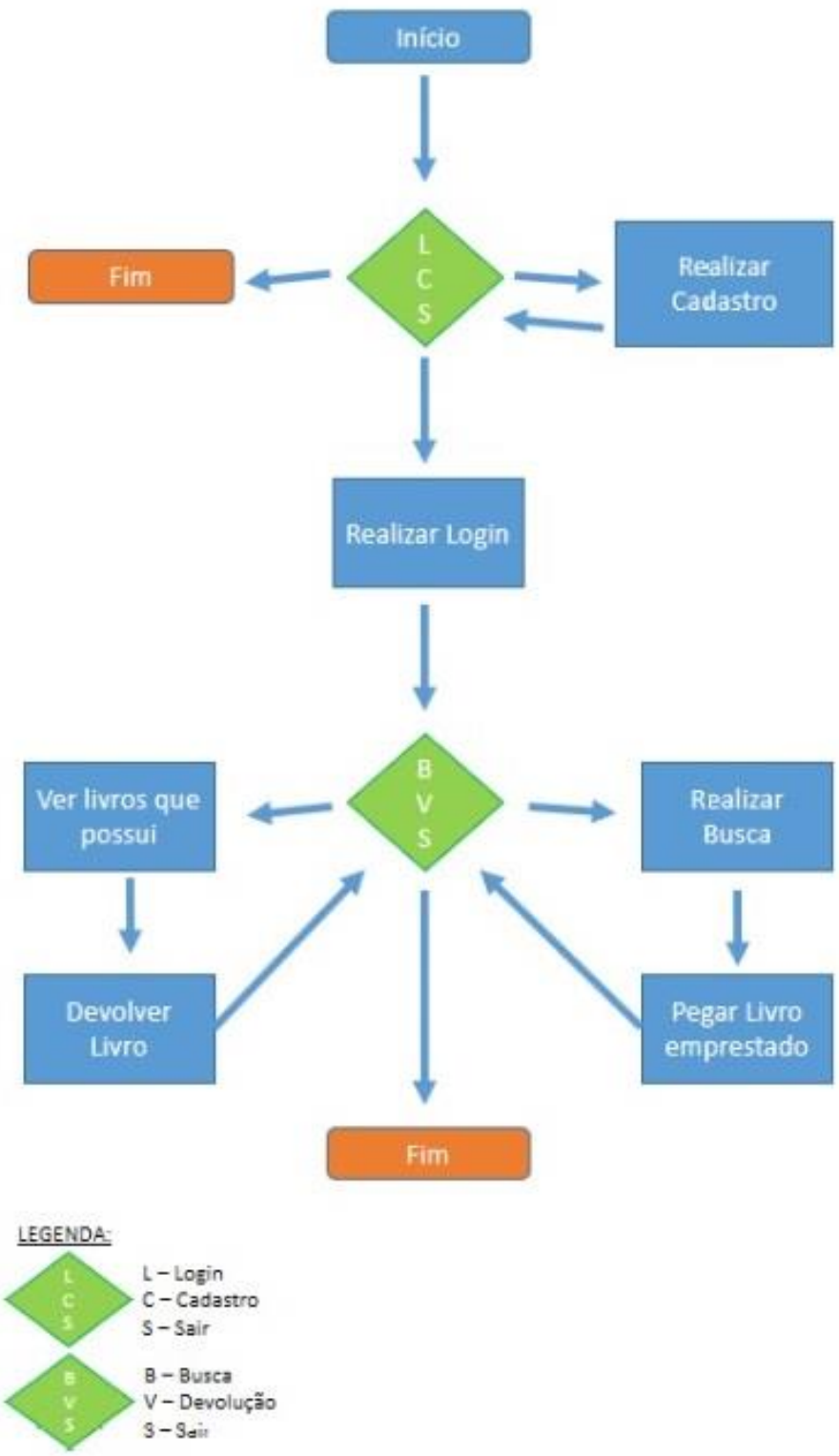
  setNome( ), setCPF( ), setEndereco( ), setContato( ), ...

  Os submétodos sets capturam as informações inseridas pelo usuário, salvando-as em arquivos específicos a depender da informação, para consultas futuras por outros métodos.

  getNome( ), getCPF( ), getEndereco( ), getContato( ), ...

Os submétodos gets leem as informações cadastradas pelo usuário, a leitura ocorre em arquivos específicos a depender da informação.

- **FLUXO DE EXECUÇÃO**



- **TESTES**

  O sistema teve uma cobertura de 60% das linhas, como pode ser observado na imagem abaixo. O módulo Gestão foi construido direcionado a testes, por esse motivo obteve uma porcentagem alta de cobertura. O módulo menu também apresentou alta cobertura, essa cobertura denotou certa facilidade em ser alcançada, pois o módulo tem como responsabilidade apenas a gestão de mensagens para a execução. No entanto o grupo se deparou com uma dificuldade com os módulos Cadastro e Login, pois não foram construidos direcionados por testes o que dificultou posteriormente a criação dos mesmos. Essa ausência de testes nos módulos login e cadastro gerou um reflexo na cobertura das classes de exceções. Para que esses dois módulos possam ter uma cobertura elevada, há a necessidade de uma reestruturação dessas classes, porém o grupo optou por não realizar essa reestruturação para poder se ater ao prazo de entrega com um software em funcionamento.

  Os arquivos de testes se encontram na pasta tests.

  Os arquivos para acesso à cobertura do código se encontram na pasta coverage.

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level				
Test:	coverage.info	**Lines:**	194	321	60.4 %
Date:	2019-06-14 19:38:56	**Functions:**	28	47	59.6 %

Directory	Line Coverage		Functions	
busca	67.6 %	23 / 34	75.0 %	3 / 4
cadastro	0.0 %	0 / 80	0.0 %	0 / 9
excecoes	45.5 %	10 / 22	45.5 %	5 / 11
gestao	97.7 %	128 / 131	93.3 %	14 / 15
login	0.0 %	0 / 21	0.0 %	0 / 2
menu	100.0 %	33 / 33	100.0 %	6 / 6

Generated by: LCOV version 1.14

➢ CONCLUSÃO

O software irá oferecer um melhoramento funcional para uma biblioteca, agilizando todas as etapas devido a sua característica self checkout.

A utilização de arquivos de texto para gerenciar dados importantes, principalmente para cadastro e autenticação deverá futuramente ser trocada por um banco de dados mais robusto e confiável.

Deste sistema fica para melhoramentos futuros: a inclusão de relatórios gerenciais para o proprietário da biblioteca, limitação no número de empréstimos possíveis, negativa de empréstimo e financeiro.

➢ BIBLIOGRAFIA

http://www.cplusplus.com/reference/fstream/ofstream/?kw=ofstream

http://www.cplusplus.com/reference/fstream/ifstream/?kw=ifstream

https://pt.wikibooks.org/wiki/Programar_em_C%2B%2B/Entrada_e_sa%C3%ADda_de_dados_2

http://www.cplusplus.com/reference/cctype/isupper/?kw=isupper

http://www.cplusplus.com/reference/cctype/tolower/

http://www.cplusplus.com/reference/algorithm/?kw=algorithm

http://www.cplusplus.com/reference/exception/exception/?kw=exception

https://virtual.ufmg.br/20191/pluginfile.php/240169/mod_resource/content/0/projeto-final.pdf

https://www.youtube.com/watch?v=Rl50qI6e7HU

http://www.doxygen.nl/manual/