



---

VC-X Solutions

---

Matheus Ferreira Santos

3 de junho de 2023

# Sumário

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introdução</b>  | <b>3</b>  |
| 1.1      | Sobre . . . . .  | 3         |
| 1.2      | Requisitos . . . . .                                     | 3         |
| 1.3      | Ferramentas Utilizadas . . . . .                         | 3         |
| 1.4      | Material de Referencia . . . . .                         | 3         |
| 1.5      | Overview do problema . . . . .                           | 3         |
| <b>2</b> | <b>Classes do sistema</b>                                | <b>4</b>  |
| 2.1      | Dominio . . . . .  | 4         |
| 2.2      | Services . . . . .                                       | 4         |
| 2.3      | Formatter . . . . .                                      | 5         |
| 2.4      | Manager . . . . .  | 5         |
| <b>3</b> | <b>Diagrama UML</b>                                      | <b>6</b>  |
| <b>4</b> | <b>Métodos Principais</b>                                | <b>7</b>  |
| 4.1      | Classe DocumentManager                                   |           |
|          | 7  |           |
| 4.1.1    | Método toText - VOID . . . . .                           | 7         |
| 4.1.2    | Método getPdfContentToList - List(String) . . . . .      | 8         |
| 4.2      | Classe AccountService . . . . .                          | 9         |
| 4.2.1    | Método getAccountData - Account . . . . .                | 9         |
| 4.3      | Classe BankStatementService . . . . .                    | 10        |
| 4.3.1    | Método getStatementIdentificationData - String . . . . . | 10        |
| 4.4      | PaymentService . . . . .                                 | 11        |
| 4.4.1    | Método getPaymentData - Payment . . . . .                | 11        |
| 4.4.2    | Método getPaymentDate - Date . . . . .                   | 12        |
| 4.5      | Classe StringFormatter . . . . .                         | 13        |
| 4.5.1    | Método getDateFormatted - Date . . . . .                 | 13        |
| <b>5</b> | <b>Telas e Funcionalidades</b>                           | <b>14</b> |
| 5.1      | Tela Principal . . . . .                                 | 14        |
| 5.2      | Funcionalidades  |           |
|          | 15   |           |
| 5.2.1    | Account Informations . . . . .                           | 15        |
| 5.2.2    | Payment Informations . . . . .                           | 15        |
| 5.2.3    | Bank Statement Identification . . . . .                  | 16        |
| 5.2.4    | Operation Date . . . . .                                 | 16        |
| 5.2.5    | All Informations . . . . .                               | 17        |
| <b>6</b> | <b>Como testar o sistema?</b>                            | <b>18</b> |

# **1 Introdução**

## **1.1 Sobre**

Projeto técnico para a vaga de Desenvolvedor Back-End (Estágio) na empresa VC-X Solutions.

## **1.2 Requisitos**

Para esse teste, foi exigido a utilização da linguagem de programação Java.

## **1.3 Ferramentas Utilizadas**

Além da linguagem de programação Java, requisito obrigatório para o teste, foi utilizado também as seguintes ferramentas ao longo desse teste:

- Apache PDFBox (Para a leitura do PDF);
- Git e GitHub (Para controle de versão do teste);
- Java Swing (Para construção das telas interativas);
- Eclipse (Construção de todo Back-End);
- NetBeans (Construção da GUI);
- Overleaf (Construção dessa Documentação);
- Linux Ubuntu 22.04 LTS (O.S.).

## **1.4 Material de Referencia**

Ao longo do meu estudo na linguagem Java sempre utilizei, por recomendação do meu amigo e professor Marcos Monteiro, o livro "Java Ensino Didático" do autor Sergio Furgeri, que me deu grande base sobre o básico de cada estrutura de dados e classe da linguagem e principalmente da biblioteca de interfaces gráficas Java Swing.

## **1.5 Overview do problema**

Dado um arquivo no formato PDF, recupere as seguintes informações:

- Identificação do extrato;
- Dados da conta debitada;
- Dados do pagamento;
- Data da operação efetuada.

## 2 Classes do sistema

Nas tabelas abaixo estão as classes que foram desenvolvidas nesse sistema, e ao lado, resumidamente, suas respectivas funções. (Obs.: Classes da lógica do sistema, Back-End)

### 2.1 Dominio

| Classe         | Função   |
|----------------|--|
| Account        | Serve para modelar os objetos do tipo conta que irá aparecer no PDF, e no sistema consequentemente.                    |
| Payment        | Serve para modelar os objetos do tipo pagamento que irá aparecer no PDF, e no sistema consequentemente.                |
| PaymentVoucher | Serve para modelar os objetos do tipo comprovante de pagamento que irá aparecer no PDF, e no sistema consequentemente. |
| Document       | Serve para modelar o arquivo PDF conforme as necessidades do sistema.  |

### 2.2 Services

| Classe               | Função  |
|----------------------|---|
| AccountService       | Classe de Serviço do sistema que, de maneira geral, serve para recuperar dados, vindos do PDF, que estão diretamente ligados com informações de conta bancária.   |
| BankStatementService | Classe de Serviço do sistema que, de maneira geral, serve para recuperar dados, vindos do PDF, que estão diretamente ligados com informações do extrato bancário. |
| PaymentService       | Classe de Serviço do sistema que, de maneira geral, serve para recuperar dados, vindos do PDF, que estão diretamente ligados com informações de pagamento.        |

Continuação...

## 2.3 Formatter

| Classe          | Função  |
|-----------------|---|
| StringFormatter | Classe de, como o nome já diz, formatação de String, de maneira geral, trabalhará com tudo relacionado a formatação dos valores String do sistema, caso seja necessário formatação. |

## 2.4 Manager

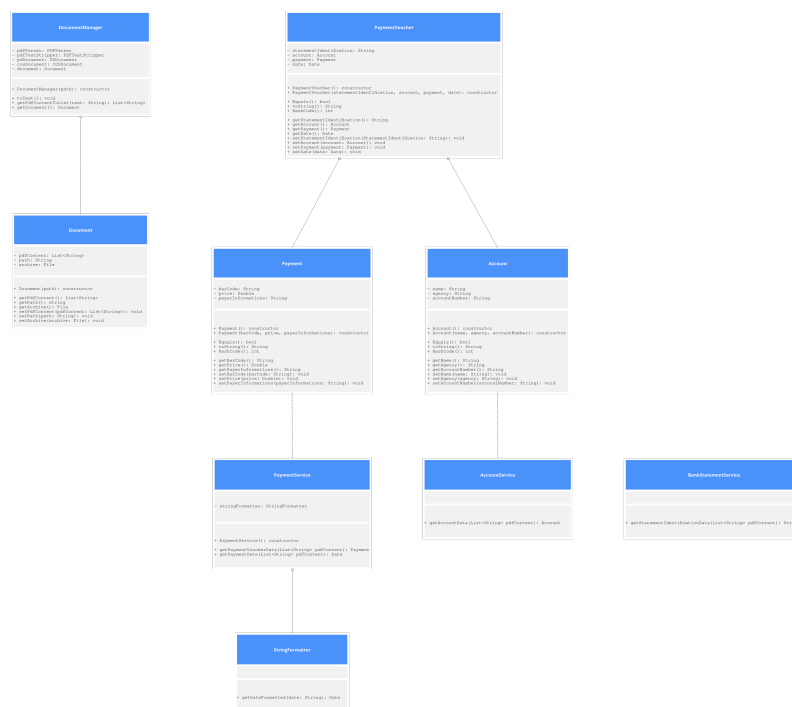
| Classe          | Função   |
|-----------------|--|
| DocumentManager | Classe que, como o nome já diz, gerenciamento do documento PDF, de maneira geral, irá fazer o carregamento do arquivo PDF para ser possível utilizá-lo em uma alguma parte do sistema. |

### 3 Diagrama UML

Para esse teste, desenvolvi um diagrama UML focado totalmente na especificação do sistema, tentando assim, deixar o projeto de mais fácil compreensão e semanticamente mais correto.

Obs.: Essa UML está voltada apenas para do Back-End e ignora totalmente a classe do Java Swing e Controller, já que, o NetBeans gera muito código no seu "GUI Maker" e as Controllers são minúsculas, porém, a regra é simples, clicou no botão, chama a devida Controller, que despacha para o método para a devida Service e apresenta o dado na tela, caso tudo estiver nos conformes.

A seguir está o diagrama UML citado anteriormente:



<https://cacao.com/diagrams/mGM0a837coHIbD2B/8A4F2>

## 4 Métodos Principais

Obs.: Simbolo de colchete deitado remete a um espaço vazio.

### 4.1 Classe DocumentManager

#### 4.1.1 Método toText - VOID

Linha 1 a 4 - Cria uma nova instancia de PDF Parser e passa o arquivo já instanciado na classe Document e o tipo do modo, que sera "r" de "Read" ou "Leitura/Ler".

Linha 7 - Passa para a variável "cosDocument" o que vier do método "getDocument" da classe "COSParser";

Linha 10 - Passa a variável "pdDocument" uma nova instância da classe PDDocument chamando o metodo Construtor da classe passando a variável "cosDocument" como parametro do construtor.

Linha 12 e 13 - Informa para "pdfTextStripper" quais são as páginas iniciais e finais do PDF, para que ele possa ler todas elas sem pular nenhuma.

Linha 15 a 21 - Passa para o atributo "pdfContent", que é uma lista, o que vier do método, que está na mesma classe deste que estou apresentando agora (DocumentManager), "getPdfContentToList".

```
1  this.pdfParser = new PDFParser(new RandomAccessFile(
2      this.document.getArchive(), "r"
3  )
4  );
5
6  this.pdfParser.parse();
7  this.cosDocument = pdfParser.getDocument();
8
9  this.pdfTextStripper = new PDFTextStripper();
10 this.pdDocument = new PDDocument(this.cosDocument);
11
12 pdfTextStripper.setStartPage(0);
13 pdfTextStripper.setEndPage(this.pdDocument.getNumberOfPages());
14
15 this.document.setPdfContent(
16     this.getPdfContentToList(
17         this.pdfTextStripper.getText(
18             this.pdDocument
19         )
20     )
21 );
```

#### 4.1.2 Método getPdfContentToList - List(String)

Linha 3 - Instancia-se a classe "StringTokenizer", que, a grosso modo, serve basicamente para quebrar uma String em tokens.

Linha 5 a 8 - Itera essa variável do tipo "StringTokenizer", até ele não encontrar um próximo elemento, separando cada linha exclusivamente e "jogando" dentro da lista pdfContent, logo após isso, o retorna, para ser possível utilizar essa lista futuramente.

```
1 List<String> pdfContent = new ArrayList<String>();  
2  
3 StringTokenizer stringTokenizer = new StringTokenizer(text, "\n");  
4  
5 while(stringTokenizer.hasMoreElements())  
6     pdfContent.add(stringTokenizer.nextToken());  
7  
8 return pdfContent;
```



## 4.2 Classe AccountService

### 4.2.1 Método getAccountData - Account

Linha 5 - Instancia-se uma nova lista contendo todos os "campos" que uma conta tem, seguindo o modelo do PDF fornecido, tornando assim a busca mais fácil, já que, todos os campos informados estão presente no PDF, afinal, tive o PDF como base para fazer esse desafio.

Linhas 7 a 10, 13 a 15 e 19 a 21 - São IFs que verificam se dentro do pdfContent, lista de cada linha do PDF, existe alguma linha com os campos informados pela lista "options"(sempre irá ter, a menos que troque o PDF por outro totalmente diferente).

Linhas 11 - Utiliza-se o metodo "replaceAll" passando como parametro o "campo" que está verificando no IF (nesse caso "Nome: ") e substitui isso por nada, ou seja, ele basicamente remove "Nome: " do PDF, deixando realmente só o dado bruto que está contido ao lado do campo, (O mesmo ocorre para as linhas 16 e 22 a 25, a única diferença é que na linha 16, utilizo o método substring(0, 4), assumindo que uma agência tem 4 números, tornando assim mais seguro e na 22 a 25 trato a String para remover os espaços em branco que veio junto do PDF).

```
1 Account newAccount = new Account();
2 Integer occurrenceIndex = 0;
3 String content = "";
4
5 List<String> options = Arrays.asList("Nome:_", "Agencia:_", "Conta:_");
6
7 for(String lineContent : pdfContent) {
8     if(lineContent.toLowerCase().contains(
9         options.get(0).toLowerCase()
10    ))
11         newAccount.setName(lineContent.replaceAll(options.get(0), ""));
12
13     if(lineContent.toLowerCase().contains(
14         options.get(1).toLowerCase()
15    ))
16         newAccount.setAgency(lineContent.replaceAll(options.get(1), "")
17             .substring(0, 4));
18
19     if(lineContent.toLowerCase().contains(
20         options.get(2).toLowerCase()
21    )) {
22         occurrenceIndex = lineContent.indexOf(options.get(2));
23         content = lineContent.replaceAll(options.get(2), "");
24         newAccount.setAccountNumber((content.substring(
25             occurrenceIndex, content.length()).replaceAll("_", "")));
26     }
27 }
28
29 return newAccount;
```

## 4.3 Classe BankStatementService

### 4.3.1 Método getStatementIdentificationData - String

Esse método faz basicamente o mesmo que o método anterior, levando em consideração o tratamento de uma String na lista pdfContent. Pode-se perceber que, grande parte dos métodos, estão se repetindo novamente, como os métodos: "replaceAll()", "indexOf()" e "substring()", a única parte que seria "nova" é a questão do "FirstEmptySpacesIndex", que, a grosso modo, guarda o índice da primeira sequência de espaços vazios na linha, tornando assim, mais fácil de formatar os dados que vieram do PDF.

```
1 String option = "Identificacao_no_extrato: ";
2 String bankStatementIdentification = "";
3 Integer firstEmptySpacesIndex = 0;
4
5 for (String lineContent : pdfContent) {
6     if (lineContent.toLowerCase().contains(option.toLowerCase())) {
7         bankStatementIdentification = lineContent
8             .replaceAll(option, "");
9
10        if (bankStatementIdentification.contains(" "))
11            firstEmptySpacesIndex = bankStatementIdentification
12                .indexOf(" ");
13
14        if (firstEmptySpacesIndex != 0)
15            return bankStatementIdentification
16                .substring(0, firstEmptySpacesIndex);
17
18        return bankStatementIdentification;
19    }
20 }
21
22 return bankStatementIdentification;
```

## 4.4 PaymentService

Obs.: Como ambos dos métodos utilizam outros métodos já citados anteriormente, só explanarei sobre o que for novo no código.

### 4.4.1 Método getPaymentData - Payment

Linha 20 - Utilizo o replaceAll para trocar a vírgula por um ponto, para que, assim, se faça possível a conversão do dado para Double.

Linha 22 - Utilizo o método Double.parseDouble() para transformar o valor "110.0" para 110.0D.

```
1 Payment payment = new Payment();
2 List<String> options = Arrays.asList("Codigo_de_barras:",
3                                     "Valor_do_documento:",
4                                     "pagador:");
5
6 String aux = "";
7 Integer firstEmptySpacesIndex = 0;
8
9 for(String lineContent : pdfContent) {
10     if(lineContent.toLowerCase().contains(options.get(0).toLowerCase()))
11         payment.setBarCode((lineContent.replaceAll(options.get(0), ""))
12                             .replaceAll("_", ""));
13
14
15     if(lineContent.toLowerCase().contains(options.get(1)
16                                           .toLowerCase())) {
17
18         aux = lineContent.replaceAll(options.get(1), "");
19         aux = aux.substring(3, aux.length());
20         aux = aux.replaceAll(",", ".");
21
22         payment.setPrice(Double.parseDouble(aux));
23     }
24
25     if(lineContent.toLowerCase().contains(options.get(2)
26                                           .toLowerCase())) {
27
28         aux = lineContent.replaceAll(options.get(2), "");
29
30         if(aux.contains("_"))
31             firstEmptySpacesIndex = aux.indexOf("_");
32
33         if(firstEmptySpacesIndex != 0)
34             payment.setPayerInformations(aux.substring(
35                                           0, firstEmptySpacesIndex
36                                           ));
37         else
38             payment.setPayerInformations(aux);
39     }
40 }
41
42 return payment;
```

#### 4.4.2 Método `getPaymentDate - Date`

Nesse método abaixo, eu queria pegar não só a data, mas sim a data e a hora da operação, então, da linha 13 a 27 é toda a lógica para recuperar a data "dd/MM/yyyy" da operação bancária realizada e da linha 29 a 33 é toda a lógica para recuperar a hora da transação "HH:mm:ss"(Obs.: "HH" porque está no padrão 24 horas).

Linha 35 e 41 - Retorna o que vier de resposta do método "getDateFormatted" da classe "StringFormatter", citada algumas páginas atrás, na tabela de classes do sistema.

```
1 String option = "Operacao_efetuada_em_";
2
3 Integer firstEmptySpacesIndex = 0;
4 Integer secoundEmptySpacesIndex = 0;
5
6 String date = "";
7 String hours = "";
8
9 String content = "";
10
11 for(String lineContent : pdfContent) {
12     if(lineContent.toLowerCase().contains(option.toLowerCase())) {
13         content = lineContent.replaceAll(option, "");
14         option = "as";
15
16         if(content.contains(option))
17             content = content.replaceAll(option, "");
18
19         if(content.contains("_ _ _"))
20             firstEmptySpacesIndex = content.indexOf("_ _");
21
22         if(firstEmptySpacesIndex != 0) {
23             date = content.substring(0, firstEmptySpacesIndex);
24             content = content.substring(
25                 firstEmptySpacesIndex + 3, content.length()
26             );
27         }
28
29         if(content.contains("_"))
30             secoundEmptySpacesIndex = content.indexOf("_");
31
32         if(secoundEmptySpacesIndex != 0)
33             hours = content.substring(0, secoundEmptySpacesIndex);
34
35         return this.stringFormatter.getDateFormatted(
36             date + "_" + hours
37         );
38     }
39 }
40
41 return this.stringFormatter.getDateFormatted(date + "_" + hours);
```

## 4.5 Classe StringFormatter

### 4.5.1 Método getDateFormatted - Date

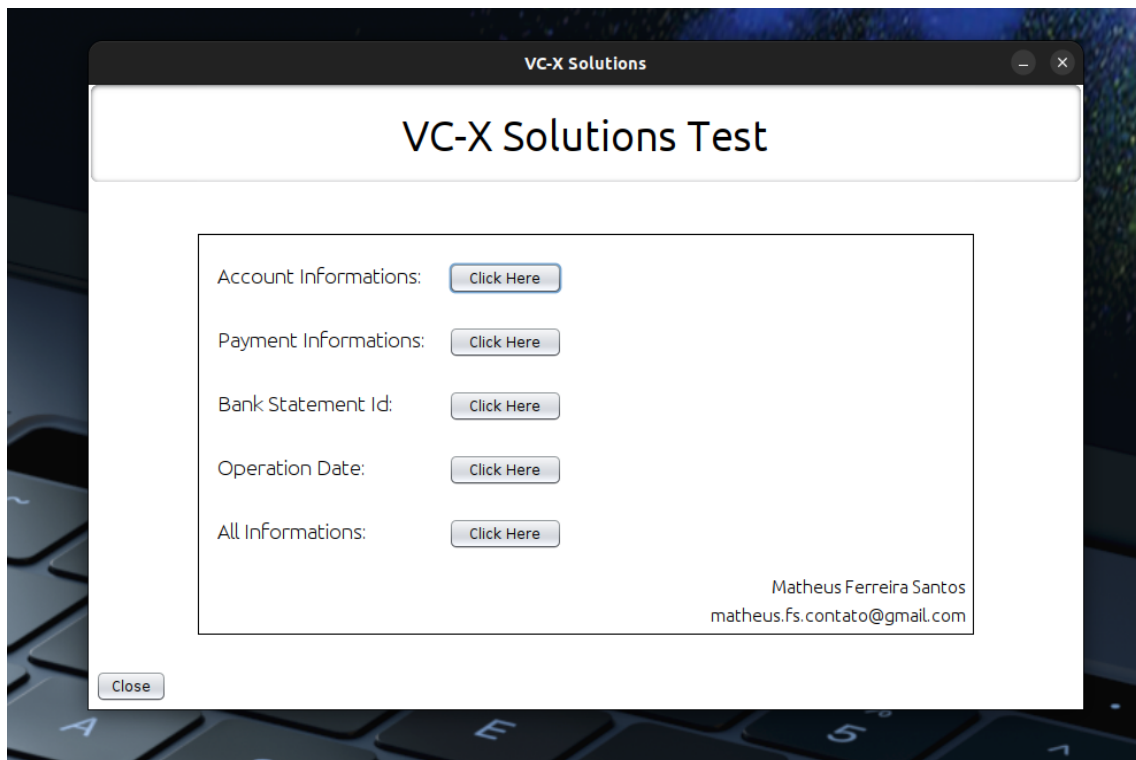
Linha 2 - Espera receber uma String, como parâmetro da função, e tenta retornar ela formatada para a classe Date, do pacote "java.util.Date", caso não consiga, imprime a mensagem de erro e retorna nulo.

```
1 try {  
2     return new SimpleDateFormat("dd/MM/yyyy_HH:mm:ss").parse(date);  
3 } catch (ParseException e) {  
4     System.out.println(e.getMessage());  
5     return null;  
6 }
```

## 5 Telas e Funcionalidades

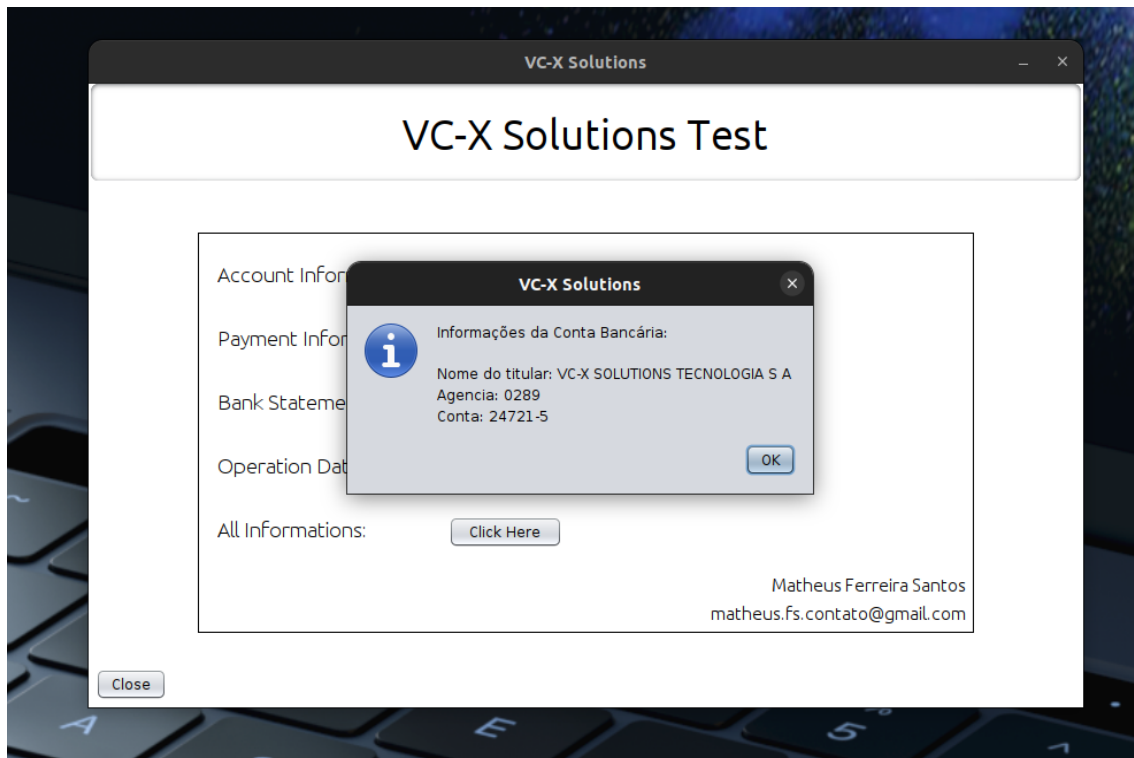
Para esse sistema, decidi utilizar a biblioteca Java Swing, utilizando o "GUI Maker" do NetBeans. Junto com a biblioteca JavaFX, o Java Swing é uma poderosa lib para quem deseja fazer interfaces gráficas utilizando a linguagem Java, com ela, é possível fazer de fato várias e várias aplicações, e para esse teste, era mais do que o suficiente para mim. A seguir imagens da tela principal do sistema e suas funcionalidades:

### 5.1 Tela Principal

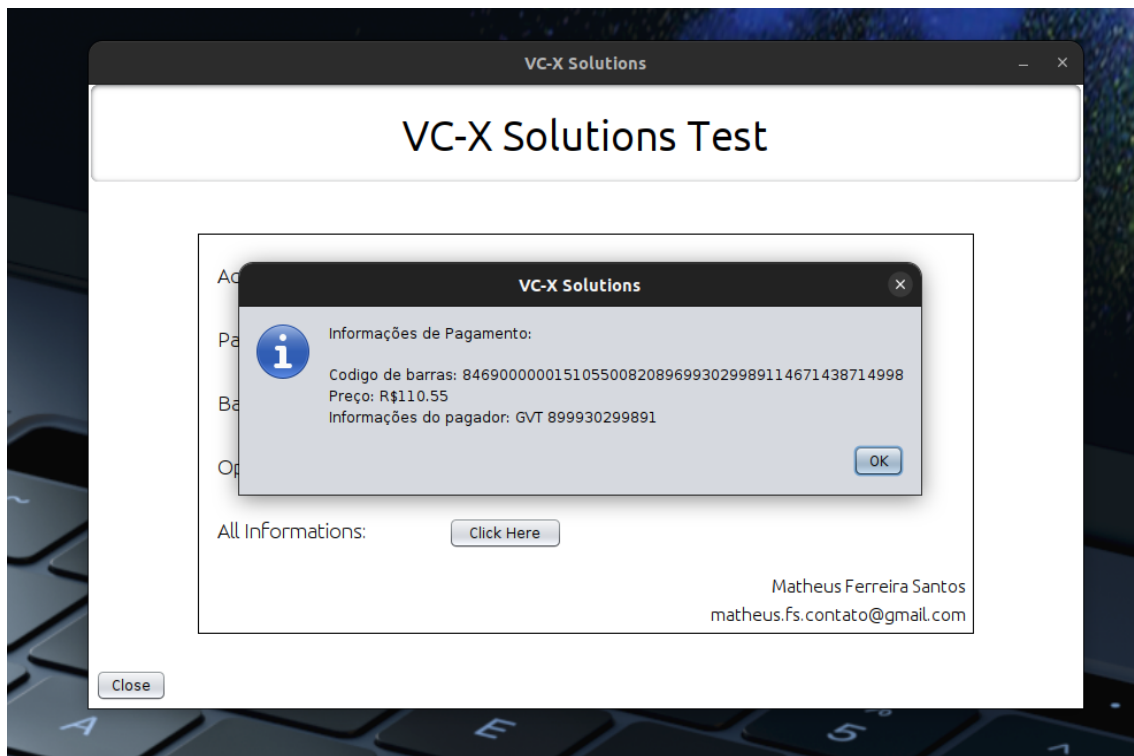


## 5.2 Funcionalidades

### 5.2.1 Account Informations

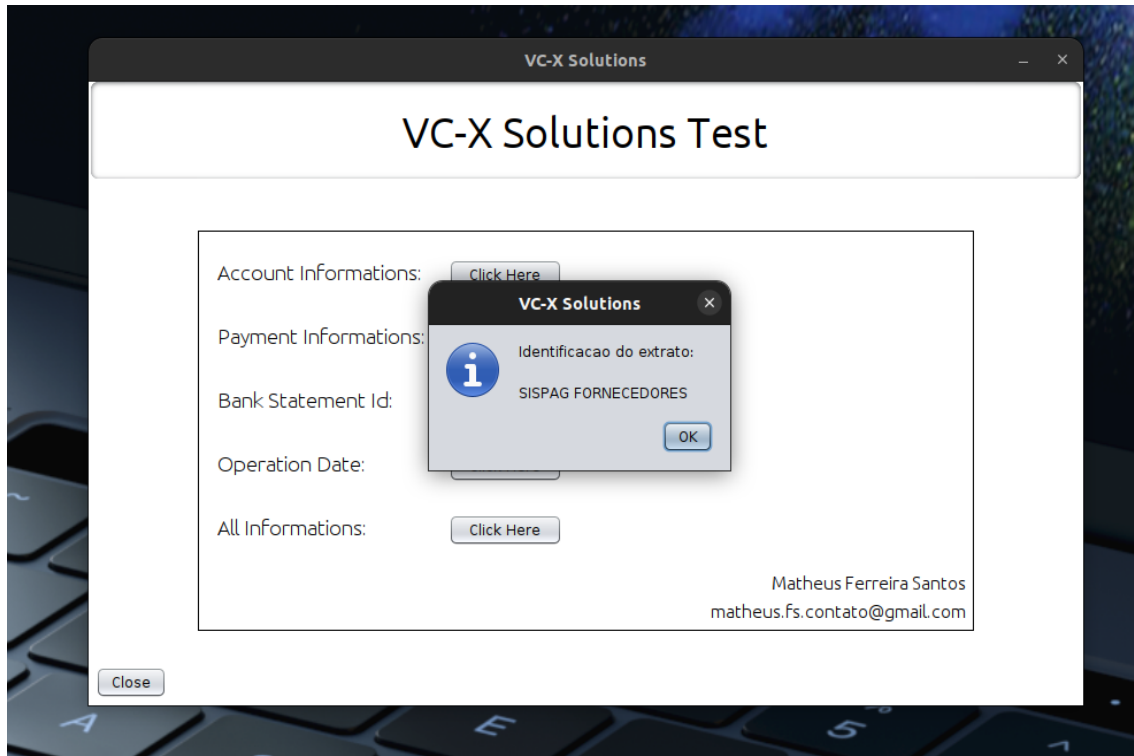


### 5.2.2 Payment Informations

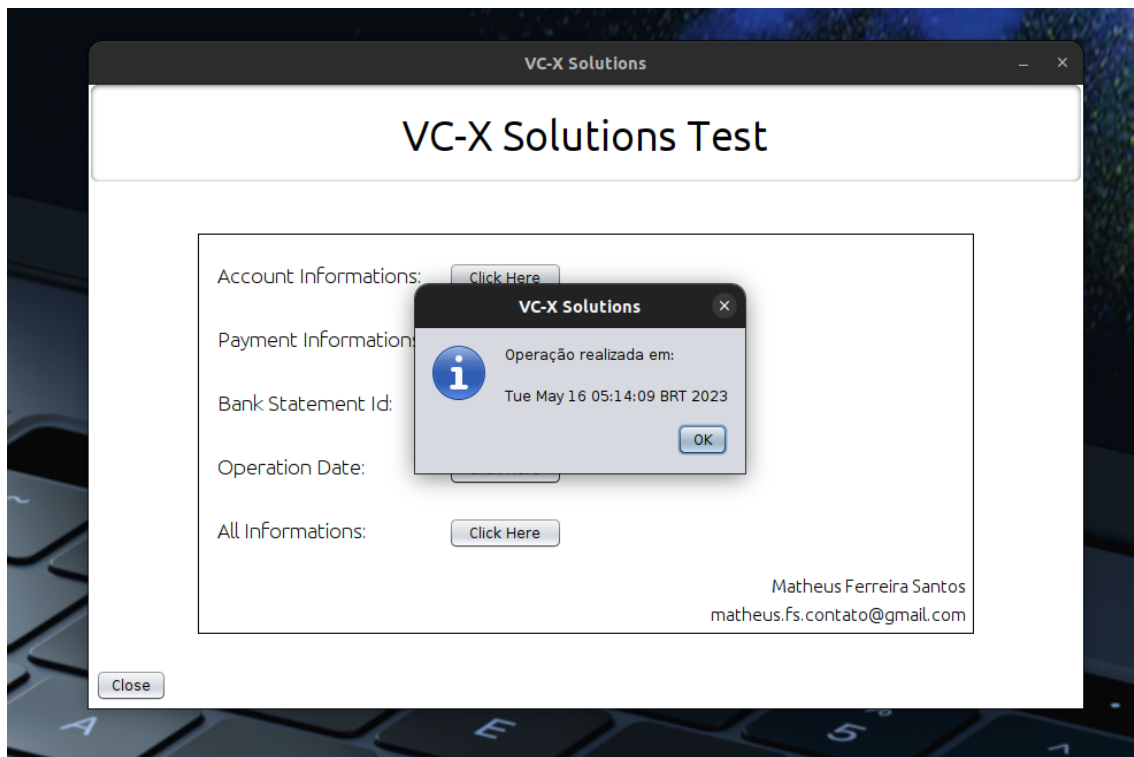


Continuação...

### 5.2.3 Bank Statement Identification



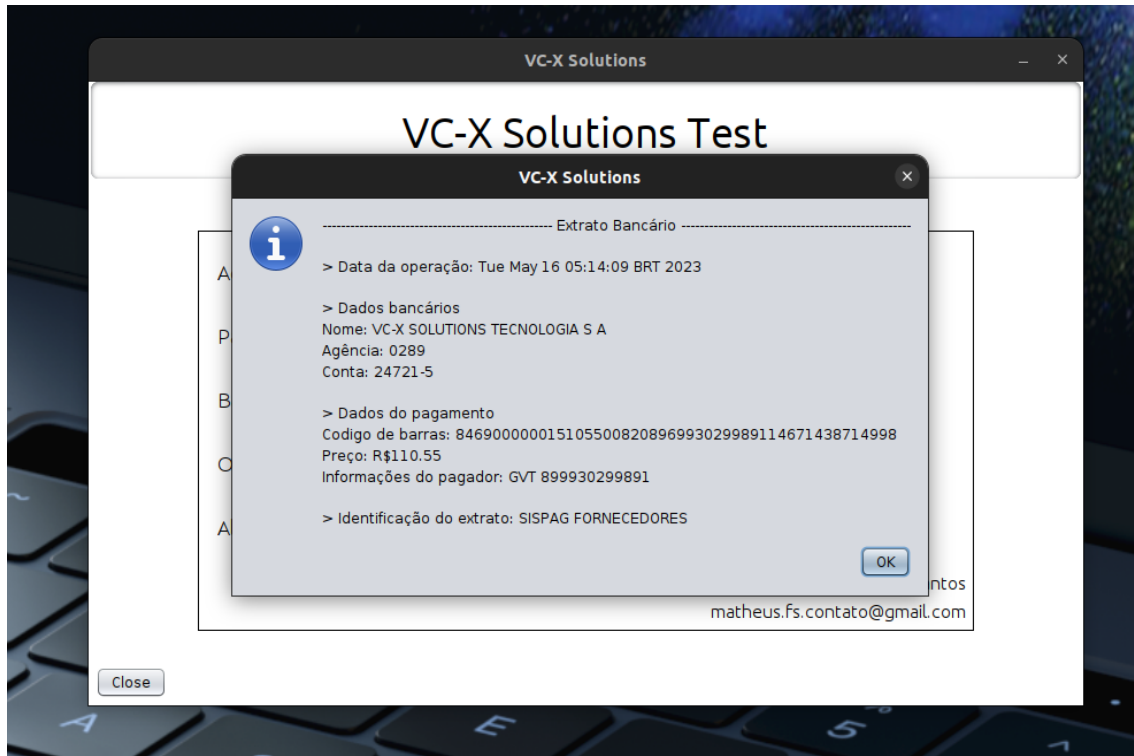
### 5.2.4 Operation Date





Continuação...

### 5.2.5 All Informations



## 6 Como testar o sistema?

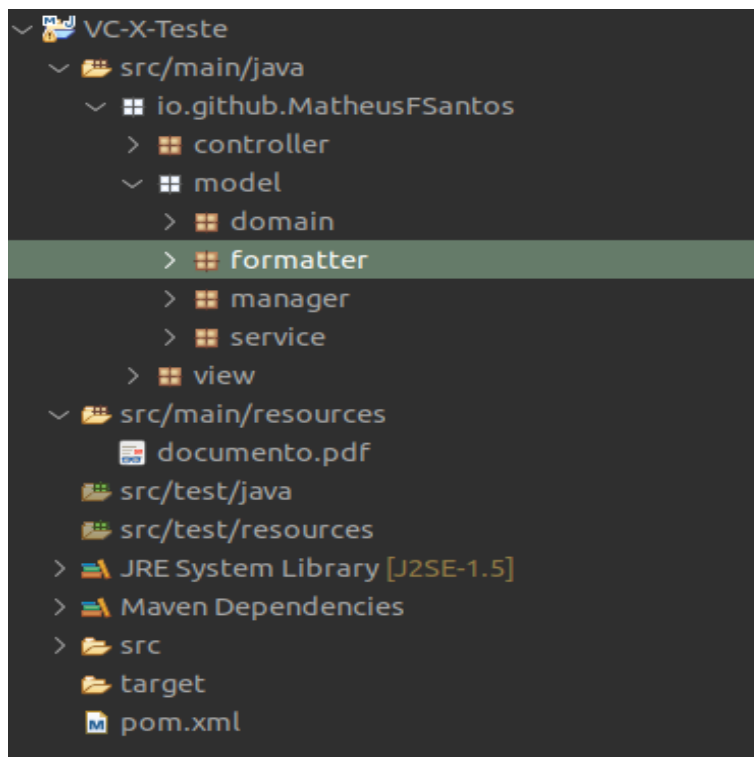
Para testar o sistema é muito simples, entre no GitHub: <https://github.com/Matheus-FSantos/vc-x-solutions> e clone o repositório no seu computador.

Caso não saiba, o código para clonar é: `git clone <url do repositório>.git`

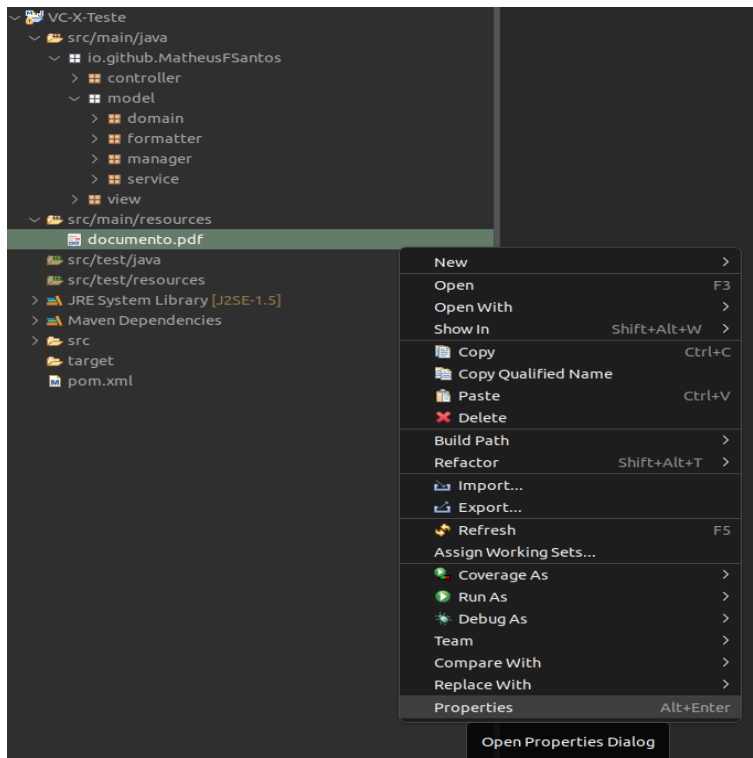
(Obs.: O projeto final está dentro da pasta Versão Final, obviamente).

Abra ele em uma IDE, de sua preferência, eu estarei utilizando o Eclipse IDE, porém, lembre-se que é um projeto Maven, e siga as seguintes instruções:

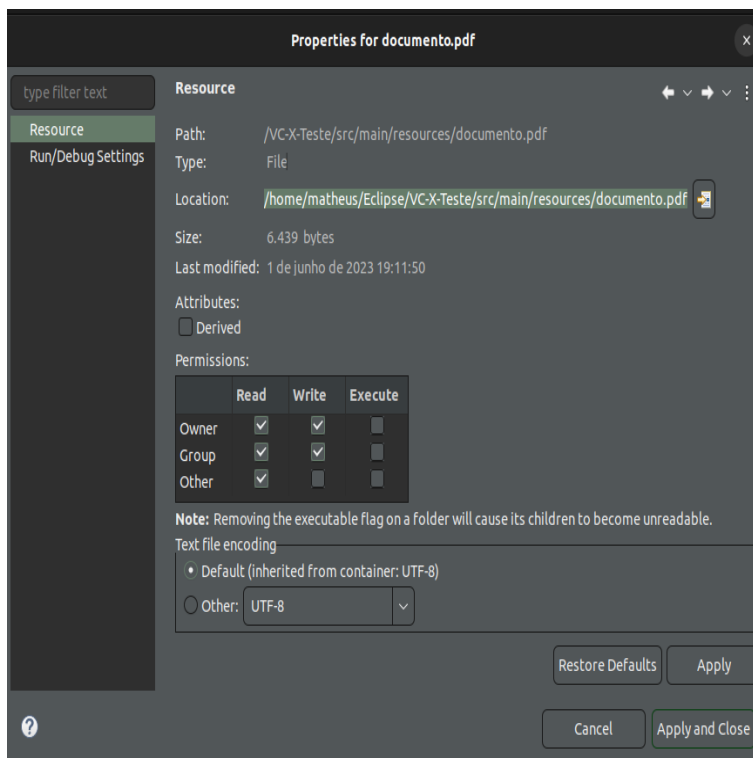
Abra a pasta "src/main/resources", dentro dela você visualizará o documento.pdf do teste.



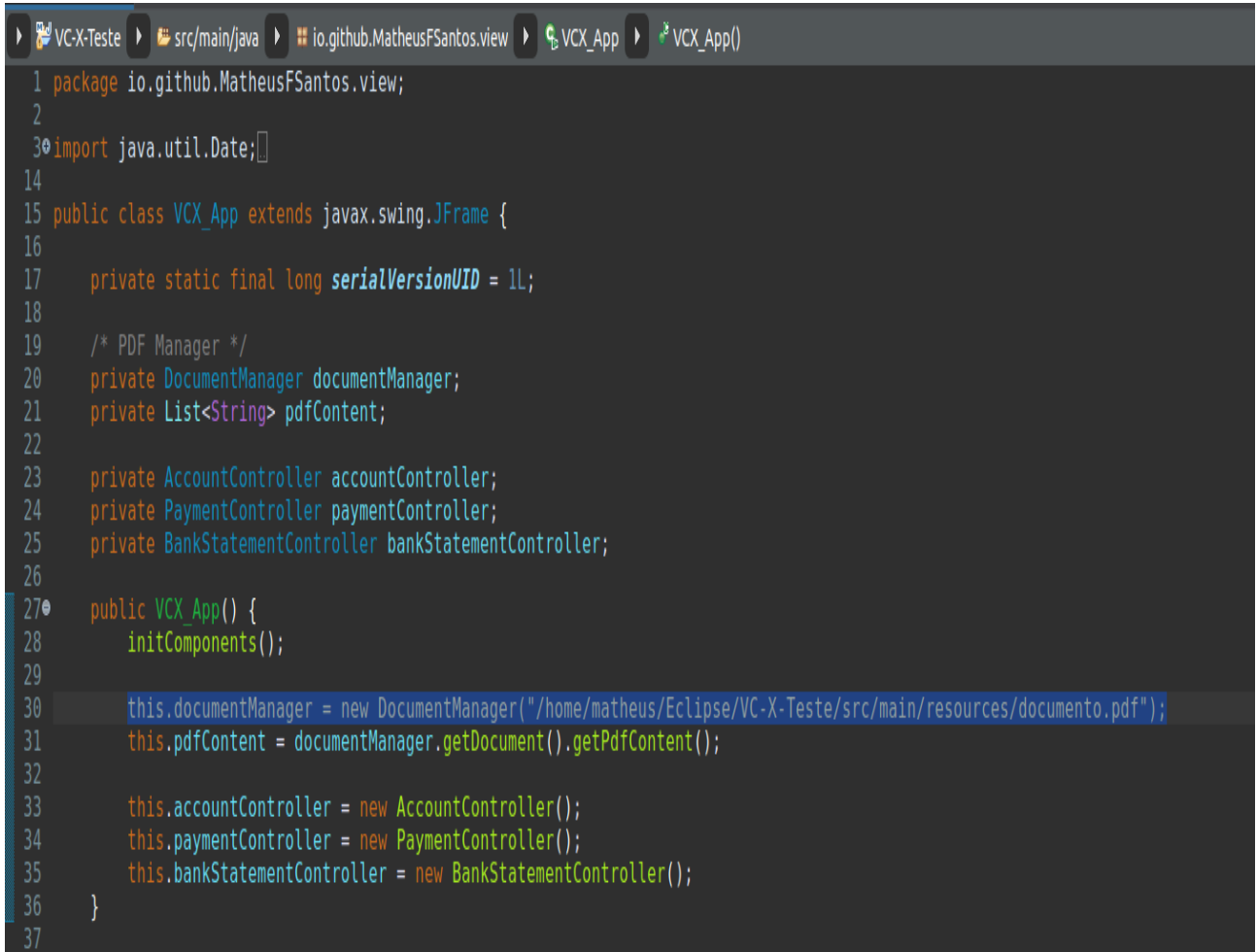
Clique com o botão direito do mouse/touchpad e vá em "Properties"



Copie o diretório que aparece e feche a janela.



Vá até o arquivo "VCXApp.java", contido no diretório: "src/main/java/io.github.MatheusFSantos/view/", e substitua o diretório padrão, que já está lá, pelo diretório que copiou anteriormente.



```
1 package io.github.MatheusFSantos.view;
2
3 import java.util.Date;
4
14
15 public class VCX_App extends javax.swing.JFrame {
16
17     private static final long serialVersionUID = 1L;
18
19     /* PDF Manager */
20     private DocumentManager documentManager;
21     private List<String> pdfContent;
22
23     private AccountController accountController;
24     private PaymentController paymentController;
25     private BankStatementController bankStatementController;
26
27     public VCX_App() {
28         initComponents();
29
30         this.documentManager = new DocumentManager("/home/matheus/Eclipse/VC-X-Teste/src/main/resources/documento.pdf");
31         this.pdfContent = documentManager.getDocument().getPdfContent();
32
33         this.accountController = new AccountController();
34         this.paymentController = new PaymentController();
35         this.bankStatementController = new BankStatementController();
36     }
37 }
```

Rode esta classe e pronto, está tudo funcionando!!!