INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE

AMANDA EMANUELE DA TRINDADE

SOFTWARE DE GERENCIAMENTO DE CLIENTES E DEVEDORES DA LISSA MODAS

LAJES – RN

2019

AMANDA EMANUELE DA TRINDADE

SOFTWARE DE GERENCIAMENTO DE CLIENTES E DEVEDORES DA LISSA MODAS

Trabalho de Conclusão de Curso apresentado em cumprimento às exigências legais como requisito parcial à obtenção do título de Técnico de Nível Médio em Informática, na modalidade Integrado.

Orientador: Prof. Me. Dannilo Martins Cunha.

LAJES – RN

2019

AMANDA EMANUELE DA TRINDADE

SOFTWARE DE GERENCIAMENTO DE CLIENTES E DEVEDORES DA LISSA MODAS

Trabalho de Conclusão de Curso apresentado em cumprimento às exigências legais como requisito parcial à obtenção do título de Técnico de Nível Médio em Informática, na modalidade Integrado.

Trabalho apresentado e aprovado em ___/___/____, pela seguinte Banca Examinadora:

_____

Dannilo Martins Cunha

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

_____

Diogo Eugênio da Silva Cortez

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

_____

Katiúscia Lopes dos Santos

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

AGRADECIMENTOS

Primeiramente, agradeço ao meu orientador, Dannilo Martins Cunha, por ter tido disponibilidade ao longo da produção de todo o trabalho. Sou grata a todos os professores do IFRN, que ao longo desses anos me proporcionaram sabedoria para conseguir implementar e documentar esse trabalho. Agradeço também aos meus pais e aos meus colegas por todo incentivo fornecido ao longo dessa trajetória.

RESUMO

Sabe-se que os estabelecimentos financeiros sempre necessitaram de um mecanismo de controle. Hoje em dia, várias empresas utilizam a tecnologia, já que essa "inovação" pode desempenhar um papel fundamental dentro das organizações. Esse trabalho apresenta um *software* desenvolvido para a loja Lissa Modas, localizada em Angicos/RN, com o intuito de auxiliar a administração, principalmente para cadastramento e atualização de informações de clientes classificados como devedores. Dessa forma, a tecnologia empregada proporciona uma praticidade, rapidez e margem de erros baixa na contabilidade dos dados, trazendo assim, ganhos organizacionais ao empreendimento. Para que esse trabalho fosse realizado, foi preciso ocorrer conversas com a gerente da loja para estabelecer as funções que o software deveria executar, as quais fossem de extrema importância para o gerenciamento da loja. Assim, o *software* foi responsável por cadastrar clientes, cadastrar contas de devedor e atualizá-las.

Palavras-chave: *software*, loja, cadastramento, atualização, devedores.

ABSTRACT

It is common knowledge that financial establishments has the need for a control mechanism. Now a days, several companies use the tecnology, due the technological revolution which can play a key role within organizations. In this way, this project presents a software developed for the store Lissa Modas, located in Angicos/RN, with the purpose of assisting the administration, mostly for registration and updating of information of clients classified as debtors. Therefore, the technology employed provides practicality, speed and a low margin of error in accounting of data, thus bringing organizational gains to the enterprise. In order for this work to be carried out, conversations were required with the store manager to establish the functions that the software should perform, which were of extreme importance for the management of the store. Ass result, the software was responsible for registering customers, registering debtor accounts and updating them.

Keywords: software, store, registration, updating, debtors.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A tecnologia possibilitou grandes avanços nas mais diversas áreas. De acordo com Fleury (1993), a tecnologia pode ser considerada um conjunto de diversas informações organizadas, as quais são resultantes de fontes conquistadas a partir de métodos, como pesquisas e desenvolvimento. Desde a Segunda Guerra Mundial, essa modernização trouxe um rápido fluxo de informações e hoje é a grande responsável por diversas inovações que colaboram para o trabalho do ser humano, inclusive em estabelecimentos financeiros.

Embora seja importante utilizar um mecanismo de controle, algumas gerências ainda optam em fazer isso manualmente, como a loja Lissa Modas que está localizada no centro do município de Angicos, estado do Rio Grande do Norte (Brasil). O estabelecimento armazena informações dos clientes, como nome, endereço, contato e sua respectiva dívida, todas por escrito. Esse processo mecânico pode atrapalhar o andamento da administração caso não seja feito de forma organizada, além de imprevistos e erros que podem ser ocasionados pelo trabalho humano, como a perda dos dados.

Segundo Igbaria, Parasuraman e Baroudi (1996), a utilização de microcomputadores feita por gerentes não realiza às expectativas das empresas. Mesmo tendo em vista que essa condição de rejeição pode delimitar o uso da tecnologia por parte dos gerentes, é importante ressaltar a praticidade que um *software* pode proporcionar ao substituir o trabalho manual. Nesse caso, o uso de um sistema pode tornar a atividade de gerenciamento mais rápida e prática, além de proporcionar uma margem de erros muito baixa, principalmente na contabilidade dos dados. Por esse motivo o seguinte trabalho tem o intuito de agregar uma ferramenta tecnológica ao funcionamento da loja que irá proporcionar um *software* que auxiliará a gerente, tendo que estar instalado em um computador do estabelecimento. No mais, o *software* não precisa de internet para funcionar e erros causados por humanos (como valores digitados errados) não são tratados.

Nessa circunstância, esse trabalho procura empregar a tecnologia de *softwares* como uma ferramenta simples na gestão de clientes voltada para pagamentos pendentes da respectiva loja. Com esse propósito, o *software* será responsável pelo armazenamento das informações dos clientes, principalmente os classificados como sendo devedores, a fim de melhorar o gerenciamento destes. Assim, espera-se que, com essa proposta, o gerenciamento dos clientes devedores seja realizado de uma maneira mais efetiva e confiável, diminuindo alguns dos possíveis prejuízos financeiros inerentes ao estabelecimento.

1.1. OBJETIVO GERAL

O seguinte trabalho tem o propósito de fornecer um *software* capaz de gerenciar informações de clientes que são classificados como devedores.

1.2. OBJETIVOS ESPECÍFICOS

Para atingir o objetivo geral desse trabalho foi preciso seguir os passos listados abaixo:

- Analisar as principais dificuldades encontradas na gestão de pagamento dos clientes do estabelecimento com foco nos devedores;
- Descrever os requisitos do *software*;
- Produzir a interface gráfica do *software*;
- Desenvolver o banco de dados para guardar as informações dos clientes;
- Implantar as funcionalidades fornecidas pelo *software* através da linguagem Java;
- Implementar o *software* proposto através da ligação das ferramentas utilizadas;
- Analisar os resultados e propor possíveis melhorias no software.

1.3 METODOLOGIA DO TRABALHO

O processo utilizado no desenvolvimento do seguinte trabalho foi o modelo incremental. Esse modelo foi usado para definir como organizar e planejar todas fases essenciais para a realização do *software*. Inicialmente, presenciei uma reunião junto com a gerente da loja para apurar os problemas, para então coletar as funcionalidades essenciais do *software*. Logo após, foi realizado a etapa do planejamento, em que foram delimitados tempos estratégicos para fazer as atividades.

Cada atividade teve um tempo variado entre 2 e 4 semanas e ao término de cada etapa foram feitas reuniões com o orientador para apurar os resultados e se preciso fazer alguns ajustes para melhorar a etapa. Além disso, esses encontros serviram para debatermos como iria ser feito o procedimento da próxima atividade, em questão de estudo, para ser executada a próxima tarefa.

Através de pesquisas realizadas em plataformas online e artigos bibliográficos foram adquiridos conhecimentos para complementar a construção desse trabalho. No desenvolvimento do trabalho, foram utilizadas tecnologias já vistas no decorrer da trajetória acadêmica. Com o NetBeans, foi possível implementar a interface gráfica e fazer a conexão

com o banco de dados. Para a criação e gerenciamento de dados, o aplicativo MySQL foi utilizado.

2 REFERENCIAL TEÓRICO

Neste capítulo são mostrados tópicos teóricos que foram de extrema importância para a execução desse trabalho, como as tecnologias que auxiliaram o andamento e criação desse trabalho.

2.1 Linguagem de programação

Uma linguagem de programação se classifica como um meio comunicativo de instruções humanas repassadas para o computador. De acordo com Gotardo (2015, p 17), uma linguagem de programação é responsável por expressar códigos para um computador de um determinado programa. Para isso, a linguagem deve ser capaz de unir regras sintáticas e semânticas para garantir um bom funcionamento do programa. Sendo assim, as regras sintáticas se referem ao modo da escrita (como letras e números) e as regras semânticas ao conteúdo, ou seja, ao significado de cada escrita de linguagem. As linguagens de programação podem ser divididas em: linguagens de baixo nível e linguagens de alto nível. As de baixo nível são interpretadas facilmente/rapidamente pelo computador, como por exemplo a linguagem binária e a assembly. Já as de alto nível necessitam de um compilador, pois não são interpretadas diretamente pelo computador, por isso ao iniciar um programa, é preciso criar um código fonte que contém instruções lógicas e logo após é necessário compilar esse arquivo para que então o programa se torne executável.

No respectivo trabalho, será utilizada a linguagem de programação Java. “A linguagem de programação Java representa uma linguagem simples, orientada, multithread, interpretada, neutra de arquitetura, portável, robusta, segura e que oferece alto desempenho” (MENDES, 2009, p 17). Essa linguagem de programação é encarregada de se apropriar de instruções escritas que são processadas e executam uma determinada tarefa.

2.2 Programação orientada a objetos

A programação orientada a objetos é um padrão de desenvolvimento seguido por várias linguagens, C# e Java são algumas delas. Nesse paradigma, os escopos dos métodos são diferentes, porém o processo dos métodos segue o mesmo padrão. As linguagens que utilizam a programação orientada a objetos possuem as seguintes características: abstração, encapsulamento, herança e polimorfismo. A abstração diz respeito em como o objeto é visto

dentro de um sistema e como ele será realizado, para isso, é preciso que se pense em uma identidade para que o objeto seja criado, pois ele necessita ter sua característica própria. Após isso, é necessário refletir sobre as particularidades que esse objeto terá e definir as ações (métodos) que ele irá executar. O encapsulamento é outra característica da POO que assegura a segurança da aplicação em uma POO, geralmente as linguagens deixam suas implementações com propriedades privadas, assim, existe uma segurança nos momentos de realização e execução. Já a herança garante que um objeto possa herdar característica de outro objeto, assim, a produção do código fica mais enxuto e algo já feito antes pode ser reutilizado. Por último, o polimorfismo é outra função útil, capaz fornecer uma modificação no funcionamento de um método herdado de um objeto. Ou seja, ele dá a alternativa de reaproveitar um objeto caso não possua o mesmo funcionamento, assim, somente uma parte precisa ser alterada. (GASPAROTTO, 2014)

2.3 Banco de dados

"Um Sistema Gerenciador de Banco de Dados (SGBD) é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados. O conjunto de dados, comumente chamado banco de dados, contém informações sobre uma empresa em particular. O principal objetivo de um SGBD é proporcionar um ambiente tanto conivente quanto eficiente para a recuperação e armazenamento das informações do banco de dados". (Siberschatz, Korth e Sudarshan, 1999, p.1). Geralmente, empresas utilizam um banco de dados para reunir informações de clientes, as quais podem ser divididas em grupos, como por exemplo nome e CPF.

Os tipos de banco de dados existentes são: hierárquico, em rede e relacional. Segundo Marçula e Benini Filho (2014, p.183 e 184), o modelo hierárquico mostra as informações armazenadas, os relacionamentos entre essas informações como registro e as ligações entre elas. Essa ligação descreve a associação entre dois registros, em que só existe uma ligação entre os dois registros, como uma coleção de árvores. Assim como o modelo hierárquico, o modelo em rede também demonstra os dados como registros e ligação entre eles. A diferença evidente entre os dois é que o modelo em rede estabelece uma ligação como registros interligados.

Ainda de acordo com Marçula e Benini Filho (2014, p. 185), o modelo relacional, o qual foi utilizado nesse trabalho, foi criado com o intuito de simplificar os SGBD para que mostrasse uma modelagem mais próxima a realidade. Por esse motivo, esse tipo de modelo é

voltado para aplicações comerciais e apresenta os dados através de tabelas, em que cada linha é um relacionamento entre um conjunto de valores.

2.3.1 Modelo relacional

Segundo Elmasri & Navathe (2004, p.90), "O modelo relacional representa o banco de dados como uma coleção de relações". Essas relações são feitas através de tabelas que armazenam informações. As tabelas possuem um nome e são compostas por linhas e colunas. As linhas armazenam campos, os quais são denominados de atributos. O conjunto desses atributos formam uma coluna. Por exemplo, uma tabela, chamada de clientes, pode conter quatro colunas (Nome, Endereço, CPF e dívida). As linhas serão responsáveis por preencher as informações da coluna (José, Rua das Araras, 000.000.000-00, 0).

2.3.2 MySQL

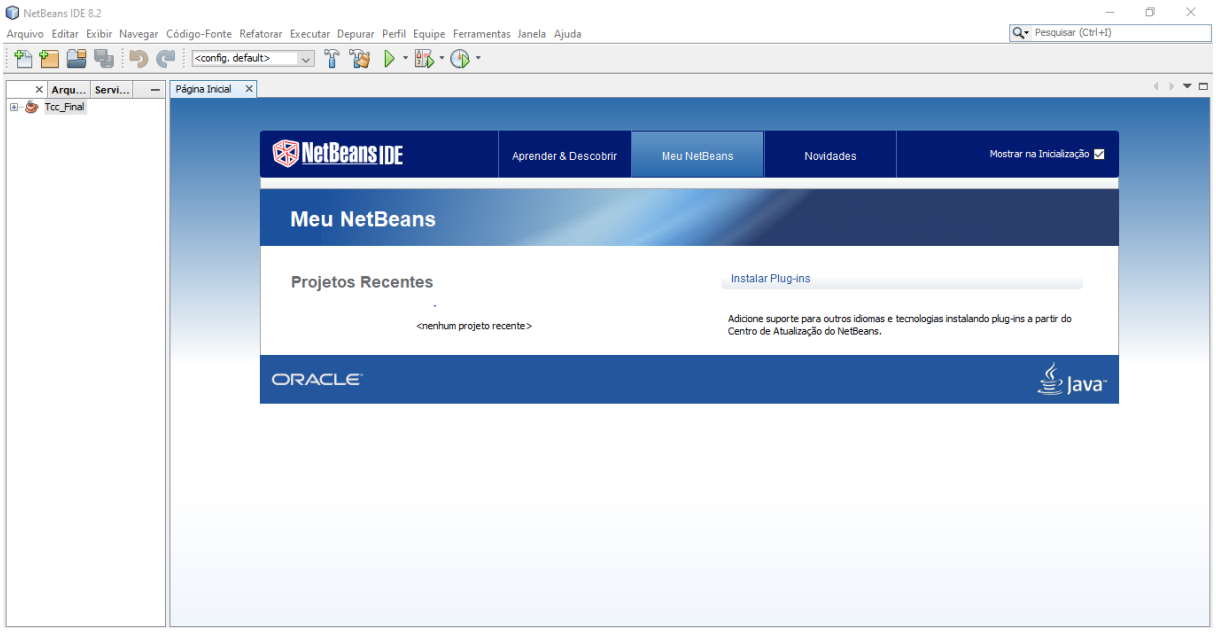
O MySQL, ferramenta utilizada para o desenvolvimento do banco de dados desse trabalho, é um servidor de gerenciamento de banco de dados (SGBD) que possui todas as funções que um banco de dados de grande porte necessita, como por exemplo o armazenamento e o gerenciamento de dados, fornecendo um alto nível de confiabilidade. Segundo André Miani (2006, p. 21), o MySQL é completo em questão de dispor as diversas características que existem nos mais reconhecidos bancos de dados pagos que existem no mercado. Por esse motivo, ele é singular devido a suas licenças para uso gratuito, as quais oferecem a oportunidade de utilização para fins acadêmicos e realização de negócios. A compatibilidade com a maioria dos sistemas operacionais faz do MySQL uma ótima alternativa a ser usada. Além disso, a API (Interface de Programação de Aplicativos) do MySQL proporciona um trabalho conjunto com a linguagem Java.

2.4 NetBeans

De acordo com a Oracle (2018), a IDE NetBeans é um software de código aberto que proporciona o desenvolvimento multiplataforma, facilitando ações de programadores dando suporte a escrever, compilar, debugar e instalar aplicações. O NetBeans foi desenvolvido pela Sun Microsystems e é uma plataforma que garante o desenvolvimento integrado Java. Ela não

estabelece limites para fornecimento e é de fácil instalação e utilização. A IDE pode ser executado em vários sistemas operacionais com o Windows, Linux, Solaris e MacOS.

Figura 1: Tela inicial do NetBeans



Fonte: Autoria própria (2019)

3 DESENVOLVIMENTO

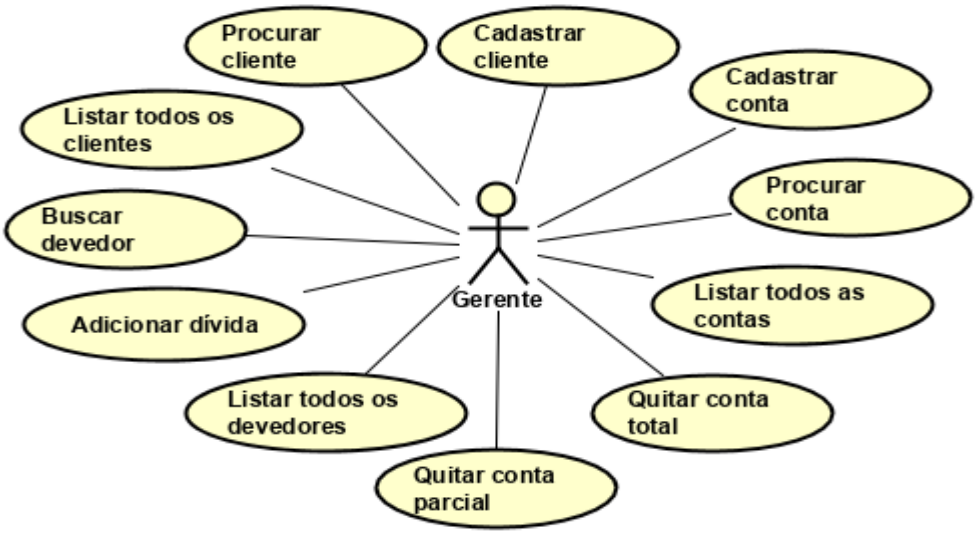
Segundo Celestino (2015), um *software* "é o programa que será instalado no computador do usuário e disponibilizado para uso". Ou seja, um software é responsável por fornecer funções e desempenhos desejados ao ser executado por quem está utilizando. O *software* proposto nesse trabalho foi desenvolvido com o intuito de realizar o controle dos clientes classificados como sendo devedores da loja. Esse irá facilitar a praticidade no dia a dia por parte da gerência, tendo em vista que o processo torna-se mais rápido. Primeiramente, para que fosse produzido o *software* foi feito o levantamento de requisitos para analisar e implementar os serviços principais no *software*. Essa observação foi realizada juntamente com a proprietária da loja, a qual vai usufruir os recursos do *software* em seu estabelecimento.

3.1 Diagrama de caso de uso

Segundo Guedes (2018), o diagrama de caso de uso tem a finalidade de mostrar uma visão externa geral das funções que o software irá oferecer aos usuários. Esse diagrama é útil na compreensão dos requisitos funcionais do *software*, pois auxilia na especificação, visualização e documentação das funcionalidades. O diagrama de caso de uso procura identificar os tipos de usuários que vão interagir com o *software*, quais são os papéis que irão assumir e quais as funções um usuário pode realizar. Em um caso de uso existem dois itens: os atores e os casos de uso. Os atores (mostrados como bonecos no diagrama) representam os papéis desempenhados pelos usuários que vão usufruir dos serviços do software. Assim, um ator pode ser um usuário, um *hardware* ou um *software*. Os casos de uso são os serviços e funcionalidades, tidos como importantes e necessários, oferecidos pelo software e os atores podem utilizá-los. Os casos de uso são representados por elipses e dentro deles existe um texto que expõe a funcionalidade que é oferecida.

Abaixo, encontra-se o diagrama de caso de uso feito para esse trabalho:

Figura 2: Diagrama de caso de uso



Fonte: Autoria própria (2019)

Todos esses casos de uso foram implementados no trabalho e abaixo encontra-se o detalhamento sobre cada um.

Quadro 1: Caso de uso Cadastrar cliente

Cadastrar cliente: Esse caso de uso possibilita que a gerente cadastre um cliente fornecendo os dados dele. **Ator:**

Gerente
Pré-condição:
A gerente fornece o CPF, nome, endereço e contato do cliente.
Pós-condição:
O software registra o cadastro do cliente.
Fluxo básico:
1. Ao clicar na aba de cadastrar cliente, a gerente informa os dados do cliente;
2. Ao clicar no botão “cadastrar”, será exibida uma mensagem informando que o cliente foi cadastrado;
3. O programa salvará os dados do cliente.

Fonte: Autoria própria

Quadro 2: Caso de uso Procurar cliente

Procurar cliente:
Esse caso de uso permite a busca de um cliente pelo seu CPF.
Ator:
Gerente
Pré-condição:
O cliente deve estar cadastrado.
Pós-condição:
Caso exista um cliente com o CPF digitado, são mostrados os dados do cliente.
Fluxo básico:
1. Ao clicar na aba de procurar cliente, a gerente informa o CPF e clica no botão “buscar”;
2. Na tabela são mostrados: CPF, nome, endereço e contato do cliente.

Fonte: Autoria própria

Quadro 3: Caso de uso Listar todos os clientes

Listar todos os clientes:
Esse caso de uso possibilita que seja mostrado todos os clientes cadastrados em uma tabela.
Ator:
Gerente
Pré-condição:
É preciso existir pelo menos um cliente cadastrado.
Pós-condição:
Caso exista clientes, são mostrados os dados dos clientes na tabela.
Fluxo básico:
1. Ao clicar na aba de procurar cliente, a gerente clica no botão “Listar todos os clientes”.
2. Na tabela são mostrados: CPF, nome, endereço e contato de todos os clientes.

Fonte: Autoria própria

Quadro 4: Caso de uso Cadastrar conta

Cadastrar conta: Esse caso de uso permite que a gerente informe o valor da conta de um devedor. **Ator:** Gerente **Pré-condição:** O CPF informado deve ser de um cliente cadastrado. **Pós-condição:** Caso o CPF digitado seja de um cliente, a conta é cadastrada. **Fluxo básico:** 1. Ao clicar na aba de cadastrar conta, a gerente informa o CPF e o valor da conta e clica no botão “cadastrar”. 2.Caso o CPF digitado seja válido e já cadastrado como um cliente, é exibida uma mensagem que a conta foi cadastrada. 3. O software registra os dados da conta.

Fonte: Autoria própria

Quadro 5: Caso de uso Procurar conta

Procurar conta: Esse caso de uso possibilita a procura de uma conta através do CPF do cliente que possui uma conta. **Ator:** Gerente **Pré-condição:** O CPF informado deve ser de um cliente que possui uma conta cadastrada. **Pós-condição:** Caso o CPF seja válido, são mostrados os dados na conta na tabela. **Fluxo básico:** 1. Ao clicar na aba de procurar contas, a gerente informa o CPF e clica no botão “buscar”; 2. Caso o CPF seja se alguém que possua conta, serão mostrados na tabela: ID, CPF, valor e estado.

Fonte: Autoria própria

Quadro 6: Caso de uso Listar todas as contas

Listar todas as contas: Esse caso de uso permite que todas as contas sejam mostradas. **Ator:** Gerente **Pré-condição:** É preciso existir pelo menos uma conta cadastrada. **Pós-condição:**

Os dados das contas são listados em uma tabela.
Fluxo básico:
1. Na aba de procurar contas, a gerente clica no botão “Listar todas as contas”;
2. Na tabela são listadas as informações de todas as contas: ID, CPF, valor e estado.

Fonte: Autoria própria

Quadro 7: Caso de uso Quitar conta parcial

Quitar conta parcial:
Esse caso de uso permite a atualização de uma conta parcialmente.
Ator:
Gerente
Pré-condição:
É preciso ser informado um CPF de alguém que já possua uma conta e é necessário digitar um valor maior que zero e menor do que o presente na conta.
Pós-condição:
A conta é atualizada para um novo valor.
Fluxo básico:
1. Na aba de atualizar, a gerente digita o CPF de um cliente que possui conta, escolhe a opção “Quitar conta parcial” e clica em “confirmar”;
2. Caso o CPF seja válido, é mostrada uma caixa de texto para digitar o valor a ser quitado;
3. Caso o valor seja maior que zero e menor que o presente na dívida, a conta é atualizada para um novo valor.

Fonte: Autoria própria

Quadro 8: Caso de uso Quitar conta total

Quitar conta total:
Esse caso de uso permite que a conta seja zerada.
Ator:
Gerente
Pré-condição:
O CPF informado deve ser válido, ou seja, ser de alguém que possui uma conta.
Pós-condição:
A conta é zerada.
Fluxo básico:
1. Na aba de atualizar conta, a gerente digita o CPF, clica na opção “Quitar conta total” e clica em “confirmar”;
2. Caso o CPF esteja registrado, a conta é zerada.

Fonte: Autoria própria

Quadro 9: Caso de uso Adicionar dívida

Adicionar dívida: Esse caso de uso possibilita a atualização de uma conta que foi aumentada. **Ator:** Gerente **Pré-condição:** O CPF informado deve ser de alguém que possui uma conta. **Pós-condição:** A conta é atualizada. **Fluxo básico:** 1. Na aba Atualizar conta, a gerente informa o CPF, seleciona a opção "Adicionar dívida" e clica em "confirmar"; 2. Caso o CPF seja de uma pessoa que possua conta, é mostrada uma caixa de mensagem solicitando o valor a ser adicionado a conta; 3. A gerente informa o valor e clica em "ok"; 4. A conta é atualizada para o novo valor.

Fonte: Autoria própria

Quadro 10: Caso de uso Buscar devedor

Buscar devedor: Esse caso de uso permite a busca de um devedor. **Ator:** Gerente **Pré-condição:** O CPF informado deve ser de uma pessoa que possua uma conta com o valor maior que zero. **Pós-condição:** São mostrados na tabela os dados do devedor. **Fluxo básico:** 1. Na tabela "Devedores", a gerente clica em "buscar devedor"; 2. É exibida uma caixa de texto solicitando o CPF do devedor que será procurado; 3. Se o CPF informado for de um devedor, são mostrados na tabela: ID, CPF do devedor, valor da conta e o estado (true).

Fonte: Autoria própria

Quadro 11: Caso de uso Listar todos os devedores

Listar todos os devedores: Esse caso de uso permite a visualização de todos os devedores. **Ator:** Gerente **Pré-condição:**

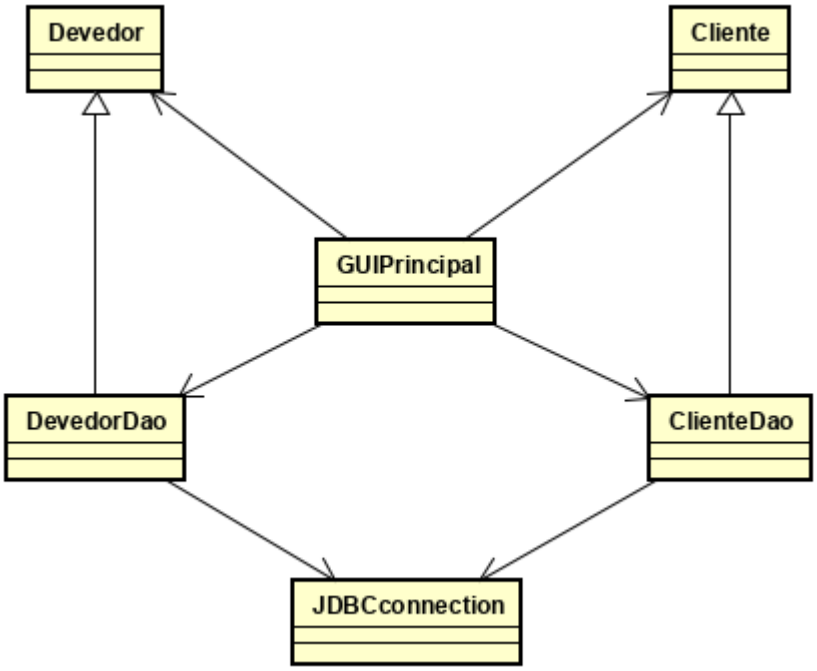
É preciso existir pelo menos um devedor.
Pós-condição:
São mostrados os dados dos devedores na tabela.
Fluxo básico:
1. Na tabela “Devedores”, a gerente clica em “Listar todos os devedores”;
2. São mostrados os seguintes dados: ID, CPF, valor e estado.

Fonte: Autoria própria

3.2 Diagrama de classe

Segundo Booch, Jacobson e Rumbaugh (2016), “um diagrama de classes exibe um conjunto de classes, interfaces e colaborações, bem como seus relacionamentos”.

Figura 3: Diagrama de classe



Fonte: Autoria própria (2019)

Esse trabalho é composto pelas seguintes classes: GUIPrincipal, Cliente, ClienteDao, Devedor, DevedorDao e JDBCConnection.

Na classe Cliente existe os seguintes atributos: cpf, nome, endereço e contato. Esses possuem uma visibilidade privada e são todos do tipo String, ou seja, cadeia de caracteres.

Os atributos citados acima são herdados pela classe ClienteDao. A classe ClienteDao tem os seguintes métodos: adicionarCliente, listarUmCliente e listarTodosClientes, todos eles têm visibilidade pública e recebem como parâmetro um objeto cliente da classe Cliente. O

método adicionarCliente não possui retorno e é responsável por adicionar informações de um cliente de acordo com as colunas da tabela Cliente. O método listarUmCliente retorna uma lista chamada Cliente, que só terá as informações do cliente buscado. O método listarTodosClientes retorna uma lista chamada Clientes, com todas as informações de todos os clientes.

A classe Devedor possui os seguintes atributos privados: id (int), cpf (String), nome (String), valor (float) e estado (String).

A classe DevedorDao herda esses atributos e possui os seguintes métodos: adicionarDevedor, listarUmaConta, listarTodasContas e listarTodosDevedores. Todos esses têm visibilidade pública. O método adicionarDevedor recebe como parâmetro um objeto devedor da classe Devedor, não possui retorno e é responsável por adicionar informações de um devedor de acordo com as colunas da tabela Devedor. O método listarUmaConta também recebe como parâmetro um objeto devedor da classe Devedor e retorna uma lista chamada Devedor, ele é o responsável por exibir os dados de uma conta. O método listarTodasContas retorna uma lista chamada Devedor que contém os dados de todas as contas. O método listarTodosDevedores retorna uma lista chamada Devedor com todas as informações de todos os devedores.

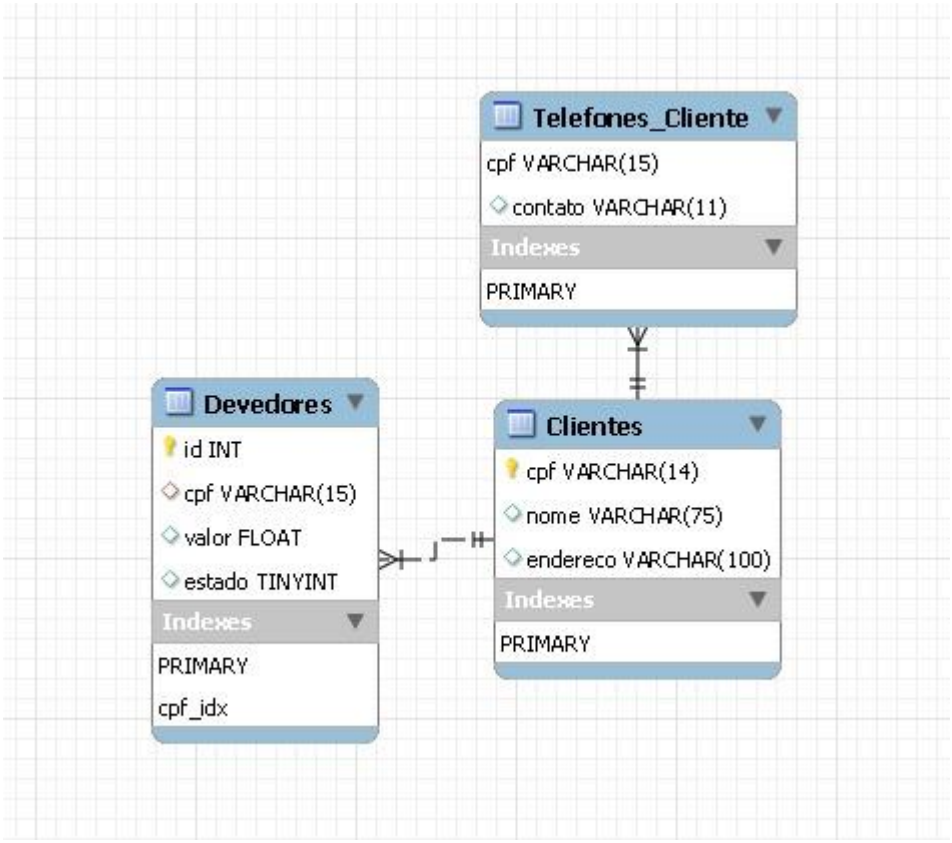
A classe JDBCConnection possui os seguintes atributos: DRIVER, URL, USER E PASS, todos do tipo string, final, static e com visibilidade privada. É responsável por fazer as classes DevedorDao e ClienteDao funcionarem, pois estabelece uma conexão com o banco de dados através de seus atributos. Essa classe possui os métodos getConnection e closeConnection, responsáveis por abrir e encerrar, respectivamente, a conexão com o banco de dados. Esses métodos possuem visibilidade pública e não possuem parâmetros e retornos.

A classe GUIPrincipal possui a interface gráfica do trabalho e as funções dos seguintes botões: botaoCadastrarClientes, botaoBuscarProcurarClientes, botaoLimparProcurarClientes, botaoCadastrarContas, botaoLimparContas, botaoBuscarProcurarContas, botaoLimparProcurarContas, botaoBuscarDevedores, botaoAtualizarContas, botaoLimparDevedores, botaoListarClientes, botaoListarContas, botaoListarDevedores são implementadas nessa classe. Todos possuem visibilidade privada e não possuem retorno. Além disso, ela possui os seguintes métodos de atualização: atualizarTabelaClientes, atualizarTabelaDevedores e atualizarTabelaContas, os quais possuem visibilidade pública e não possuem um retorno. As ações de botaoCadastrarClientes e botaoCadastrarContas são responsáveis por "pegar" os dados fornecidos nos campos de texto e armazenar no banco de dados. A ação executada ao clicar em botaoLimparClientes, botaoLimparContas, botaoLimparDevedores, botaoLimparProcurarClientes e botaoLimparProcurarContas é limpar os campos de textos dessas abas. Os seguintes métodos: botaoBuscarProcurarClientes, botaoBuscarProcurarContas e botaoBuscarDevedores são capazes de listar apenas um cliente, conta e devedor, respectivamente, através de um CPF. As ações implementadas dos métodos botaoListarClientes, botaoListarContas e botaoListarDevedores são responsáveis por mostrar os dados fornecidos em tabelas de clientes, contas e devedores, respectivamente. Portanto, a classe GUIPrincipal tem relação direta com as classes Cliente, ClienteDao, Devedor e DevedorDao para poder executar todas essas funções citadas acima.

3.3 Modelo relacional

Abaixo segue o modelo relacional feito para esse trabalho:

Figura 4: Modelo relacional



Fonte: Autoria própria

A tabela *clientes* possui como atributos cpf (chave primária), nome e endereço. Essa tabela é responsável por armazenar os dados dos clientes. A tabela *devedores* possui os seguintes atributos: id (chave primária), cpf (chave estrangeira), valor e estado. A tabela *Telefones_Clientes* tem os atributos cpf (chave estrangeira) e contato para guardar os dados multivalorados dos clientes.

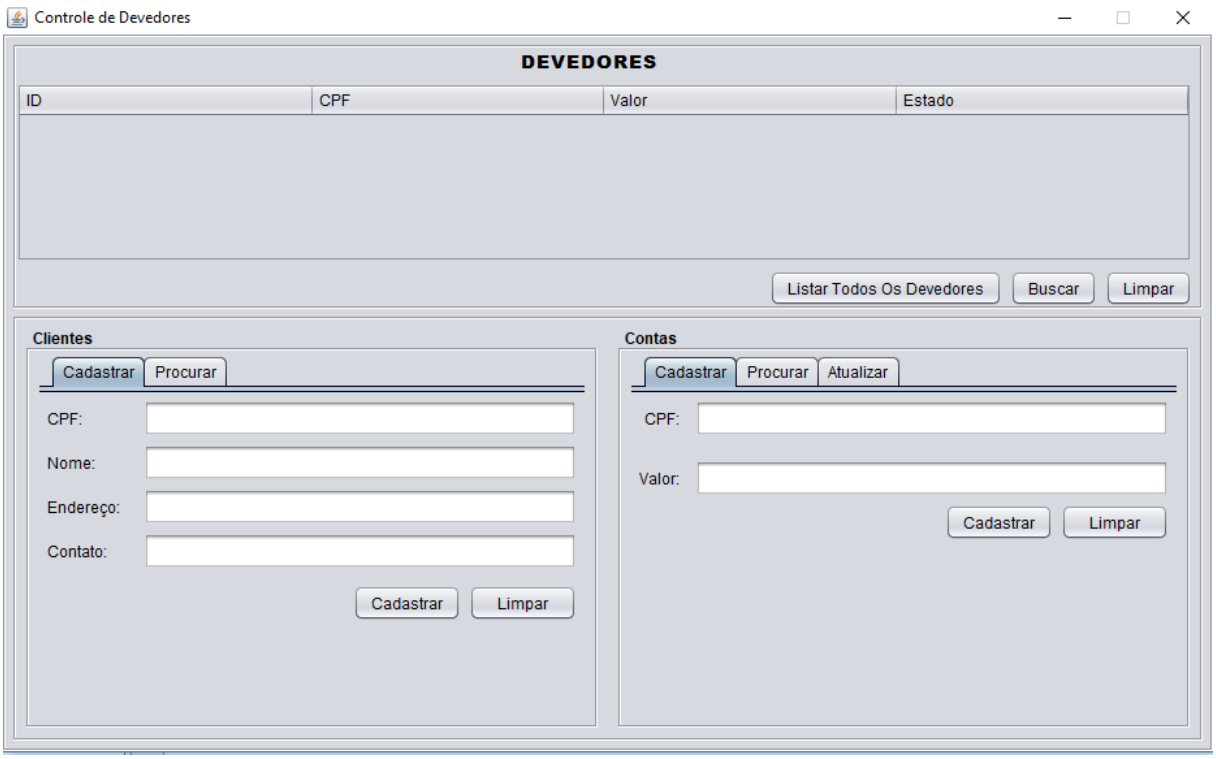
3.4 Resultados

A seguir, será apresentada a tela do programa e suas funcionalidades. Todos os dados mostrados não são de clientes ou devedores reais.

- Tela inicial

Essa tela possui todos as funções do *software*. Nela, encontram-se duas abas, uma de Clientes, outra de devedores. Além delas, existe uma tabela que possui os dados de devedores.

Figura 5: Tela inicial do programa

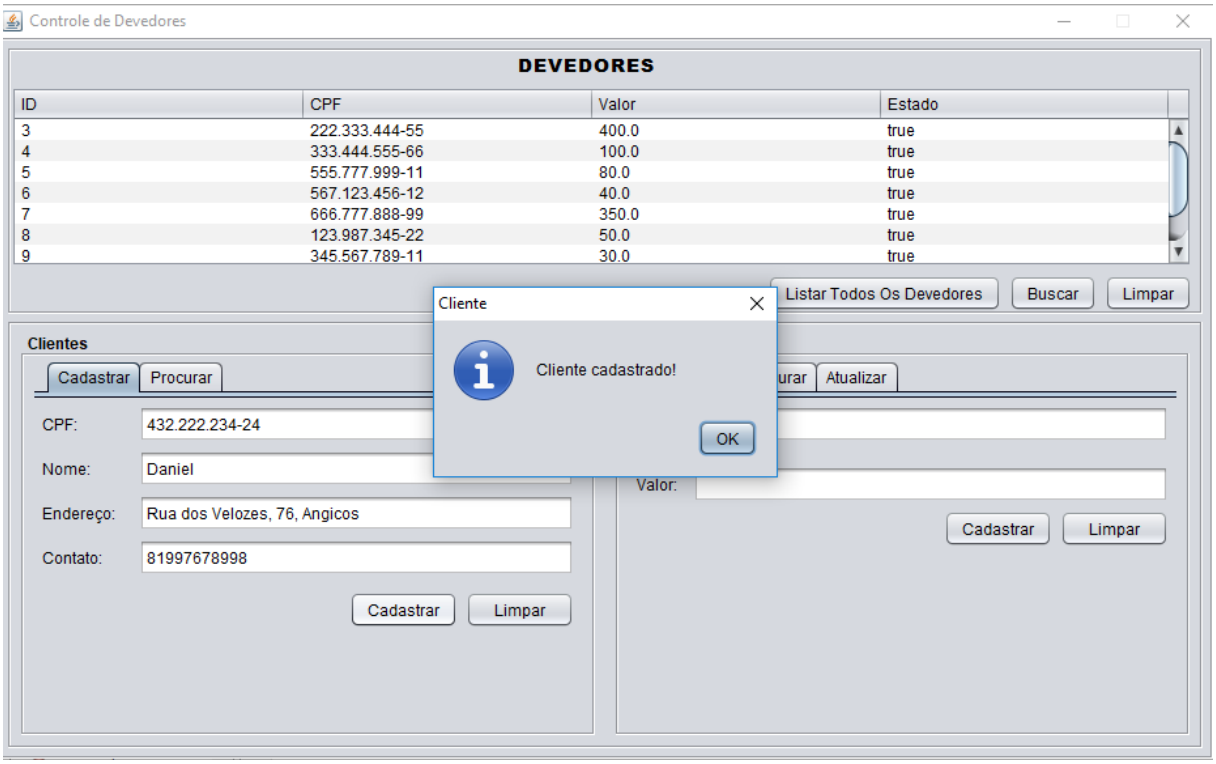


Fonte: Autoria própria (2019)

- Cadastro de cliente

Na aba de Cadastrar cliente, é solicitado o CPF, nome, endereço e contato para que um cliente seja cadastrado.

Figura 6: Cadastro de cliente

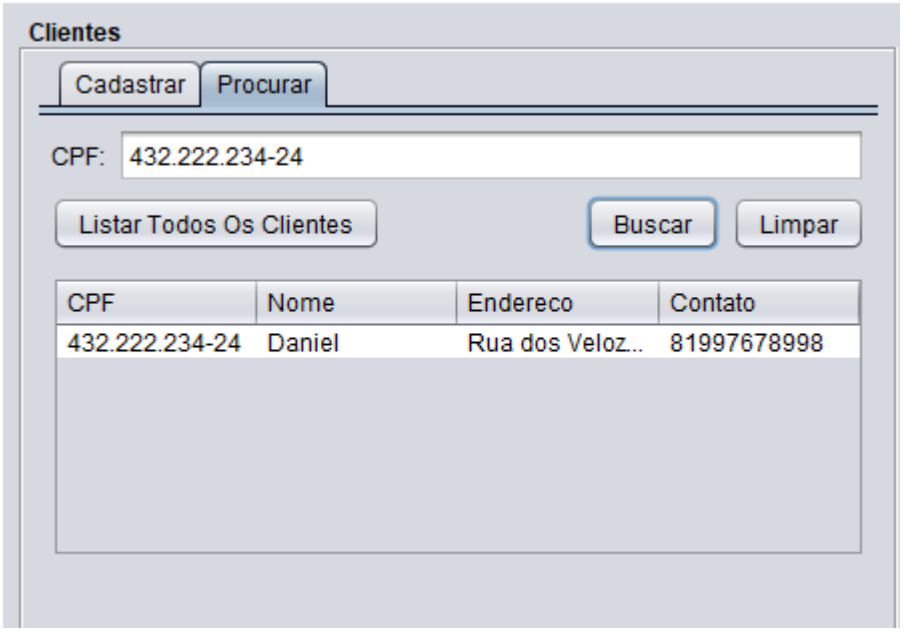


Fonte: Autoria própria (2019)

- Procurar cliente

Na aba de procurar cliente, é informado o CPF, a busca é feita e os dados aparecem na tabela, como é visto a seguir:

Figura 7: Tela de procura pelo cliente

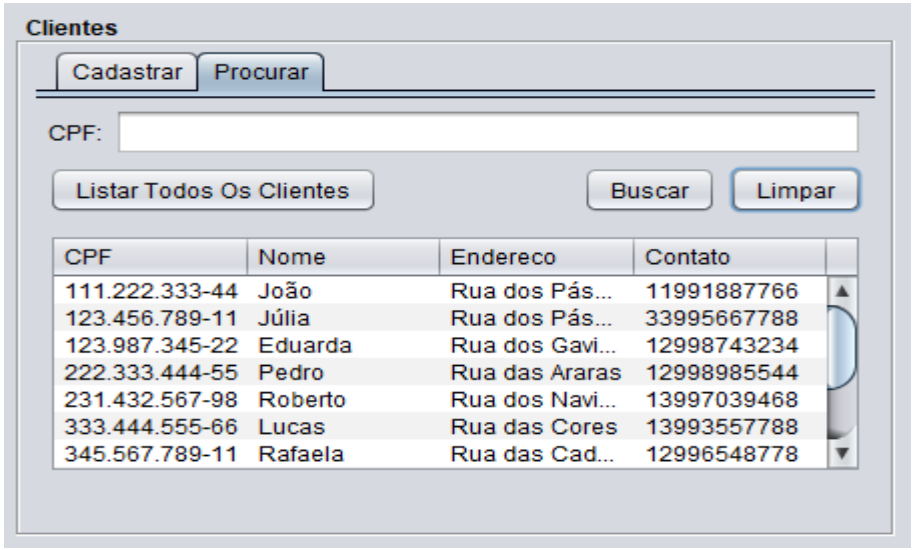


Fonte: Autoria própria (2019)

- Listar todos os clientes

Essa função permite a visualização de todos os clientes cadastrados sendo mostrados na tabela com suas informações (CPF, nome, endereço e contato).

Figura 8: Listar todos os clientes

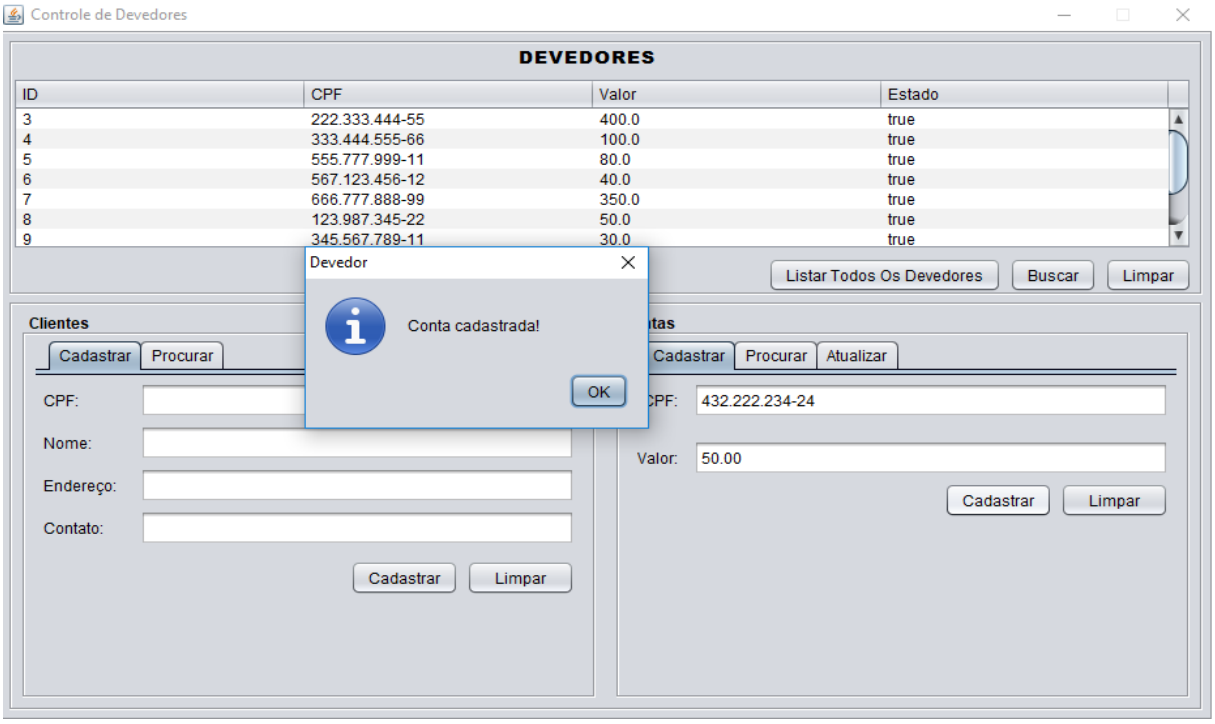


Fonte: Autoria própria (2019)

- Cadastrar conta

Nessa aba, é solicitado o CPF e o valor da conta de uma pessoa que agora passa a ser devedora.

Figura 9: Cadastro de conta de um cliente

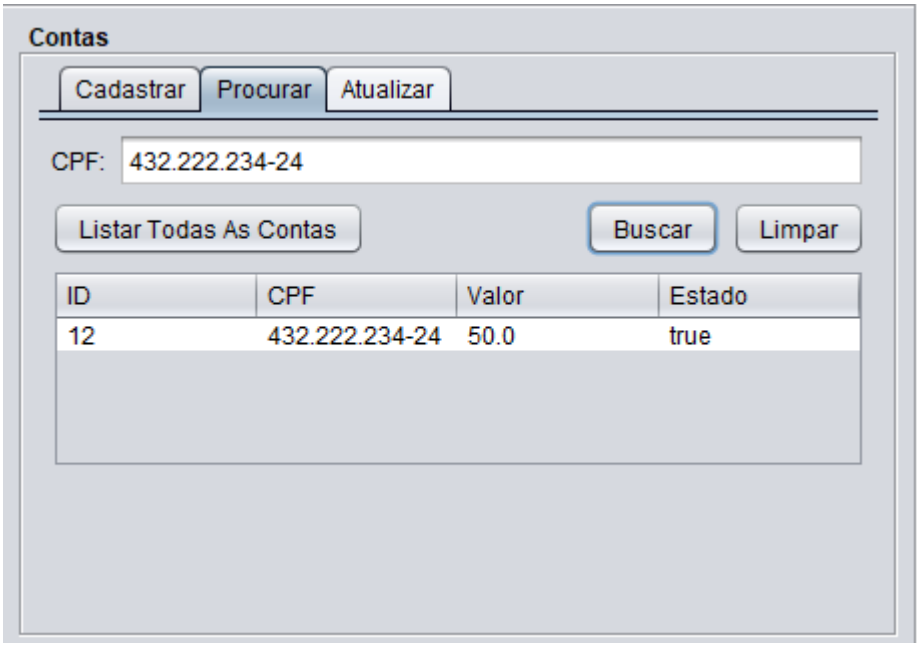


Fonte: Autoria própria (2019)

- Procurar conta

Nessa aba de procurar é informado o CPF, e os dados da conta aparecem na tabela (ID, CPF, valor e estado).

Figura 10: Procurar conta de um cliente

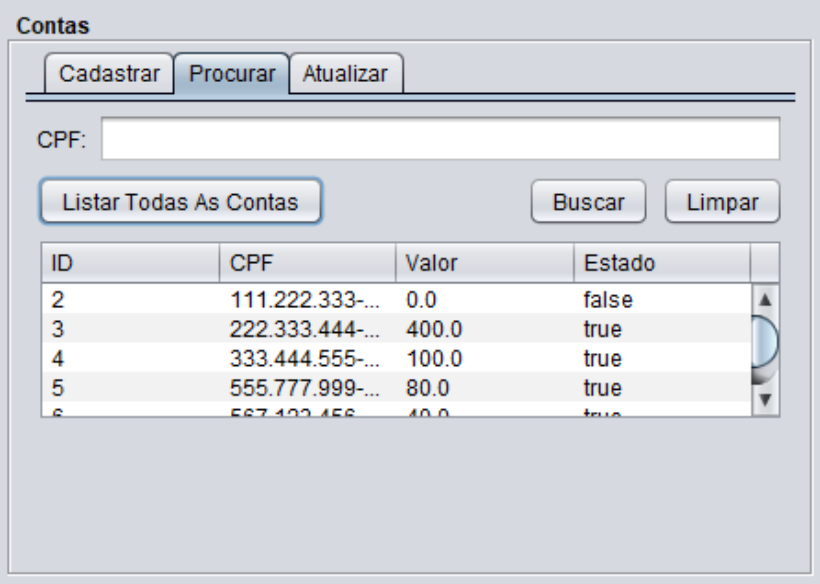


Fonte: Autoria própria (2019)

- Listar todas as contas

Ao solicitar a visualização de todas as contas, a tabela mostra as informações de todas as contas.

Figura 11: Listar todas as contas

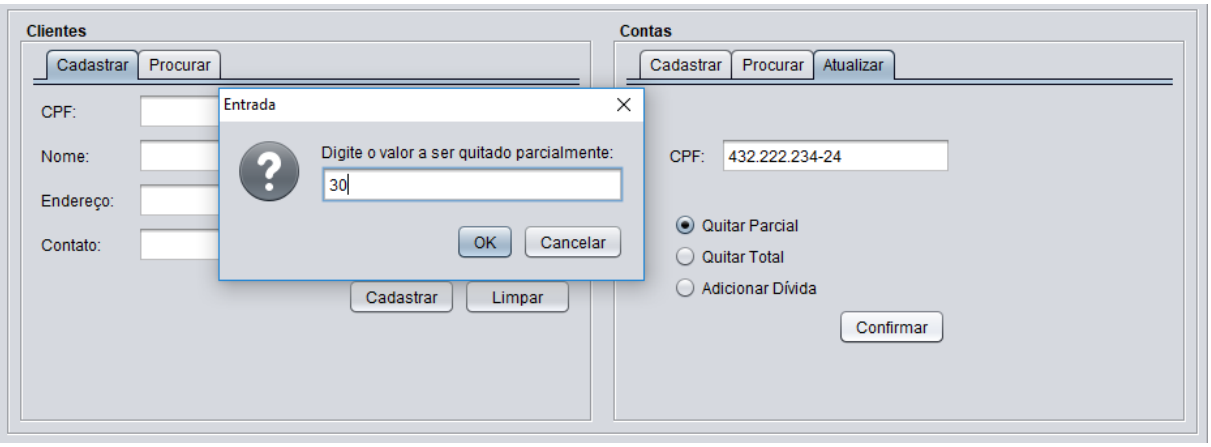


Fonte: Autoria própria (2019)

- Quitar conta parcial

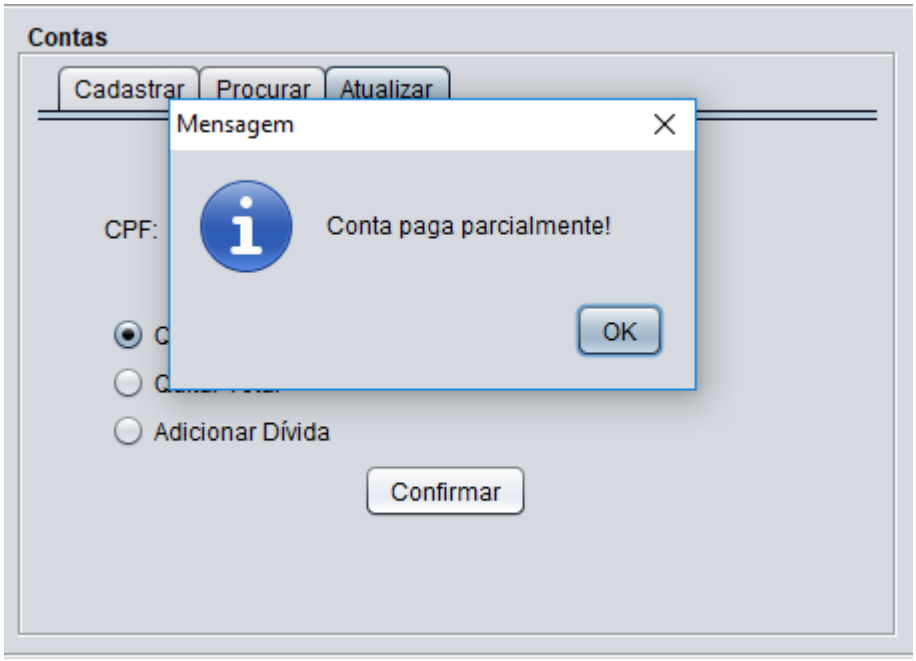
Após informar o CPF e solicitar a opção “quitar parcial” é mostrada a caixa abaixo para atualizar a conta:

Figura 12: Pagar conta parcialmente



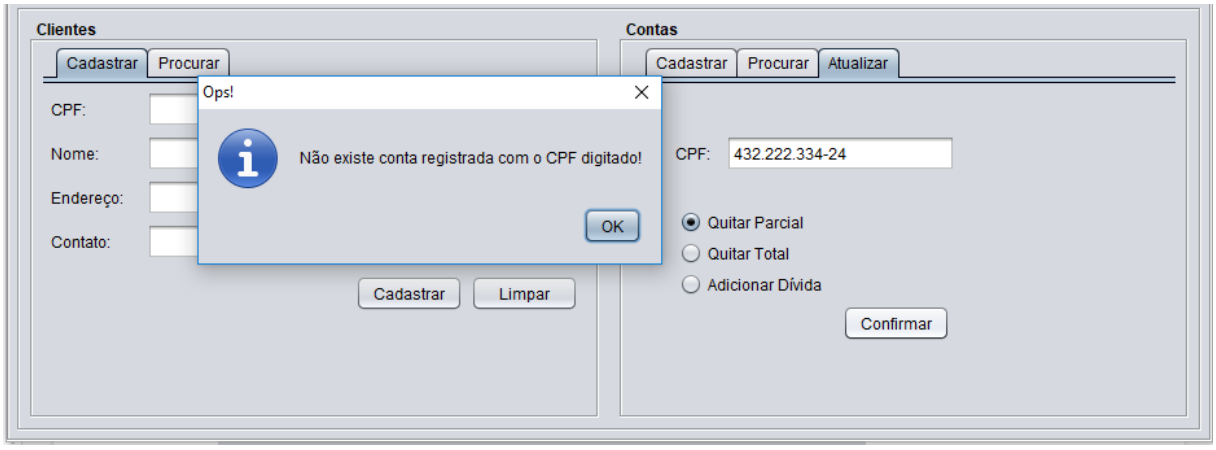
Fonte: Autoria própria

Figura 13: Mensagem mostrada após a atualização de conta parcial



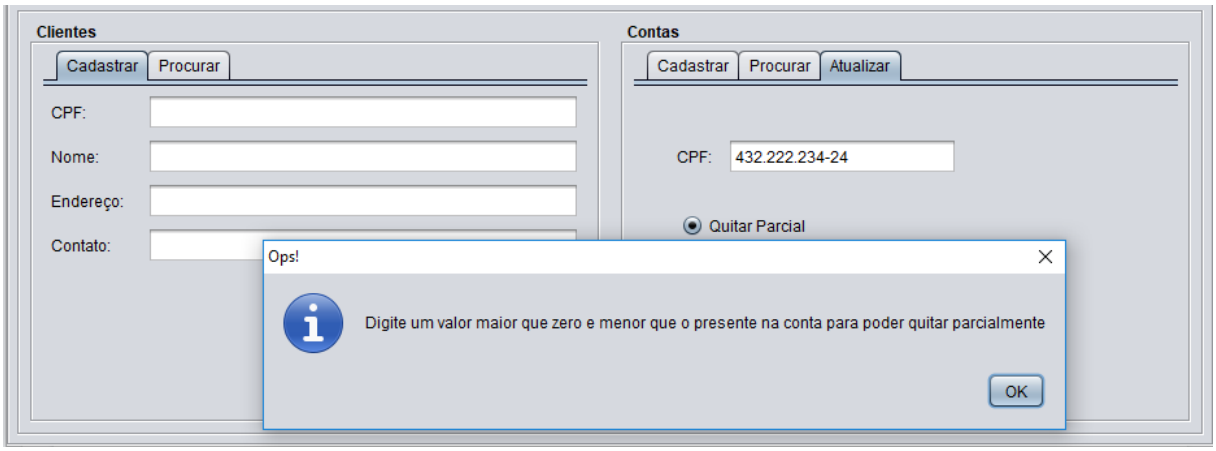
Fonte: Autoria própria (2019)

Figura 14: Mensagem mostrada caso o CPF seja de alguém que não possua conta



Fonte: Autoria própria (2019)

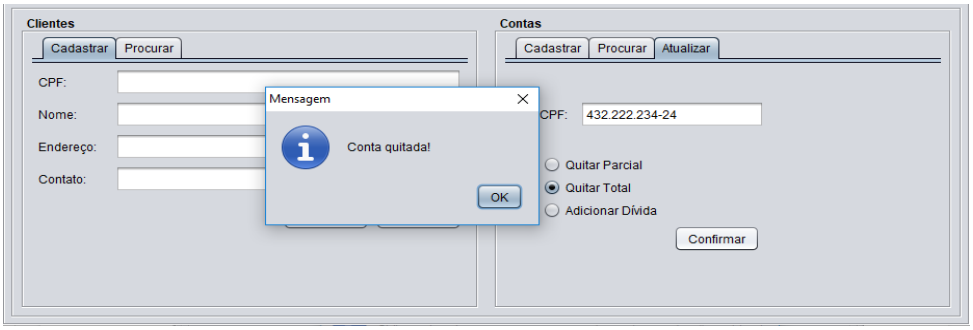
Figura 15: Mensagem mostrada caso o valor informado não seja menor que a conta



Fonte: Autoria própria (2019)

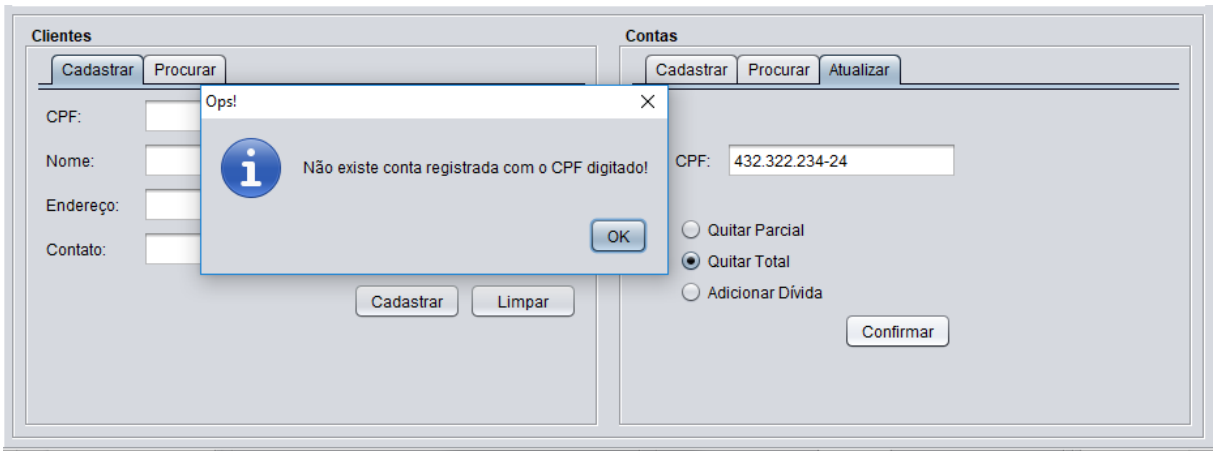
- Quitar conta total

Figura 16: Mensagem mostrada após quitar uma conta total



Fonte: Autoria própria (2019)

Figura 17: Mensagem mostrada caso ocorra um erro

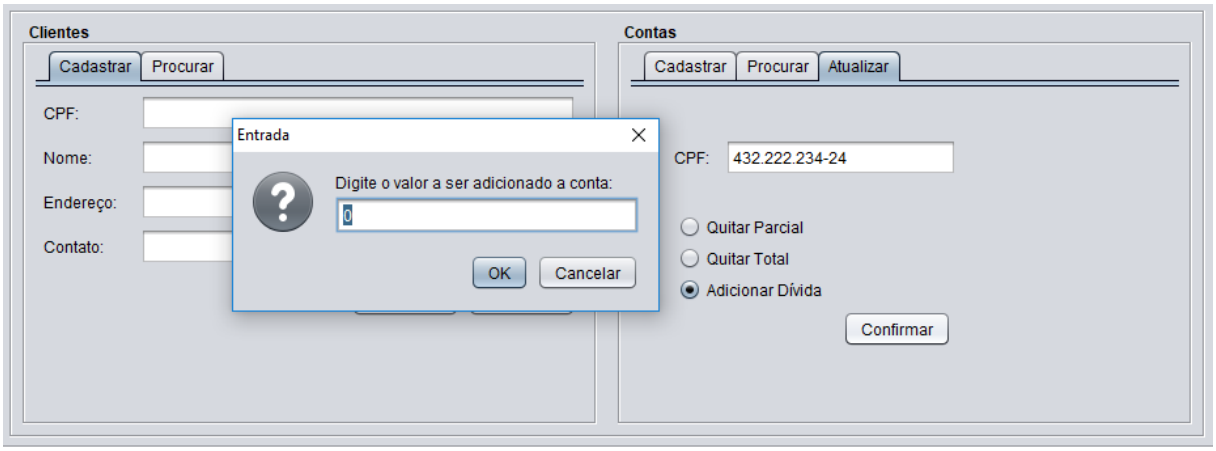


Fonte: Autoria própria (2019)

- Atualizar conta

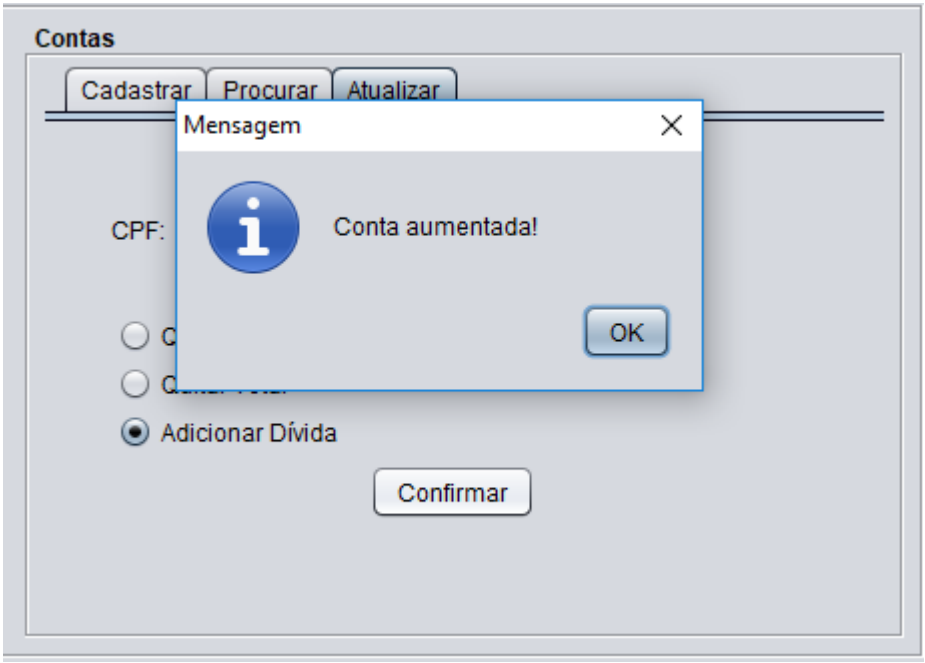
Ao solicitar a adição de uma dívida, é mostrada uma caixa de texto solicitando o valor da adição:

Figura 18: Atualização de conta



Fonte: Autoria própria (2019)

Figura 19: Mensagem mostrada após a atualização da conta

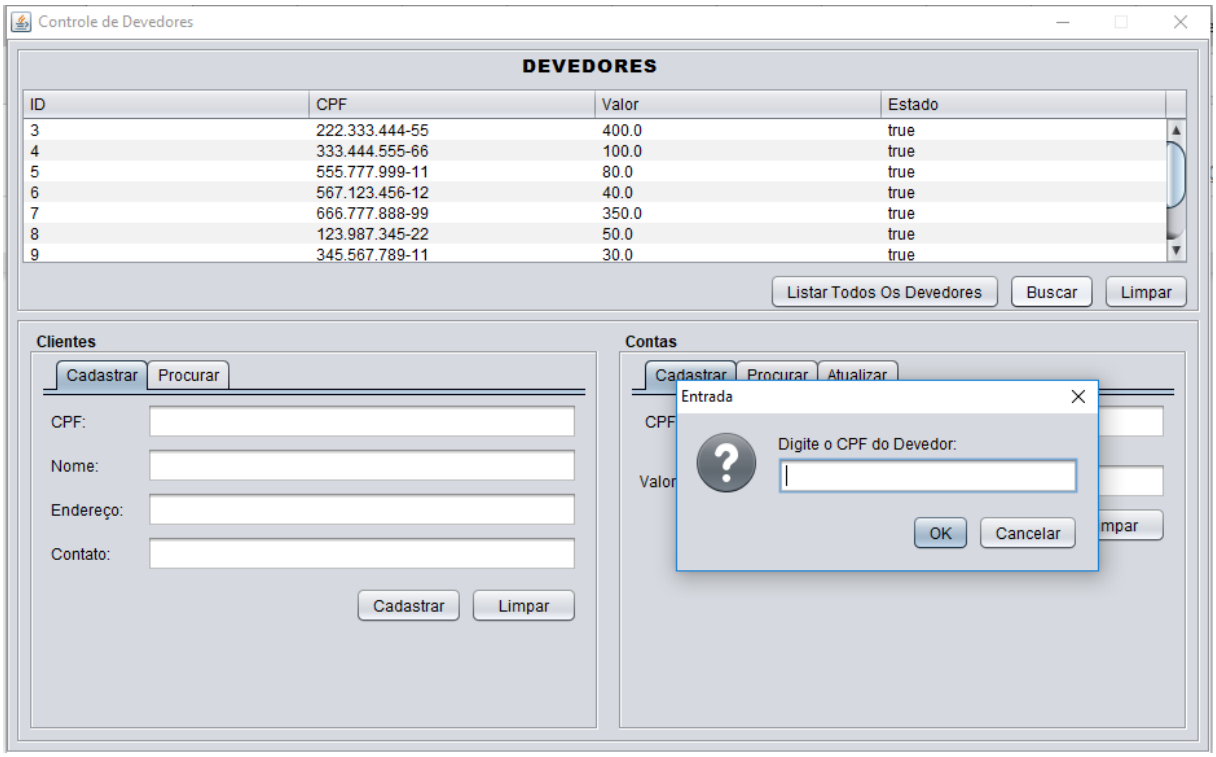


Fonte: Autoria própria (2019)

- Buscar devedor

Ao solicitar a busca de um devedor, é mostrada uma caixa de texto para informar o CPF do devedor que está sendo procurado:

Figura 20: Procura por um devedor



Fonte: Autoria própria (2019)

Após o passo acima, são mostradas as informações da conta do devedor:

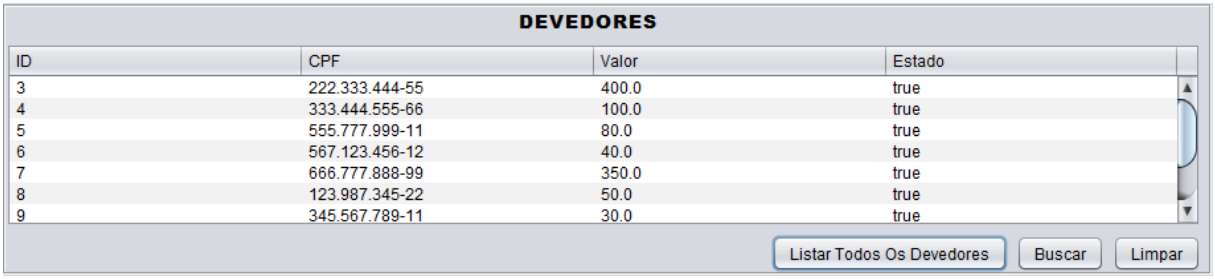
Figura 21: Informações do devedor

Fonte: Autoria própria (2019)

- Listar todos os devedores

Ao clicar no botão “Listar todos os devedores”, são mostradas as informações de todos os devedores:

Figura 22: Listando todos os devedores



Fonte: Autoria própria (2019)

4 CONCLUSÃO

O intuito desse trabalho foi fornecer um software que fosse utilizado pela loja Lissa Modas como uma ferramenta de auxílio no gerenciamento de contas de clientes do estabelecimento, principalmente os tidos como devedores. Pensando na ideia de que um trabalho manual é demorado e pode causar imprevistos, foi pensado na elaboração desse trabalho para melhorar a organização da loja.

Através do *software*, foram fornecidas as funções com extrema importância para o controle de devedores. Mesmo que ainda não tenha sido fornecido para a gerente do estabelecimento, o trabalho é visto como útil, já que através de testes feitos pela desenvolvedora, o *software* foi capaz de guardar e atualizar informações de forma rápida, prática e segura.

Para deixar o trabalho mais completo, foi pensado em deixá-lo com mais utilidades. Portanto, os trabalhos futuros desse trabalho são: excluir clientes e devedores, alterar dados (como nome, CPF, endereço e contato) e proporcionar que mais de um contato seja adicionado (outro número de celular e e-mail).

Por fim, com a realização desse trabalho, foi perceptível o quanto o uso da tecnologia pode ser útil no nosso dia a dia, já que essa é a grande responsável na atualidade por facilitar processos de diversas áreas profissionais.

REFERÊNCIAS

CELESTINO, André Luis. **Você conhece a diferença entre Software e Sistema?**. 2015. Disponível em: https://www.profissionaisti.com.br/2015/04/voce-conhece-a-diferenca-entre-software-e-sistema/. Acesso em: 31 de dez. de 2018.

FLEURY, Afonso. "**Novas tecnologias, capacitação tecnológica e processo de trabalho** - Comparações entre o modelo japonês e o brasileiro." *In*: Sobre o modelo japonês. São Paulo, Editora USP, 1993.

GASPAROTTO, Henrique Machado. **Os 4 pilares da Programação Orientada a Objetos.** 2014. Disponível em: https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264. Acesso em: 23 fev. 2019.

GOTARDO, R. **Linguagem de Programação I**. 1. Ed. Rio de Janeiro, 2015. p.17.

GUEDES, Gilleanes T. A. **UML 2**: Uma Abordagem Prática. 3. ed. São Paulo: Novatec, 2018. 496 p. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=lang_pt&id=mJxMDwAAQBAJ&oi=fnd&pg=PA2&dq=diagrama+de+caso+de+uso+livro&ots=x8vVWp-Pn0&sig=y0dMb4gcALPO5wXzP9jlCcVtBLc. Acesso em: 08 jan. 2019.

HEUSER, Carlos Alberto. Banco de dados: Compartilhamento de dados. In: HEUSER, Carlos Alberto. Projeto de banco de dados. 6. ed. Porto Alegre: Bookman, 2009. cap. 1, p. 20-23. v. 4.

MARÇULA, Marcelo; BENINI FILHO, Pio Armano. Informática: Conceitos e Aplicações. 4. ed. São Paulo: Érica Ltda., 2014.

MENDES, Douglas Rocha. Programação Java com Ênfase em Orientação a Objetos. 1. Ed. São Paulo, 2009. p 17

MILANI, André. MySQL: Guia do Programador. São Paulo: Novatec, 2006. 393 p.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.. Sistema de Banco de Dados. 3. ed. São Paulo: Pearson Makron Books, 1999. 767 p.