Universidade Estadual do Oeste do Paraná – UNIOESTE
Acadêmicos: Hamã Cândido Carvalho Lopes
Maurício Hiraaki Ishida
Disciplina: Tecnologias para Desenvolvimento de Sistemas
Docente: Luiz Antonio Rodrigues

Descrição do Sistema Client Manager desenvolvido na disciplina de Tecnologias para Desenvolvimento de Sistemas

Problema: Atualmente uma empresa não possui controle de seus gastos e ganhos, então é necessário que seja implementada uma ferramenta para que seja feito o lançamento de despesas e lucros, tanto com pessoas físicas como jurídicas, o sistema deve permitir cadastro de clientes, notas fiscais e descrição de serviços. Deverá emitir relatórios de lucros e despesas, que auxiliarão no controle contábil da empresa.

Solução: Será implementado um software em linguagem JAVA, utilizando como padrão de projeto MVC (Modelo, Visão, Controlador) realizando persistência com banco de dados PostgreSQL. Para desenvolvimento da interface gráfica foi utilizada a ferramenta JavaFX Scene Builder. Foi utilizada a IDE IntellijIDEA para o desenvolvimento do projeto.

Foram utilizadas as seguintes classes no desenvolvimento do sistema:

Client: A classe *Client* é composta pelos seguintes atributos: code: int (Código que identifica o usuário, sendo gerado por um hashcode), name: String (Nome do cliente e/ou serviço), spent: float (valor total dos serviços prestados), balance: float (Balanço de gastos e ganhos sobre um determinado cliente), winnings: float (total pago pelo serviço), services: ArrayList<Invoice>(Lista de todos os serviços contratados por um cliente). A partir da classe Client, temos duas classes que herdam seus atributos, sendo elas, *ClientF* e *ClientJ.* Onde *ClientF*, representa clientes do tipo pessoa física, e o *ClientJ*, é utilizado para representar clientes do tipo pessoa jurídica.

ClientF: Além dos atributos herdados da classe *Client*, a classe *ClientF* possui um atributo próprio, cpf: String (Armazena o número do Cadastro de Pessoa Física).

ClientJ: Além dos atributos herdados da classe *Client*, a classe *ClientJ* possui um atributo próprio, cnjp: String (Armazena o número do Cadastro de Pessoa Jurídica).

Invoice: A classe *Invoice* é composta pelos seguintes atributos: id: int (Código que identifica a despesa), ClientID: int (Número gerado pelo hashcode para identificar um cliente), description: String (Descrição sucinta para identificação do gasto), client: Client (Ponteiro de referência para a classe *Client*), spent: float (Valor gasto para realizar um serviço), totalcost: float (Valor total do serviço), winningPercentage: float (Porcentagem de ganho em cima de serviço prestado).

Main: A classe *Main,* Inicializa o banco de dados, controlador e view.

As classes contidas no pacote **DAO**, são utilizadas na conexão com o banco de dados, sendo elas:

BDConnection: A classe BDConnection é responsável por abrir a conexão com o banco de dados, e emite erro caso não conecte.

ClientDao: A classe ClientDao é responsável por realizar a inserção, remoção e atualização do banco de dados, via SQL.

InvoiceDao: A classe InvoiceDao é responsável por realizar a inserção e remoção do banco de dados, via SQL

As classes do pacote **Controls** realizam as manipulações de dados entre o banco e a view, sendo elas:

CadastroClienteController: A classe CadastroClienteController é responsável por realizar as operações referentes a sua interface fxml, sob o nome de CadastroCliente.fxml

CadastroInvoiceController: A classe CadastroInvoiceController é responsável por realizar as operações referentes a sua interface fxml, sob o nome de CadastroInvoice.fxml

ClientDAOController: A classe ClientDAOController é responsável por chamar os métodos da classe ClientDao.

HistoricoViewController: A classe HistoricoViewController é responsável por realizar as operações referentes a sua interface fxml, sob o nome de HistoricoFaturas.fxml

InvoiceDAOController: A classe InvoiceDAOController é responsável por chamar os métodos da classe InvoiceDao.

UserView: A classe UserView é responsável por realizar as operações referentes a sua interface fxml, so o nome de sample.fxml

Os arquivos contidos na **Views**, são responsáveis por gerir a interface gráfica do sistema, foi utilizada a ferramenta JavaFX Scene Builder para o desenvolvimento da interface gráfica do sistema, sendo gerados os arquivos .fxml para cada tela referente ao sistema. Dentro de cada fxml é especificado o que cada tela controla (botões, labels, tabelas e outros).