UNIVERSIDADE FEDERAL DE VIÇOSA

CAMPUS FLORESTAL

Programação Orientada a Objetos

RELATÓRIO FINAL DO PROJETO DE DESENVOLVIMENTO

Layon Fonseca Martins	4220
Melissa Araújo	4244

FLORESTAL

2022

Sumário

1 - Introdução

No presente trabalho foi desenvolvido um Sistema para controle de pedidos e cardápio virtual para ser utilizado em uma cafeteria/livraria física denominado JAVA Li. Neste sistema, a experiência será dividida entre clientes e funcionários da loja, onde os clientes poderão fazer pedidos no cardápio e os funcionários realizarão o controle de pedidos e cadastro de produtos. Este programa foi implementado para estar disponível em máquinas dispostas pela cafeteria/livraria.

Para o controle de pedidos, os funcionários terão acesso à fila de pedidos feitos pelos clientes do estabelecimento, remover itens da fila, e checar a disponibilidade de livros e de produtos do café. Além disso, poderão adicionar itens no cardápio e livros na biblioteca. Já no caso dos clientes, eles poderão acessar o espaço de cardápio com as opções de bebidas e comidas, personalizar seus pedidos (mais açúcar, mais creme, etc.) e acessar o acervo de livros disponíveis, podendo escolher entre comprar ou ler no local.

1.1 - Execução do arquivo .jar

Para a execução do arquivo .jar, é necessário estar no diretório “target”, para isso, através da pasta principal, o caminho é “JAVALi\javali\target”. Em seguida, introduzir o comando a seguir em um terminal localizado na pasta mencionada anteriormente. Além disso, é preciso estar conectado à internet para o bom funcionamento do banco de dados.

```
java -jar javali-1.0-SNAPSHOT-jar-with-dependencies.jar
```

De maneira semelhante, pode-se fazer o download do pacote .jar e rodar o comando acima na pasta onde o arquivo foi baixado. Por fim, é possível acessar o sistema através do login “admin” e senha “123”, assim como ilustrado na Figura 1.

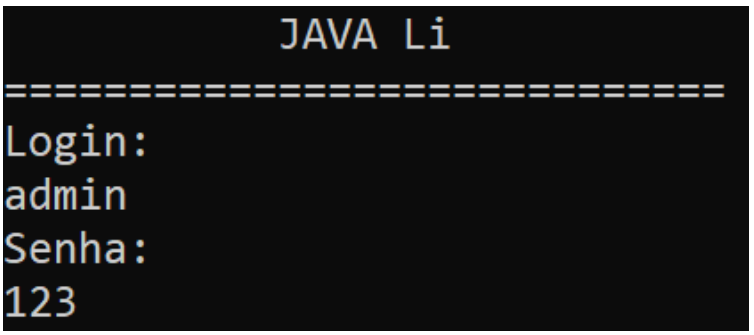

Figura 1 - *login* e senha de acesso ao sistema

2 - Banco de dados

Na Figura 2, está exposto o diagrama Entidade-Relacionamento do banco de dados relacional aplicado no sistema. Primeiramente, a tabela “funcionario” é uma identidade fraca definida pela entidade forte “usuario” que contém os dados referentes à permissão de acesso ao programa. Em seguida, as relações “bebida”, “comida” e “livro” são os produtos vendidos na cafeteria/livraria e cada uma tem os atributos necessários para as funcionalidades implementadas no sistema que serão explicadas na seção “Implementação”. Ademais, também tem-se a tabela “cliente” que contém a mesa em que um cliente está localizado para facilitar o atendimento dos funcionários. As entidades “bebida”, “comida”, “livro” e “cliente” estão relacionadas com “pedido” através de “pedidoBebida”, “pedidoComida” e “pedidoLivro”, que são tabelas que representam relacionamentos muitos para muitos.

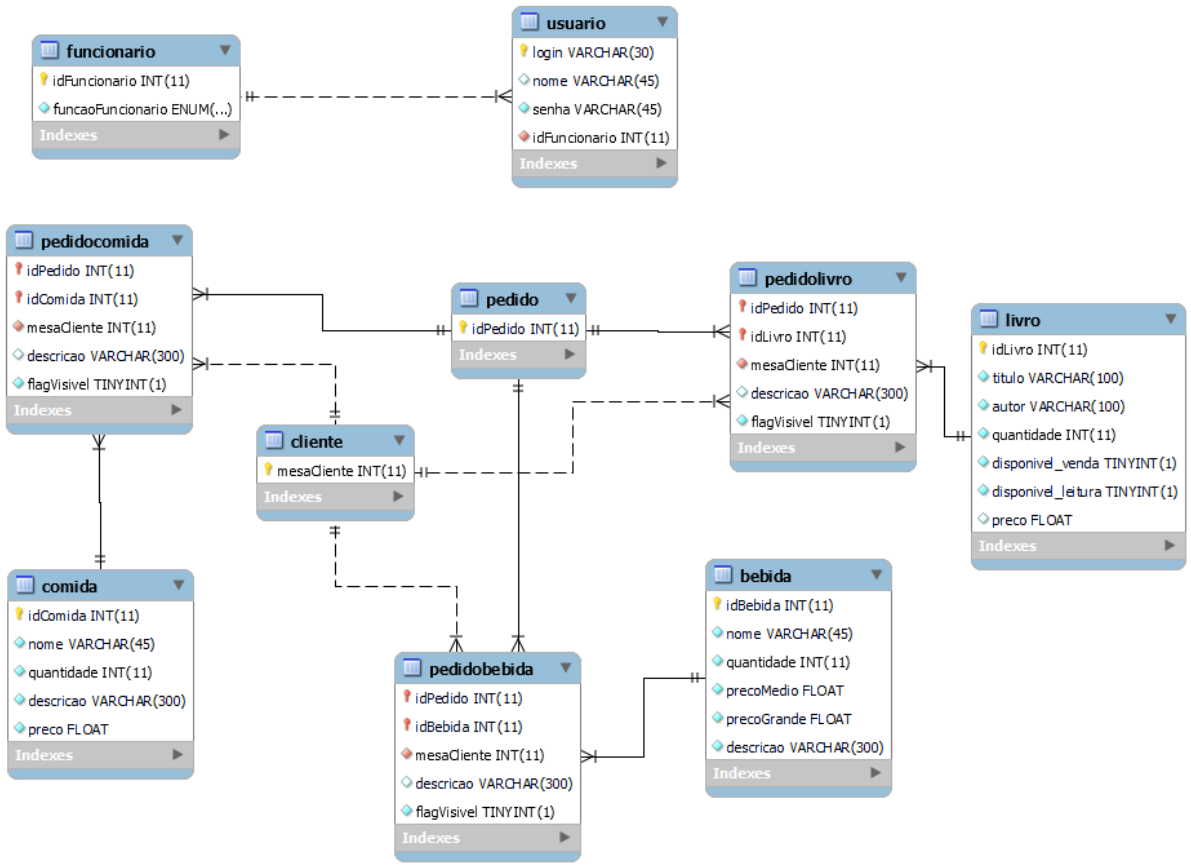

Figura 2 - Diagrama Entidade-Relacionamento utilizado no sistema.

Vale ressaltar que o banco de dados foi hospedado na internet através do site “Heroku”, então não é necessário nenhuma configuração adicional para que o mesmo funcione. No entanto, é preciso estar conectado a uma rede de internet.

3 - Implementação

A implementação do código se deu dentro do padrão Modelo, Visão e Controle (Model-View-Controller - MVC), sendo assim, as classes desenvolvidas estão separadas em pastas de mesmo nome, sendo Modelo as classes voltadas para a definição da estrutura das entidades do sistema, tratamento de exceção e o acesso ao banco de dados, Visão as classes de interação gráfica com o usuário e Controle as classes que utilizam os dados introduzidos pelo usuário nas demais classes. As classes que compõem cada uma dessas categorias serão explicadas em detalhe a seguir.

3.1 - Modelo

As classes que pertencem à categoria Modelo são responsáveis pela definição das entidades, acesso ao banco de dados e tratamento de exceções. Referente às entidades do sistema, foram utilizadas Aplicativo, Bebida, Cliente, Comida, Funcionario, Livro, Pedido, Produto, Usuario, além de FuncaoFuncionario, que é uma classe enum desenvolvida como um dos atributos de Funcionario. A classe Aplicativo representa a classe principal, que contém a função main e executa o código. Produto, que está ilustrada no Trecho de Código 1 é uma interface que descreve os produtos vendidos na cafeteria/livraria, esses sendo descritos com mais detalhes em classes que implementam Produto, como Bebida, Comida e Livro. Foram considerados atributos gerais o identificador de cada produto, que corresponde à chave primária de cada entidade e utilizou-se do polimorfismo para aplicar as funções “getIdProduto()” e “setIdProduto()”, alterando seu comportamento de acordo com a entidade em que está sendo utilizada.

```
public interface Produto {
    public int getIdProduto();
    public void setIdProduto(int idProduto);
}
```

Trecho de Código 1 - Interface Produto

Ademais, temos Usuario, Funcionario e Cliente, que são os usuários que interagem com o sistema, sendo que Funcionario e Cliente herdam de Usuario. Usuário contém os atributos “login”, “nome” e “senha”, onde “login” e “senha” são utilizados para acessar o sistema. Além dessas classes, tem-se o enum FuncaoFuncionario que pode assumir os valores "BARISTA", "CONFEITEIRO", "GERENTE" e "ATENDENTE" que descrevem a função do funcionário na cafeteria. Vale citar que, caso o funcionário tenha função "GERENTE", ele terá permissão especial de cadastrar um funcionário.

3.1.1 - Exceção

Neste pacote, implementou-se as classes que foram utilizadas para personalizar novas exceções no projeto. Para isso, as classes de exceção criadas herdaram da classe Exception o método getMessage() através da palavra reservada extends. Assim, foi possível lançar as exceções criadas e tratá-las evitando eventuais conflitos no fluxo do projeto.

Notou-se a viabilidade de implementar três novas classes de exceção: **ExcecaoUsuarioInvalido**, **ExcecaoLivroIndisponivelVenda** e **ExcecaoLivroIndisponivelLeitura**, elas estão descritas a seguir:

- **ExcecaoUsuarioInvalido**: esta exceção ocorre quando uma tentativa de login é solicitada e os atributos para realizá-lo estão inválidos, ou seja, ao verificar o login e senha salvos no banco de dados esses dados ou estão errados ou este usuário não foi cadastrado. Desse modo, a exceção é lançada em UsuarioDAO, que realiza a verificação de login, pelo método throw como uma nova exceção sendo propagada pelo método throws em ControleUsuario até ser tratada pelo bloco try-catch em TelaAplicativo onde ocorre a tentativa de login.
- **ExcecaoLivroIndisponivelVenda** e **ExcecaoLivroIndiposnivelLeitura**: estas exceções, foram implementadas com a finalidade de monitorar os pedidos de livros realizados pelos clientes na TelaLivraria pois, os livros disponíveis na livraria possuem duas maneiras de serem solicitados, por compra ou por leitura no local (como um empréstimo). No entanto pode ocorrer do cliente tentar comprar um livro que não está disponível para venda e da mesma maneira tentar realizar o empréstimo de um livro que não está disponível para leitura. Assim, as duas exceções são lançadas em seus respectivos métodos de LivroDAO, verificarDisponibilidadeCompraDAO e verificarDisponibilidadeLeituraDAO que são propagadas para o ControleLivro pelo

método throws para serem tratadas quando um novo pedido de livro for solicitado pelo telaFazerPedidoLivro em TelaPedidos. Vale ressaltar que as duas exceções são tratadas pelo mesmo try com 2 catch pois o processo para pedir um livro é apenas um.

3.1.2 - Persistência

Em "Persistencia" estão presentes as classes referentes às operações no banco de dados. Primeiramente, tem-se a classe "BancoDeDados", que está responsável por realizar a conexão com o banco de dados, que está hospedado na plataforma "Heroku", assim como disposto no Trecho de Código 2, e criar operações pré-compiladas que são generalizadas para que possam ser personalizadas de acordo com os dados necessários. As demais classes são divididas na interação de cada entidade com as relações do banco de dados. "ProdutoDAO" é uma interface que contém funções generalizadas para os produtos da cafeteria/livraria, essas sendo "lerProdutosDAO()" e "atualizarEstoqueProdutosDAO()" e utiliza-se do polimorfismo para alterar o comportamento dessas funções nas classes referente ao DAO de Bebida e Comida.

```
      Connection conexao = null;
            try{
                Class.forName("com.mysql.cj.jdbc.Driver");
                conexao = DriverManager.getConnection

("jdbc:mysql://us-cdbr-east-05.cleardb.net:3306/heroku_16b6b58
624fc4f3",
                "b4d39d2a4a36d8", "9ac48a14");
                return conexao;
            }
```

Trecho de Código 2 - Conexão com o banco de dados

Concomitantemente, "BebidaDAO", "ComidaDAO" e "LivroDAO" contém métodos que permitem a inserção, recuperação e atualização dos dados desses produtos no banco de dados na relação análoga a cada um deles, que são utilizados para as funcionalidades do sistema de cadastro, cardápio e atualização de estoque, respectivamente. No caso específico

de “LivroDAO”, existem também funções que verificam a disponibilidade de um livro a ser comprado ou lido no local.

Em relação às classes “FuncionarioDAO” e “UsuarioDAO”, tem-se métodos relacionados à interação do funcionário com o programa, sendo elas a recuperação de dados para o processo de *login* no sistema, cadastro de funcionários – que só pode ser realizado pelo gerente, cuja verificação também está incluída nessas classes – e cadastro de mesas. Outrossim, a classe “PedidoDAO” está relacionada às operações nas tabelas que relacionam “pedido” com “cliente” e as relações dos produtos. Sendo assim, estão incluídos os métodos que recuperam os pedidos a serem exibidos na fila de pedidos, atualiza a visibilidade dos pedidos para que estes sejam escondidos da fila e adiciona pedidos à fila de pedidos estão presentes nessa classe.

3.2 - Visão

Esta seção descreve as classes incluídas no pacote relacionada à interação gráfica com o usuário. A experiência deste sistema é diferente para um funcionário e para um cliente, de forma que os funcionários têm acesso às telas de ambos os usuários e os cliente podem visualizar apenas o cardápio e os livros presentes na livraria. A primeira tela requer a entrada de um “login” e “senha”, como os clientes vão utilizar o sistema apenas para realizar pedido, não foi considerado necessário que o mesmo acessasse o sistema, então essa tela só está disponível para os funcionários. Esta tela está descrita na classe “TelaAplicativo” que é referenciada no método *main* da classe principal “Aplicativo”. Ainda nessa classe, o funcionário pode escolher se a próxima tela será a tela inicial para clientes ou para funcionários. É importante ressaltar que caso escolha executar a parte do sistema voltada para os clientes, o fluxo não permite que estes usuários acessem as telas presentes em “TelaAplicativo”, então a execução do programa deve ser interrompida e reiniciada para acessar essa tela novamente.

Na seção do sistema voltada para clientes, estão incluídas as classes “TelaCardapio”, “TelaCliente” e “TelaLivraria”. Em “TelaCliente”, é possível visualizar as opções de ser redirecionado para o cardápio ou para as opções de livro do estabelecimento. Em “TelaCardapio”, o usuário pode escolher se deseja acessar o cardápio referente a bebidas ou a comidas, onde estarão expostos o nome, descrição e preço de um produto, um exemplo de um produto do cardápio de bebidas está ilustrado na Figura 3. Em “TelaLivraria”, o usuário pode

acessar todos os livros da livraria e os livros que estão disponíveis para compra ou para leitura separadamente. Nos cardápios de cada produto e nas opções da livraria também é possível realizar um pedido, onde a tela está implementada em “TelaPedidos”.

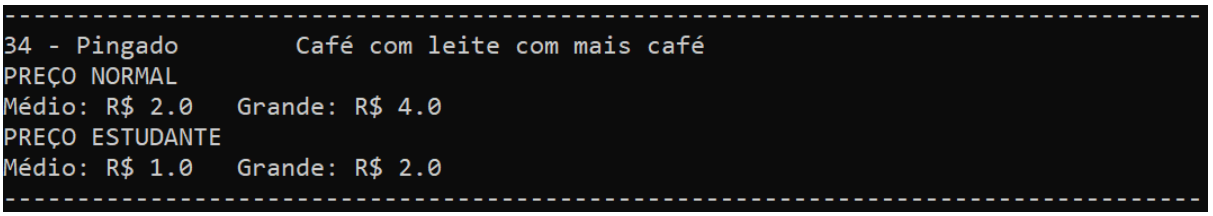

Figura 3 - Exemplo de produto no cardápio.

Já na parte do sistema desenvolvida para os funcionários, além daquelas descritas anteriormente presente em “TelaAplicativo”, tem-se “TelaFuncionario”, como podemos observar na Figura 4, estão dispostas as telas de cadastrar produtos, fila de pedidos, cadastrar funcionários, atualizar estoque e cadastrar mesa. Para as telas de cadastrar produtos, funcionários e mesas, o usuário deve inserir todas as informações necessárias para inserir as referentes entidades no banco de dados. Na tela onde a fila de pedidos está disposta, o funcionário pode visualizar os pedidos que ainda não foram atendidos e também remover pedidos da fila caso estes já tenham sido atendidos, onde as informações serão inseridas em métodos presentes na classe “TelaPedidos”.

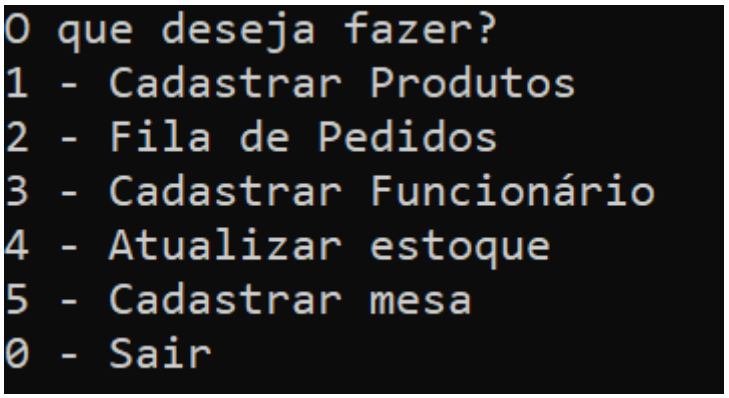

Figura 4 - Tela de opções do funcionário

3.3 - Controle

Como citado anteriormente, as classes pertencentes ao Controle são responsáveis por gerenciar os dados que são passados através da experiência do usuário com a Visão, com o objetivo de criar objetos e definir prioridades que serão passados para o Modelo e assim para o banco de dados.

Foram implementadas as seguintes classes:

- ControleBebida, ControleComida e ControleLivro: são referentes ao gerenciamento dos produtos vendidos na cafeteria como cadastro e atualizações.
- ControleUsuario e ControleFuncionario: responsáveis pelo controle das permissões e ações de um funcionário.
- ControleCardapio: referente ao resultado de solicitações ao DAO para disponibilizar os itens cadastrados para montar um cardápio para os clientes.
- ControlePedidos: realiza o controle principal dos pedidos realizados pelos clientes, criando objetos para serem enviados ao banco de dados pelo PedidoDAO para que se possa montar uma fila de pedidos para os funcionários.

Cada método de uma classe do Controle, conforme a necessidade, recebe como parâmetro os dados necessários para criar um objeto da entidade correspondente para enviá-lo para os métodos que realizam o relacionamento com o banco de dados, como no Trecho de Código 3.

```
public void controleCadastrarComida(String nome, String
descricao, int quantidade, double preco)throws SQLException,
ClassNotFoundException{
    Comida comida = new Comida(0, nome, descricao,
quantidade, preco);
         dao.cadastrarComidaDAO(comida);
       }
```

Trecho de código 3 - controleCadastrarComida();

Desse modo, todas as classes do controle criam uma instância de um objeto de sua entidade referente em “Persistencia” para acessar os métodos DAO.

Assim, as classes desenvolvidas na seção Visão criam instâncias de objetos do Controle que serão utilizados para acessar esses métodos.

3.4 - Testes

A fim de verificar as funções desenvolvidas, foram realizados alguns testes através da biblioteca JUnit que serão explicados a seguir. O teste “logarTest1()” utiliza da função logar para verificar se retorna o valor verdadeiro para um login válido. Já o método “cadastrarFuncTest1()” utiliza da função que verifica se o funcionário é um gerente para permitir que o mesmo cadastre um funcionário, e verifica se o valor retornado é verdadeiro para o *login* de um gerente, enquanto “cadastrarFuncTeste2()” verifica se a função retorna um valor falso para funcionários com *logins* válidos que não são gerentes. O teste “disponivelLeituraTest1()” identifica se um livro que está disponível para leitura tem como resultado verdadeiro se passado como parâmetro para a função “controleLivroDisponivelLeitura()” de Controle Livro, assim como “disponivelCompraTest2()” faz a mesma verificação para para livros que estão disponíveis para compra na função “controleLivroDisponivelCompra()”. Vale notar que outros testes foram feitos ao decorrer do desenvolvimento do programa mas foram utilizados para detectar a necessidade de tratamentos de exceção, como *logins* inválidos para acessar o sistema e para cadastrar um funcionário e livros que não estão disponíveis para compra e leitura sendo passados para as função “controleLivroDisponivelLeitura()” e “controleLivroDisponivelCompra()” respectivamente.

4 - Conclusão

Neste trabalho, foi implementado um sistema que busca auxiliar um estabelecimento de cafeteria e livraria a administrar seus pedidos e produtos presentes no estoque. Dessa forma, o objetivo deste trabalho foi aprofundar os conhecimentos de programação orientada a objetos na linguagem de programação Java adquiridos na disciplina.

Além do âmbito do aprendizado, com a finalidade de manter uma boa organização entre o grupo e, consequentemente, um bom desenvolvimento do trabalho, foram utilizadas algumas ferramentas, as quais serão descritas a seguir. Primeiramente, utilizou-se o GitHub para possibilitar o controle de versões do código fonte e a extensão Live Share da IDE Visual Studio Code para permitir uma codificação colaborativa nos momentos síncronos de reunião. Para a manipulação e planejamento do banco de dados, utilizou-se o Sistema de Gerenciamento de Banco de Dados MySQL Workbench e para hospedar o mesmo, foram aplicados o serviço ClearDB ofertado pela plataforma de aplicação em nuvem Heroku. Como ferramenta de compilação do sistema, utilizou-se o Maven. Além disso, o Google Meet foi a plataforma escolhida para os encontros síncronos do grupo.

Por fim, a partir dos esforços do grupo, foi possível completar a proposta do projeto e alcançar o objetivo principal. O transcorrer do trabalho se deu com algumas complicações que foram resolvidas eventualmente.