Bot Telegram Bank

Integrantes:
Diogo Brito - RM35829
Gabriel Borba – RM39394
Leandro - RM35873
Ismael Maria- RM44591

Table of Contents

1. COMPONENTES

2. BIBLIOTECAS

3. FRAMEWORKS

4. PACOTES, CLASSES E MÉTODOS

O Projeto está distribuído nos seguintes pacotes:

com.telegrambotbank.datatype – Entidades
com.telegrambotbank.enumeration - Enums
com.telegrambotbank.exception – Exceptions Personalizadas
com.telegrambotbank.file.helper – Helpers para manipulação de arquivos
com.telegrambotbank.file.util – Classes complementares para tratamento de arquivos
com.telegrambotbank.main – Possui classe main que executa os procedimentos do bot
com.telegrambotbank.messages – Contém classe com as mensagens mostradas no sistema
com.telegrambotbank.opcoes.helper – Classes que fazem a comunicação com o usuário do sistema
com.telegrambotbank.opcoes.mediator -
com.telegrambotbank.opcoes.util – Classes complementares para utilzar nas opções do sistema
com.telegrambotbank.services – Interfaces do sistema
com.telegrambotbank.services.impl – Implementação das interfaces do sistema
com.telegrambotbank.services.impl.conf

4.1. com.telegrambotbank.datatype

Pacote que contém todas entidades do sistema.

Classes:
ClienteVO
ContaBancariaVO
DependenteVO
DepositoVO
EmprestimoVO
LancamentoVO

Métodos:
Os métodos referentes as classes mencionadas, são respectivamente Getters e Setters do atríbutos de cada classe (conforme diagrama de classes)

4.2. com.telegrambotbank.enumeration

Classes:
OpcoesBotEnum
PosicoesCamposEnum
StringUtilsEnum
TipoContaCorrenteEnum
TipoLancamentoEnum

OpcoesBotEnum – Enum com as opções de funcionalidades que serão mostradas para o cliente:

```
    CRIAR_CONTA("/criarConta"),
    START("/start"),
    DEPOSITAR("/depositar"),
    SACAR("/sacar"),
    EMPRESTIMO("/emprestimo"),
    HELP("/help"),
    TARIFAS("/tarifas"),
    INCLUIR_DEPENDENTE("/incluirDependente"),
    EXTRATO("/extrato"),
    EXIBIR_MINHAS_INFORMACOES("/ExibirMinhasInformacoes");
```

PosicoesCamposEnum – Enum que contém o tamanho mínimo e máximo de cada atributo que será lido/persistido no arquivo texto.

```
    // Dependentes
    NOME_DEPENDENTE(0, 100),
    CPF_DEPENDENTE(11, 11),

    // Conta Corrente
    NU_CONTA_CORRENTE(6,6),
    AGENCIA_CONTA_CORRENTE(0,4),

    // Valor Deposito
    VALOR_DEPOSITO(0, 10),

    // Lancamentos
    TIPO_LANCAMENTO(1, 1),
    VALOR_LANCAMENTO(0, 10),
    DATA_LANCAMENTO(10, 10),

    // Emprestimo
    VALOR_CONTRATADO(0,10),
    PRAZO(0,2);
```

StringUtilsEnum – Enum que contém o atríbuto com String vazia, para completar os espaços que sobram do layout dos arquivos

```
BLANK(" ");
```

TipoContaCorrenteEnum – Enum que contém os tipos de Conta Corrente: conjunta e simples.

```
CONJUNTA(1), SIMPLES(2);
```

TipoLancamentoEnum – Enum que contém os tipos de lançamentos que podem ser realizados.

```
TARIFA("T"),
CREDITO("C"),
DEBITO("D");
```

As classes do pacote de Enums, pussuem construtor que recebe os atributos da classe e métodos Getter e Setters.

4.3. com.telegrambotbank.exception

Classes:
ArquivoInvalidoException
CampoInvalidoException
ContaOuAgenciaInvalidaException
GravarArquivoDependenteException
SaldoInsuficienteException
ValorInvalidoException

As classes do pacote de exception, possuem a seguinte estrutura

```
	private String message = “...”;

	public Classe(String message) {
		super();
		this.message = message;
	}

	public Classe() {
		super();
	}

	public String getMessage() {
		return message;
	}

	public void setMessage(String message) {
		this.message = message;
	}
```

A String que possui a mensagem de erro da classe;
Um Construtor;
Um Construtor recebe a String como parâmetro;
O método de Get da mensagem da exception
O método de Set da mensagem da exception

4.4. com.telegrambotbank.file.helper

Classes:
ArquivoContaCorrenteReaderHelper
ArquivoContaCorrenteWriterHelper

ArquivoContaCorrenteReaderHelper – Classe responsável por efetuar a leitura do arquivo de conta corrente.

Métodos:

Construtor que recebe o Path do arquivo a ser manipulado
```
public ArquivoContaCorrenteReaderHelper(Path destino)
```

Método que faz a leitura de um arquivo enviado como parâmetro e seta as informações pertinentes a ele, retorna FileVisitResult com status CONTINUE para manter o arquivo acessível à leitura.
```
public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) throws
IOException
```

Método que exibe falha ao tentar acessar um arquivo.
Retorna FileVisitResult com status TERMINATE para indicar que o acesso a aquele arquivo foi encerrado.
```
public FileVisitResult visitFileFailed(Path file, IOException exc) throws
IOException
```

Método que verifica se o arquivo enviado como paramêtro existe no caminho.
Retorna true caso o arquivo exista e false se o arquivo não for encontrado.
```
public boolean isArquivoExistente(Path file) throws ArquivoInvalidoException
```

A classe também possui os getters e setters referentes aos atributos Path destino e String dadosArquvos, respectivamente.
ArquivoContaCorrenteWriterHelper - Classe responsável por efetuar a escrita no arquivo de conta corrente

Métodos:

Método responsável por alterar o conteúdo de uma linha de um arquivo recebido como parâmetro
```
public void alteraLinha(String dadoAntigo, String dadoNovo, String arquivo)
                    throws IOException, ArquivoInvalidoException
```

Método que grava um novo arquivo texto no caminho enviado no parametro `String caminho,` utilizando os atributos do VO, com a estrutura definada no layout enviado
```
public void gravarNovoArquivo(DependenteVO dependente, String caminho,
StringBuffer layout)throws IOException, GravarArquivoDependenteException
```

Método que registra no arquivo de lançamentos, os dados do lançamento efetuado. Recebe como parametro o VO a ser inserido, o caminho do arquivo e seu layout.

```
public void inserirLinha(LancamentoVO lancamento, Path caminho, StringBuffer layout) throws GravarArquivoDependenteException
```

4.5. com.telegrambotbank.file.util

Classes:
ArquivoContaCorrenteUtil

ArquivoContaCorrenteUtil – Classe que possui métodos auxiliares para leitura e escrita dos arquivos referentes a conta corrente.

Métodos:

Método que obtém o caminho de um arquivo texto, utilizando os parâmetros de nuContaBancaria e agenciaBancaria para pesquisa.

```
public static Path obterCaminhoArquivo(String nuContaBancaria, String agenciaBancaria)
```

Método que obtém o caminho de um arquivo texto de Dependentes, utilizando os parâmetros de nuContaBancaria e agenciaBancaria para pesquisa.

```
public static Path obterCaminhoArquivoDependentes(String nuContaBancaria, String agenciaBancaria)
```

Método que retorna uma mensagem de sucesso na obtenção do arquivo de texto desejado.

```
public static String obterMensagemSucesso(String string)
```

Método que obtém o caminho de um arquivo texto de Lancamentos, utilizando os parâmetros de nuContaBancaria e agenciaBancaria para pesquisa.

```
public static Path obterCaminhoArquivoLancamentos(String contaBancaria, String agenciaBancaria)
```

4.6. com.telegrambotbank.main

Classes:
Main

Método Main, que tem como função, fazer toda comunicação com o usuário, captando as mensagens que o cliente envia, direcionando cada funcionalidade para as classes que vão fazer a manipulação de seus dados, e enviando as respostas, conforme andamento da solicitação.

4.7. com.telegrambotbank.messages

Classes:
GeneralMessages

Classe que possui as mensagens gerais que são utilizadas no sistema, como mensagem de Boas Vindas, de ajuda e de tarifas.

4.8. com.telegrambotbank.opcoes.helper

Classes:
DependenteHelper
DepositoBancarioHelper
EmprestimoHelper
ExibirInformacoesContaHelper

DependenteHelper - Classe que recebe as informações referentes a dependente.
Métodos:

Método que solicita ao usuário, o nome do dependente, também faz a validação do tamanho do nome informado

```
public static String solicitarNomeDependente(TelegramBot bot, Update update)
throws CampoInvalidoException
```

Método que solicita o CPF do dependente, também faz a validação do tamanho do CPF informado

```
public static String solicitarCPFDependente(TelegramBot bot, Update update,
DependenteVO dependente)
```

DepositoBancarioHelper – Classe que solicita informações de conta, agência e valor para realizar depósito bancário, e também faz validações para garantir que o depósito seja realizado com sucesso.

Métodos:

Seta atributos da classe de Deposito com informações recebidas de Conta Depositante, Conta de Destino e valor do depósito.

```
public static DepositoVO montarDadosDepositoBancario(ContaBancariaVO
contaDepositante, ContaBancariaVO contaDestino,BigDecimal valorDeposito)
```

```
Faz validação do valor de depósito, retorna exception personalizada
public static DepositoVO validarValorDeposito(DepositoVO dadosDeposito) throws
ContaOuAgenciaInvalidaException
```

Seta atributos de uma operação de Débito

```
public static LancamentoVO montarDadosOperacaoDebito(DepositoVO dadosDeposito)
```

Seta atributos de uma operação de Crédito

```
public static LancamentoVO montarDadosOperacaoCredito(DepositoVO dadosDeposito)
```

Solicita o número da conta na qual o valor será depositado, e faz a validação do tamanho da conta informada.

```
public static String solicitarNuContaDestinoDeposito(TelegramBot bot, Update
update, BigDecimal valorDeposito)
```

Solicita o número da agência na qual o valor será depositado, e faz a validação do tamanho da agência informada.

```
public static String solicitarNuAgenciaDestinoDeposito(TelegramBot bot, Update
update)
```

Solicita o valor informado que será depositado, e faz a validação do tamanho do compo.

```
public static BigDecimal solicitarValorInformadoDeposito(TelegramBot bot, Update
update)
```

Verifica se o arquivo correspondente a conta e agência informada, existe no sistema

```
    public static void validarArquivoAgenciaEContaDeposito(LancamentoVO
dadosOperacaoDebito,LancamentoVO dadosOperacaoCredito)
```

EmprestimoHelper – Essa classe tem como função, verificar o valor de empréstimo disponível do cliente, solicitar o número de parcelas e fazer o cálculo completo do empréstimo levando em consideração taxas e juros.

Métodos:

Método que faz o cálculo do valor em que poderá ser feito o empréstimo, e solicita para o cliente, o valor que o mesmo deseja.

```
public static BigDecimal valorEmprestimoDisponivel(TelegramBot bot, Update
update, BigDecimal saldo) throws CampoInvalidoException
```

Método que solicita o número de parcelas que o cliente deseja pagar o empréstimo e faz a validação do número de parcelas de acordo com o layout do arquivo.

```
public static Integer prazoEmprestimo(TelegramBot bot, Update update) throws
CampoInvalidoException
```

Faz o calulo do empréstimo, utilizando as váriaveis de taxa e juros, e informa para o cliente.

```
public static BigDecimal calculaEmprestimo(TelegramBot bot, Update update,
EmprestimoVO vo) throws CampoInvalidoException
```

ExibirInformacoesContaHelper -

```
public static String buscarDadosConta(LancamentoVO dadosOperacao) throws
ArquivoInvalidoException, ContaOuAgenciaInvalidaException
```

4.9. com.telegrambotbank.opcoes.mediator

Classes:

OpcoesMediator

Métodos:

Método responsável por realiazar o depósito na conta, o parâmetro contaCorrenteDepositante possui os dados do cliente que solicitou o depósito, o parâmetro contaCorrenteDestino contém os dados da conta que receberá o depósito, o valor depositado está no parâmetro: valorDeposito.

```
public String depositar(ContaBancariaVO contaCorrenteDepositante,
ContaBancariaVO contaCorrenteDestino, BigDecimal valorDeposito)
                  throws SaldoInsuficienteException,
ContaOuAgenciaInvalidaException, IOException, ArquivoInvalidoException
```

```
Método que faz o cadastro de um depentende a partir dos dados que foram
informados pelo cliente
public String cadastrarDependente(DependenteVO dependente, LancamentoVO
dadosOperacao) throws IOException, GravarArquivoDependenteException
```

Método que busca o arquivo correspondente a conta informada, e mostra as informações relevantes para o cliente.

```
public String exibirInformacoesConta(LancamentoVO dadosOperacao) throws
ArquivoInvalidoException, ContaOuAgenciaInvalidaException
```

4.1.1. com.telegrambotbank.opcoes.util

Classes:

ClienteUtils

Utils

ClienteUtils – Classe que faz a comunicação com o cliente para obter as informações do mesmo.

Métodos:

Método responsável por solicitar e armazenar o nome do cliente.

```
public static String solicitarNomeCliente(TelegramBot bot, Update update)
```

Método responsável por solicitar e armazenar o cpf do cliente

```
public static String solicitarCpfCliente(TelegramBot bot, Update update)
```

Método responsável por solicitar e armazenar a data de nascimento do cliente

```
public static String solicitarDtNascCliente(TelegramBot bot, Update update)
```

Método responsável por solicitar e armazenar o email do cliente

```
public static String solicitarEmailCliente(TelegramBot bot, Update update)
```

Utils – Possui funcionalidades que serão utilizadas por diversas classes
Métodos:

Método que faz a validação do tamanho do dado que foi informado pelo cliente, se o tamanho da inforamçao, não corresponder ao tamanho informado pelos parâmetros, uma mensagem de erro será mostrada para o cliente.

```
public static void validarTamanhoMensagem(String resp, int positionsMin, int positionsMax)throws CampoInvalidoException
```

Método que faz a conversão de uma data em String para um LocalDate

```
public static LocalDate converteData(String data)
```

Retorna o valor fixo da agência.

```
public static String agencia()
```

Gera um número de conta para o cliente.

```
public static String gerarContaCorrente()
```

Método que retorna o número de espaços em branco informado pelo parâmetro nuPosicoesDesejadas

```
public static StringBuffer completarBlanks(int nuPosicoesDesejadas)
```

```
public Object getObjectByLine(String line, Object obj) throws
IllegalArgumentException, IllegalAccessException
```

4.1.2. com.telegrambotbank.services

Pacote que contém as Interfaces do sistema

Classes:

IContaCorrenteService
IDepositoService
ILancamentoServices

IContaCorrenteService – interface que contém os métodos referentes a operações em uma conta corrente de um cliente.

Credita uma conta bancária.

```
public String creditarContaBancaria(LancamentoVO dadosOperacao) throws
SaldoInsuficienteException, IOException, ArquivoInvalidoException,
GravarArquivoDependenteException;
```

Debita uma conta bancária.

```
public String debitarContaBancaria(LancamentoVO dadosOperacao) throws
SaldoInsuficienteException, IOException, ArquivoInvalidoException,
GravarArquivoDependenteException;
```

IdepositoService – interface que contém métodos referente a operações de depósito.

Método responsável por fazer um depósito entre contas.

```
public String depositar(DepositoVO dadosDeposito) throws
SaldoInsuficienteException, ContaOuAgenciaInvalidaException, IOException,
ArquivoInvalidoException;
```

IlancamentoServices - interface que contém métodos referente lançamento.

Método responsável por gravar informações do lançamento de um arquivo.

```
public String gravarLancamentoArquivo(LancamentoVO dadosOperacao) throws
GravarArquivoDependenteException;
```

4.1.3. com.telegrambotbank.services.impl

Classes:

ContaCorrenteServiceImpl
DependenteServicesImpl
DepositoServiceImpl
LancamentoServicesImpl

ContaCorrenteServiceImpl – Classe responsável por realizar operações de crédito e débito na conta corrente do cliente.

Métodos:

Obtém o saldo do cliente após identificar o arquivo do mesmo, faz o cálculo do novo saldo, atualiza o saldo da conta corrente, mostra mensagem de sucesso e grava o lançamento atualizado.

```
public String creditarContaBancaria(LancamentoVO dadosOperacao) throws
SaldoInsuficienteException, IOException, ArquivoInvalidoException,
GravarArquivoDependenteException
```

```
public String debitarContaBancaria(LancamentoVO dadosOperacao) throws
SaldoInsuficienteException, IOException, ArquivoInvalidoException,
GravarArquivoDependenteException
```

DependenteServicesImpl – Classe responsável por realizar o cadastro de depententes

Métodos:

Grava o dependente no arquivo e mostra mensagem de sucesso após realizar operação

```
public String cadastrarDependente(DependenteVO dependente, LancamentoVO
dadosOperacao) throws IOException, GravarArquivoDependenteException
```

DepositoServiceImpl – Classe que contém métodos referente a operação de depósito

Métodos:

Verifica se a conta é valida, usa o serviço de conta corrente para efetuar o débito da conta depositante, efetua o crédito na conta corrente destino e por fim mostra mensagem de sucesso da operação

```
public String depositar(DepositoVO dadosDeposito)throws
SaldoInsuficienteException, ContaOuAgenciaInvalidaException, IOException,
ArquivoInvalidoException
```

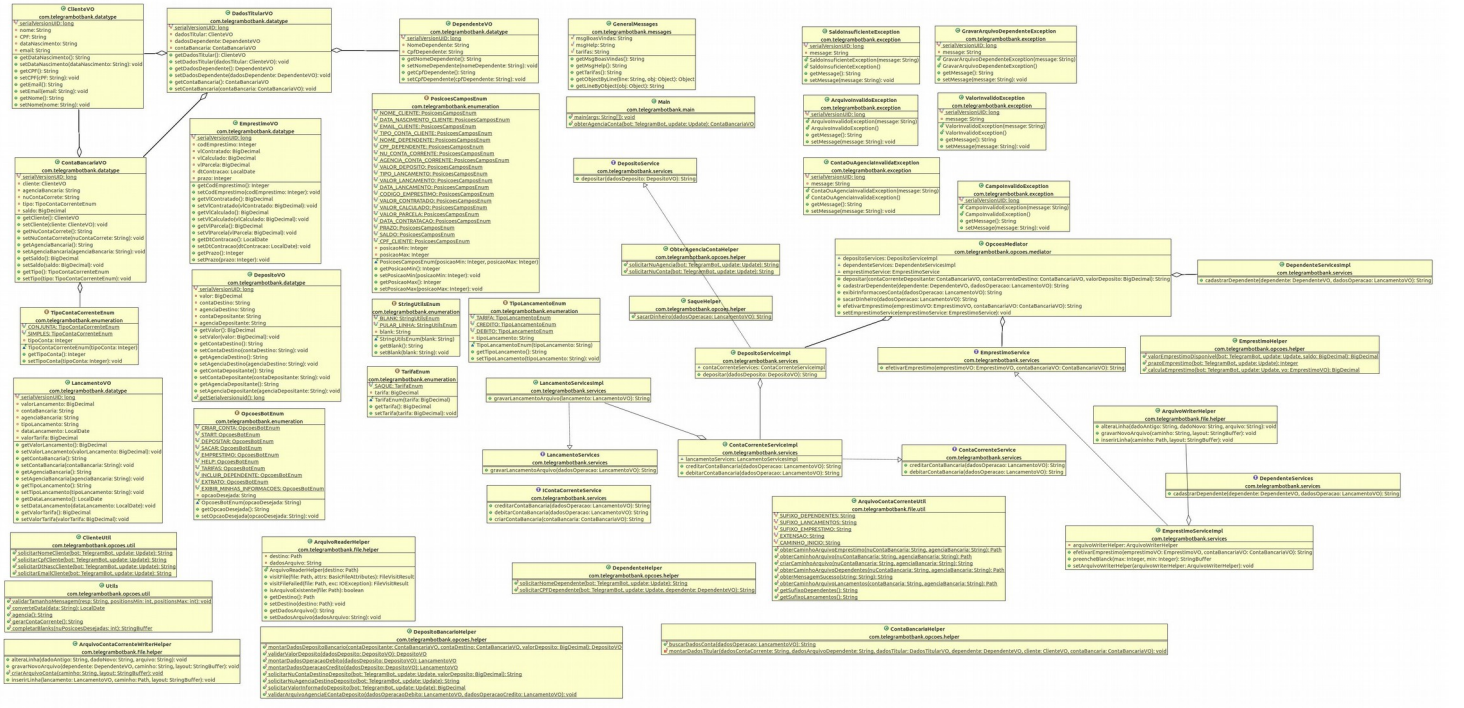
LancamentoServicesImpl – Classe responsável por realizar operações de lançamentos

Métodos:

Grava a operação de lançamento no arquivo, utilazando o layout de lançamento.

```
public String gravarLancamentoArquivo(LancamentoVO lancamento) throws GravarArquivoDependenteException
```

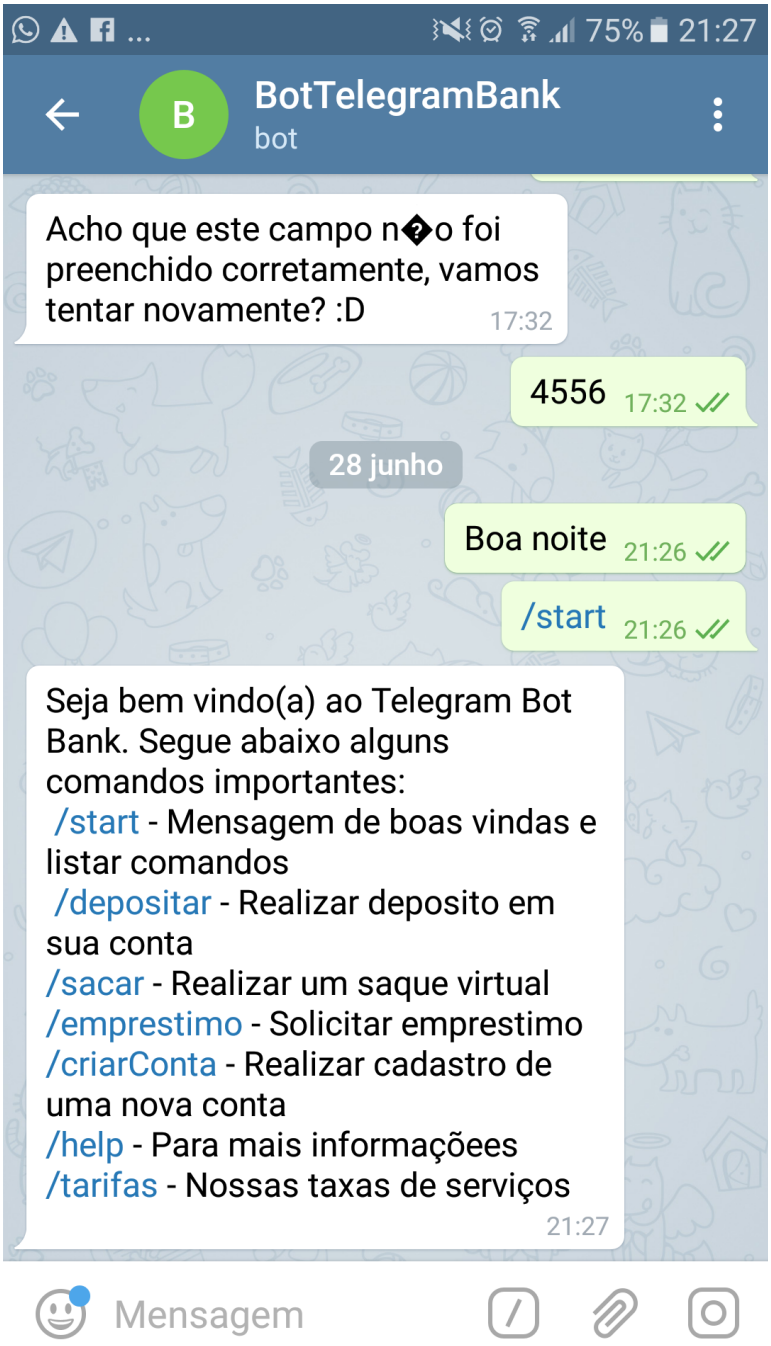
5. DIAGRAMA DE CLASSES



6. DIAGRAMAS DE SEQUÊNCIA

7.CAPTURA DE TELAS

7.1 Menu inicial do bot



7.2. Operação de Depósito

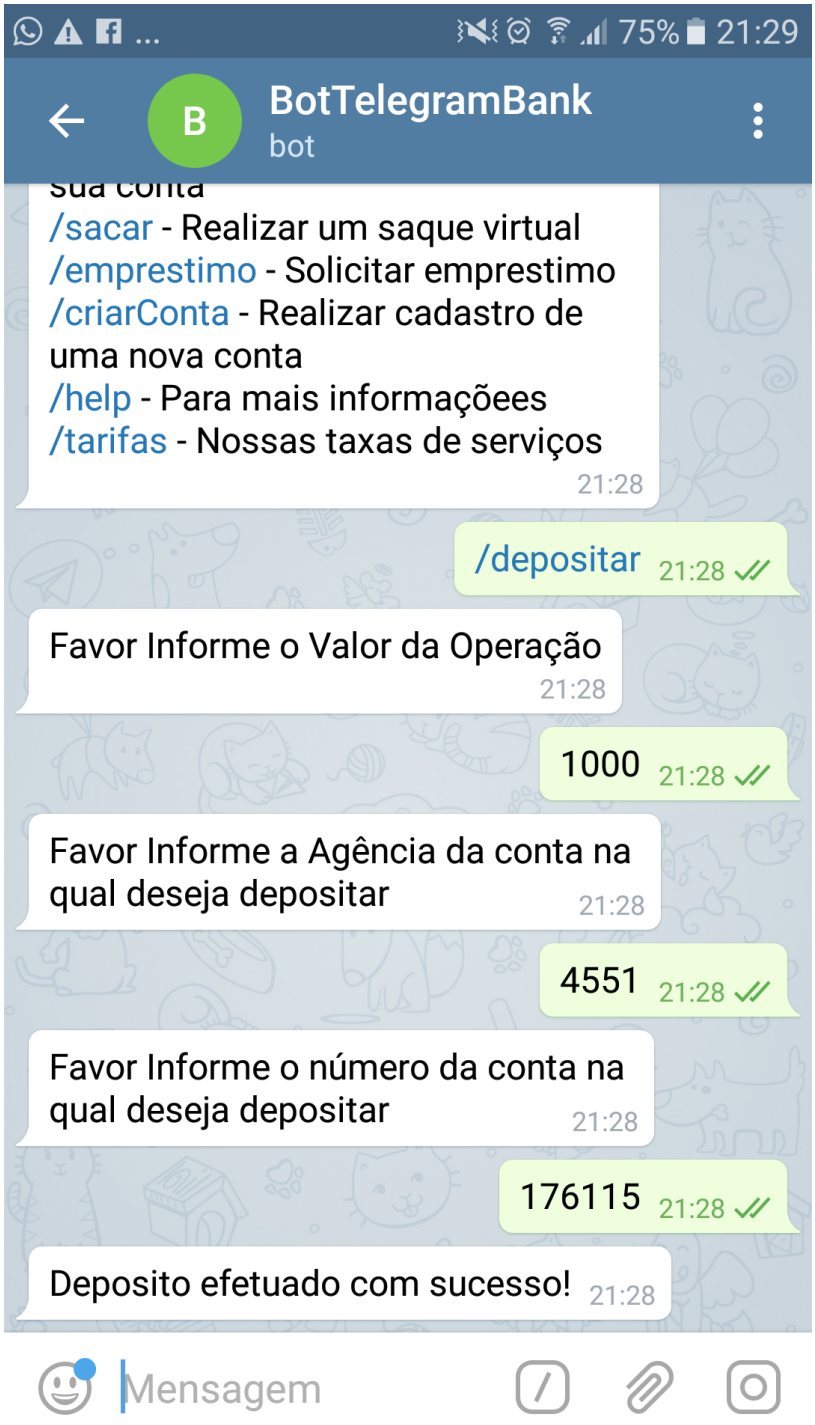
1- O cliente solicita a opção de depósito: /depositar;

2- O cliente informa o valor da operação;

3- O cliente informa a agência da conta;

4- O cliente informa a conta em que será realizado o depósito;

5- O sistema realiza o depósito, e retorna a mensagem de sucesso.



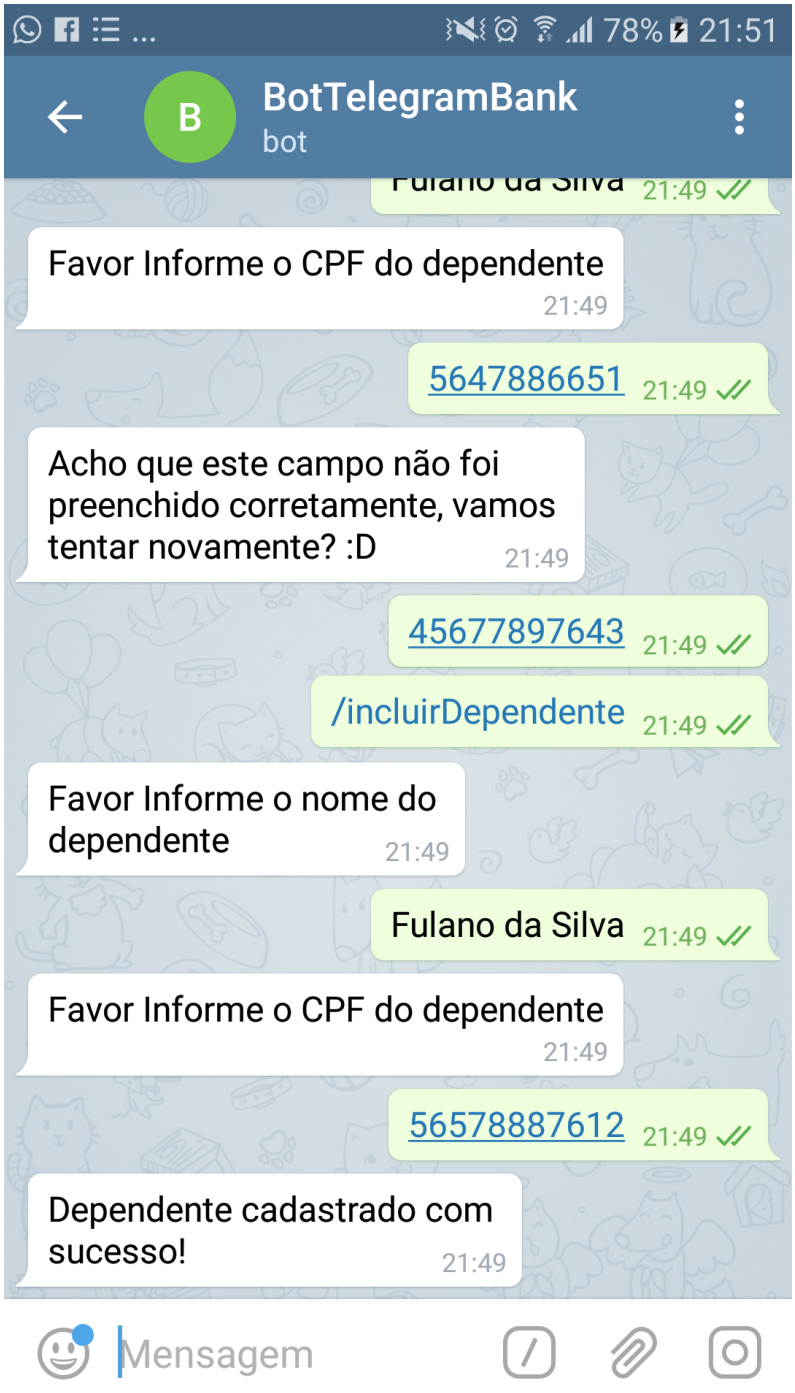
7.3. Incluir Dependente

1- O cliente solicita a opção de Inclusão de Dependente: /incluirDependente;

2- O cliente informa o nome do dependente;

3- O cliente informa o CPF do dependente;

4- O sistema realiza a inclusão do dependente, e retorna mensagem de sucesso.



7.4. Criar Conta

1- O cliente solicita a opção de Criação de Conta: /criarConta;

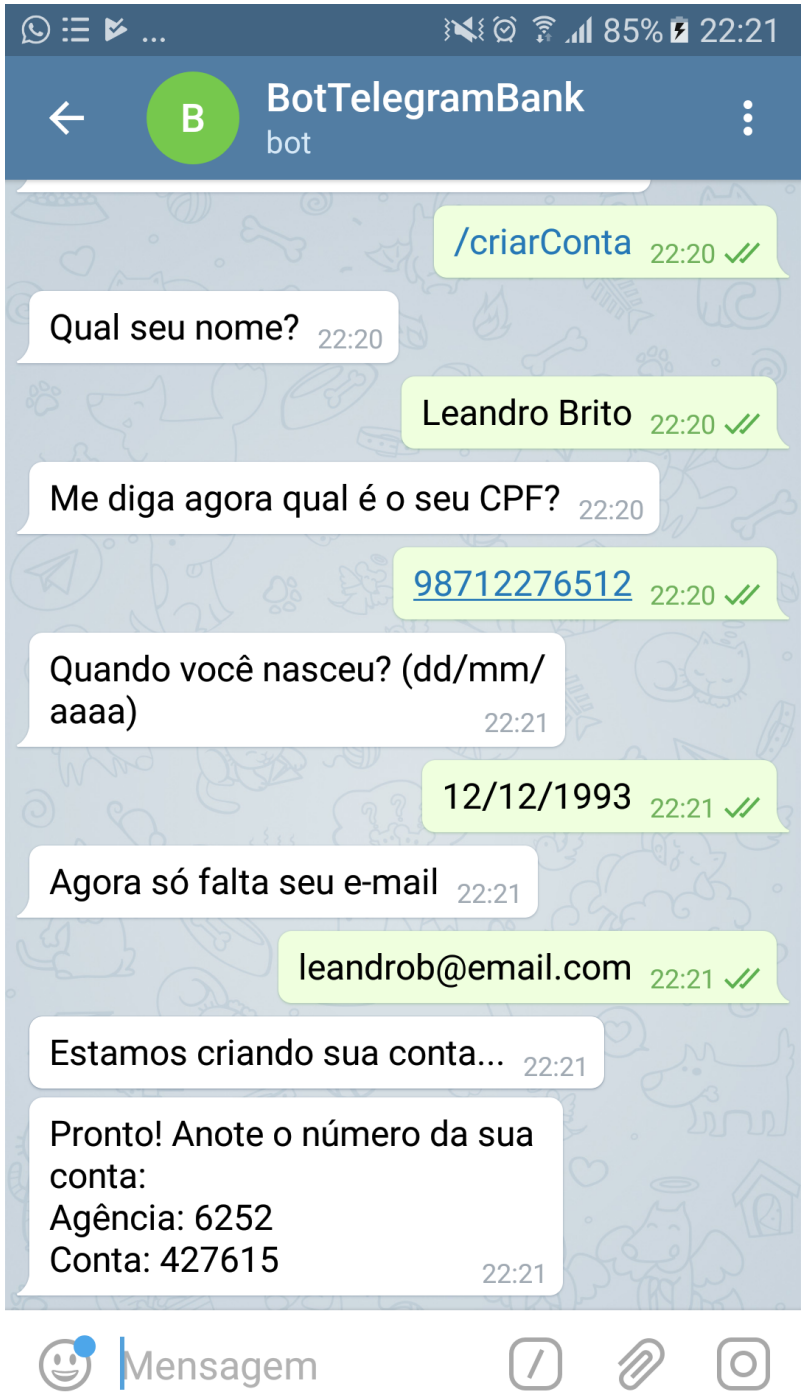
2- O cliente informa o Nome

3- O cliente informa o CPF

4- O cliente informa a Data de Nascimento

5- O cliente informa o e-mail

6- O sistema faz o cadastro da conta, informa mensagem de sucesso mostrando para o cliente, sua conta e agência.



7.4. Tarifas

1- O cliente solicita a opção de Tarifas: /tarifas;