Sistema Cliente Com Interface Gráfica Para Conexão FTP
Exercício Computacional I - Redes De Computadores

Rafael Gonçalves de Oliveria Vianal[1]

[1]Sistemas de Informação – Universidade Federal do Mato Grosso do Sul (UFMS)
Caixa Postal 79400-000 – Coxim – MS – Brazil

rafael.viana@aluno.ufms.br

Resumo. Este relatório descreve como foi implementado um sistema cliente com interface gráfica para conexão FTP, utilizando JavaFX em conjunto do Apache Commons Net 3.6, uma biblioteca de conexão FTP.

1. JavaFX

Foi escolhido o JavaFX para criar uma interface gráfica onde o usuário terá um melhor desempenho, ao utilizar o sistema. Para criar um sistema elegante foi utilizado uma biblioteca com novos elementos CSS, a bibliteca utilizada para essa finalidade foi a JFoenix, essa bibliteca é open source. Para icones foi utilizado a biblioteca fontawesomefx-8.9 essa biblioteca é open source.

2. Apache Commons Net 3.6

Para melhor desempenho nas conexões ftp, foi utilizada a biblioteca de conexão FTP da Apache Commons, onde a mesma se encontra atualmente na versão 3.6.

3. Problematica

O trabalho proposto tem como objetivo criar um client FTP, no qual tenha como Adicionar, Renomear, realizar Download e Upload de arquivos e pastas. Tendo como restrição o sistema deve apenas deixar criar 5 pastas e 2 arquivos por diretório, sendo que no máximo deve ser criados 3 níveis de diretórios.

A implementação de restrições no software cliente e não no software servidor, coloca a aplicação em risco, um usuário mal intencionado poderia utilizar outro software cliente para acessar os serviços do software servidor, já que o mesmo não possui restrições, assim o usuário mal intencionado passaria a ter privilégios.

Assim podemos observar uma falha na segurança dessas restrições, porém como essa aplicação e para fins acadêmicos, não focaremos nessa questão.

4. Metodologia

Como a aplicação é pequena a estrutura da mesma, foi dividida em 3 pacotes sendo eles: Cliente, Icons e Socket.

1. A pasta cliente é responsável pela parte de Interface Gráfica do Usuário, envolvendo controllers.
2. A pasta Icons armazena icons utilizados na pasta cliente.

3. A pasta socket e responsável por toda comunicação da Lib da Apache com os controllers do cliente.

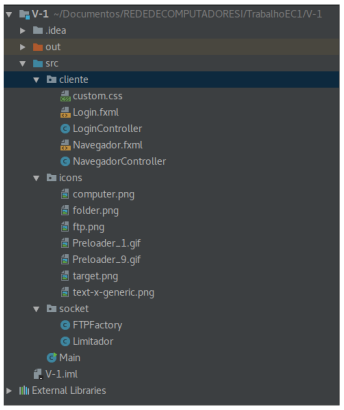

Figura 1. Imagem da Tela de Login.

Primeiramente a Class Main e invocada, chamando a Scene do Login.fxml, o controller do login e responsável por fornecer o necessario para o FTPClient poder fazer a conexão.

```
1
2  public class Main  extends Application{
3
4  @Override
5      public void start(Stage stage) throws Exception {
6      Parent root = FXMLLoader.load(getClasssingleton().
           getResource("/cliente/Login.fxml"));
7      Scene scene = new Scene(root);
8      stage.setScene(scene);
9      stage.show();
10 }
11
12
13 public static void main(String[] args) {
14     launch(args);
15
16 }
17
18 }
```

Figura 2. Imagem da Tela de Login.

Com a tela de login aberta o usuário entra com as informações login , senha, endereço do host, port do host mostrado no código abaixo.

```
1  private void login(ActionEvent event) {
2      btnLogin.setVisible(false);
3      imgProgress.setVisible(true);
4
5      PauseTransition pauseTransition = new PauseTransition();
6      pauseTransition.setDuration(Duration.seconds(3));
7      pauseTransition.setOnFinished(ev -> {
8
9  try {
10
11     int reply = FTPFactory.getInstance().FTPConecta(txtHostName.
           getText(),Integer.parseInt(txtHostPort.getText()),this.
           txtUsername.getText(),this.txtPassword.getText());
12     System.out.println("Igual:" + reply);
13
14     if (reply == 230) {
15
16         btnLogin.getScene().getWindow().hide();
17         completeLogin();
18
19     } else {
```

```
20
21          imgProgress.setVisible(false);
22          btnLogin.setVisible(true);
23          JOptionPane.showMessageDialog(null,"Erro Senha ou
                Usuário incorreto !!", "Erro ao Logar",JOptionPane.
                ERROR_MESSAGE);
24      }
25
26  } catch (IOException ex) {
27      Logger.getLogger(LoginController.class.getName()).log(Level.
            SEVERE, null, ex);
28  } catch (Exception ex) {
29      Logger.getLogger(LoginController.class.getName()).log(Level.
            SEVERE, null, ex);
30  }
31
32  });
33    pauseTransition.play();
34  }
35
36  private void completeLogin() throws IOException {
37
38      imgProgress.setVisible(false);
39      Stage dashboardStage = new Stage();
40      dashboardStage.setTitle("");
41      Parent root = FXMLLoader.load(getClass().getResource("
            Navegador.fxml"));
42      Scene scene = new Scene(root);
43      dashboardStage.setScene(scene);
44      dashboardStage.show();
45
46  }
```

Para poder realizar a comunicação entre as classes e o FTPClient da apache, foi criado uma classe singleton chamada de FTPFactory onde a mesma cria uma getInstance de FTPClient, podendo ser chamada de qualquer classe sem ter que que ser instanciada novamente, oque iria ocasionar a perda da conexão FTP.

```
1
2   public class FTPFactory {
3
4       private final FTPClient ftp;
5       private TreeItem<FTPFile> file;
6
7       private FTPFactory() {
8               this.ftp = new FTPClient();
9       }
10
11      public static FTPFactory getInstance() {
12
```

```
13          return FTPFactoryHolder.INSTANCE;
14      }
15
16
17      /**
18      * Classe privada que armazena a única instância de
           FTPFactory.
19      */
20      private static class FTPFactoryHolder {
21
22          private static final FTPFactory INSTANCE = new
               FTPFactory();
23      }
24
25
26      public FTPClient getFTP() {
27          return this.ftp;
28      }
29
30
31      public boolean Excluir(FTPFile file) {
32      try {
33              if (file.isDirectory()) {
34                  System.out.println(file.getLink());
35                  return ftp.removeDirectory(file.getLink());
36              } else {
37                  System.out.println(file.getLink());
38                  return ftp.deleteFile(file.getLink());
39              }
40      } catch (IOException e) {
41          e.printStackTrace();
42          }
43          return false;
44      }
45
46
47      public int FTPConecta(String host, int port, String user,
           String pwd) throws Exception {
48          int reply;
49          ftp.connect(host, port);
50          reply = ftp.getReplyCode();
51          if (!FTPReply.isPositiveCompletion(reply)) {
52              ftp.disconnect();
53              throw new Exception("Exception in connecting to FTP
                   Server");
54          }
55          ftp.login(user, pwd);
56          reply = ftp.getReplyCode();
57          ftp.setFileType(FTPClient.BINARY_FILE_TYPE);
```

```
58          ftp.enterLocalPassiveMode();
59          ftp.setAutodetectUTF8(true);
60          return reply;
61      }
62
63
64      public void disconnect() {
65          if (this.ftp.isConnected()) {
66          try {
67              this.ftp.logout();
68              this.ftp.disconnect();
69              } catch (IOException f) {
70
71              }
72          }
73      }
74  }
```

Após a Class login em conjunto com a Class FTPFactor, validar os dados de usuário a tela de navegação e aberta, essa tela nada mais é do que um conjunto de botões em uma Grid a esquerda e um TreeView do JavaFX no centro para poder navegar pela estutura dos diretórios.

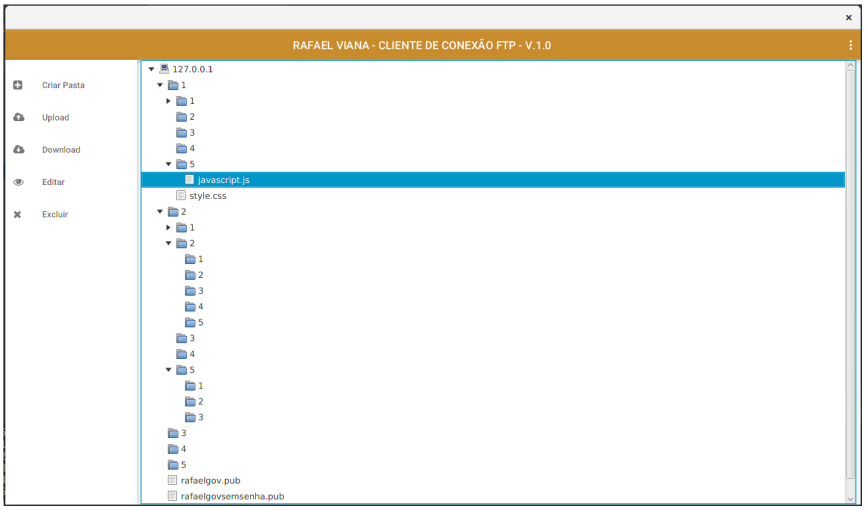

Figura 3. Imagem da Tela de Navegação.

Toda lógica de abastecimento da TreeView e a criação dos TreeItem utilizados na TreeView estão na Class navegação, onde inicia a criação dos TreeItem da toda a estrutura da arvore como ADICIONAR, REMOVER, LISTAR, EDITAR, DOWNLOAD e UPLOAD de arquivos/pastas, o método CarregarFiles(), cria recursivamente Nodes/TreeItem apartir dos files no servidor FTP, posteriormente encadeando os TreeItems e lançando-os na TreeView, facilitando assim a navegação.

```
1
```

```
2  private void Navegacao() throws IOException {
3      FTPFile files[];
4      TreeItem<FTPFile> treeRoot;
5      files = FTPFactory.getInstance().getFTP().listFiles();
6      Tree.setEditable(true);
7      if (files != null && files.length > 0) {
8          files[0].setRawListing(FTPFactory.getInstance().getFTP()
               .getPassiveHost());
9          treeRoot = CarregarFiles(files[0], true);
10     } else {
11         FTPFile file = new FTPFile();
12         file.setType(FTPFile.DIRECTORY_TYPE);
13         file.setLink(FTPFactory.getInstance().getFTP().
               printWorkingDirectory());
14         file.setRawListing(FTPFactory.getInstance().getFTP().
               getPassiveHost());
15         treeRoot = new TreeItem<>(file , new ImageView(computador
               ));
16     }
17 Tree.getSelectionModel().select(treeRoot);
18 Tree.setRoot(treeRoot);
19
20
21 btnBaixar.disableProperty().bind(Tree.getSelectionModel().
       selectedItemProperty().isNull()
22 .or(Tree.getSelectionModel().selectedItemProperty().isEqualTo(
       treeRoot)));
23
24
25 btnBaixar.setOnAction(e -> {
26 TreeItem<FTPFile> selected = Tree.getSelectionModel().
       getSelectedItem();
27 if (selected.getValue().isFile()) {
28 JFileChooser local = new JFileChooser();
29 local.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
30 local.setDialogTitle("Escolha um local para salvar");
31 local.setFileHidingEnabled(false);
32 int res = local.showSaveDialog(null);
33 if (res == JFileChooser.APPROVE_OPTION) {
34 String caminho = String.valueOf(local.getSelectedFile());
35 try {
36 FileOutputStream fos = new FileOutputStream(caminho);
37 if (FTPFactory.getInstance().getFTP().retrieveFile(selected.
       getValue().getLink(), fos)) {
38 JOptionPane.showMessageDialog(null, "Arquivo Baixado com Sucesso
       !" + "\n\n",
39 "Sucesso", JOptionPane.INFORMATION_MESSAGE);
40 }
41
```

```
42 } catch (FileNotFoundException e1) {
43 e1.printStackTrace();
44 } catch (Exception ex) {
45 JOptionPane.showMessageDialog(null, "Erro ao Baixado  Arquivo :
       " + ex.getMessage(), "Erro", JOptionPane.ERROR_MESSAGE);
46 }
47 }
48 } else {
49 JOptionPane.showMessageDialog(null, "Somente Arquivos!" + "\n\n"
       ,
50 "Erro", JOptionPane.ERROR_MESSAGE);
51 }
52 });
53
54
55 btnUp.setOnAction(e -> {
56 TreeItem<FTPFile> selected = Tree.getSelectionModel().
       getSelectedItem();
57 try {
58 FTPFactory.getInstance().getFTP().changeWorkingDirectory(
       selected.getValue().getLink());
59
60 if (limiteNivel() <= 5) {
61 if (limiteArquivo()) {
62 JFileChooser fc = new JFileChooser();
63 fc.setFileSelectionMode(JFileChooser.FILES_ONLY);
64 int result = fc.showOpenDialog(null);
65 if (result == JFileChooser.APPROVE_OPTION) {
66 File arquivo = fc.getSelectedFile();
67 InputStream isArquivo = null;
68 try {
69 isArquivo = new FileInputStream(arquivo.getAbsolutePath());
70 } catch (FileNotFoundException e1) {
71 e1.printStackTrace();
72 }
73 try {
74 FTPFile f = new FTPFile();
75
76 if (selected.getValue().isDirectory()) {
77 FTPFactory.getInstance().getFTP().changeWorkingDirectory(
       selected.getValue().getLink());
78 f.setLink(selected.getValue().getLink() + separador + arquivo.
       getName());
79 } else {
80 FTPFactory.getInstance().getFTP().changeWorkingDirectory(
       selected.getParent().getValue().getLink());
81 f.setLink(selected.getParent().getValue().getLink() + separador
       + arquivo.getName());
82 }
```

```
83
84  if (FTPFactory.getInstance().getFTP().storeFile(arquivo.getName
        (), isArquivo)) {
85
86  f.setName(arquivo.getName());
87  f.setRawListing(arquivo.getName());
88  f.setType(FTPFile.FILE_TYPE);
89  TreeItem<FTPFile> newItem = new TreeItem<FTPFile>(f, new
        ImageView(this.arquivo));
90  if (selected.getValue().isDirectory()) {
91  selected.getChildren().add(newItem);
92  } else {
93  selected.getParent().getChildren().add(newItem);
94  }
95
96  JOptionPane.showMessageDialog(null, "Arquivo Enviado!");
97  } else {
98  JOptionPane.showMessageDialog(null, "Arquivo não enviado!");
99  }
100
101
102 } catch (IOException e1) {
103 e1.printStackTrace();
104 }
105
106 } else {
107 JOptionPane.showMessageDialog(null, "Arquivo não selecionado!");
108 }
109 } else {
110 JOptionPane.showMessageDialog(null, "Apenas 2 arquivos por pasta
        ", "Limite Atingido", JOptionPane.WARNING_MESSAGE);
111 }
112 } else {
113 JOptionPane.showMessageDialog(null, "Apenas 3 níveis de
        Diretórios", "Limite Atingido", JOptionPane.WARNING_MESSAGE);
114
115 }
116 } catch (IOException e1) {
117 e1.printStackTrace();
118 }
119 });
120
121
122 btnEditar.setOnAction(e -> {
123 TreeItem<FTPFile> selected = Tree.getSelectionModel().
        getSelectedItem();
124 String novolink = "";
125 try {
126 FTPFactory.getInstance().getFTP().changeWorkingDirectory(
```

```
    selected.getParent().getValue().getLink());
127 novolink = FTPFactory.getInstance().getFTP().
    printWorkingDirectory();
128 } catch (IOException e1) {
129 e1.printStackTrace();
130 }
131 try {
132 String novoNome = JOptionPane.showInputDialog("Digite um novo
    nome para " + selected.getValue().getName());
133
134 if (FTPFactory.getInstance().getFTP().rename(selected.getValue()
    .getName(), novoNome)) {
135 novolink = novolink + separador + novoNome;
136 selected.getValue().setRawListing(novoNome);
137 selected.getValue().setLink(novolink);
138
139 RenameRecursivao(selected);
140
141 JOptionPane.showMessageDialog(null, " Renomeado ! ", "Rename",
    JOptionPane.INFORMATION_MESSAGE);
142
143 Tree.refresh();
144 } else {
145 JOptionPane.showMessageDialog(null, "Erro ao Renomear! ", "Erro
    Rename", JOptionPane.ERROR_MESSAGE);
146
147 }
148
149 } catch (IOException e1) {
150 e1.printStackTrace();
151 }
152 });
153
154 btnEditar.disableProperty().bind(Tree.getSelectionModel().
    selectedItemProperty().isNull()
155 .or(Tree.getSelectionModel().selectedItemProperty().isEqualTo(
    treeRoot)));
156
157
158 btnExcluir.setOnAction(e -> {
159 TreeItem<FTPFile> selected = Tree.getSelectionModel().
    getSelectedItem();
160 int reply = JOptionPane.showConfirmDialog(null, "Deseja deletar
    esse arquivo ?", "Confirma Exclusão", JOptionPane.
    YES_NO_OPTION);
161 if (reply == JOptionPane.YES_OPTION) {
162 if (DeletarRecursivo(selected)) {
163 selected.getParent().getChildren().remove(selected);
164 if (selected.getValue().isDirectory()) {
```

```
165 JOptionPane.showMessageDialog(null, "Pasta Foi Apagada com
        sucesso !.", "Exclusão", JOptionPane.INFORMATION_MESSAGE);
166 } else {
167 JOptionPane.showMessageDialog(null, "Arquivo Foi Apagado com
        sucesso !.", "Exclusão", JOptionPane.INFORMATION_MESSAGE);
168 }
169 } else {
170 JOptionPane.showMessageDialog(null, "Erro ao tentar Excluir !.",
        "Exclusão", JOptionPane.WARNING_MESSAGE);
171
172 }
173 }
174
175 });
176
177 btnExcluir.disableProperty().bind(Tree.getSelectionModel().
        selectedItemProperty().isNull()
178 .or(Tree.getSelectionModel().selectedItemProperty().isEqualTo(
        treeRoot)));
179
180 TextField textField = new TextField();
181
182
183 EventHandler<ActionEvent> addAction = e -> {
184 try {
185 TreeItem<FTPFile> selected = Tree.getSelectionModel().
        getSelectedItem();
186 FTPFactory.getInstance().getFTP().changeWorkingDirectory(
        selected.getValue().getLink());
187 if (limiteNivel() <= 5) {
188 if (limitePasta()) {
189 if (selected == null) {
190 selected = treeRoot;
191 }
192 String text = JOptionPane.showInputDialog("Nome da Pasta");
193 if (text.isEmpty()) {
194 text = "NovaPasta";
195 }
196 FTPFile f = new FTPFile();
197 f.setType(FTPFile.DIRECTORY_TYPE);
198 f.setRawListing(text);
199 f.setName(text);
200 TreeItem<FTPFile> newItem = new TreeItem<FTPFile>(f, new
        ImageView(pasta));
201
202 if (selected.getValue().isDirectory()) {
203 f.setLink(selected.getValue().getLink() + separador + text);
204 if (FTPFactory.getInstance().getFTP().makeDirectory(f.getLink())
        ) {
```

```
205 selected.getChildren().add(newItem);
206 Tree.getSelectionModel().select(newItem);
207 } else {
208 JOptionPane.showMessageDialog(null, "Erro pasta nao pode ser
       criado com esse nome.", "Diretório Existente", JOptionPane.
       WARNING_MESSAGE);
209 }
210 } else {
211 if (selected.getParent().getValue() != null) {
212 f.setLink(selected.getParent().getValue().getLink() + separador
       + text);
213 if (FTPFactory.getInstance().getFTP().makeDirectory(f.getLink())
       ) {
214 selected.getParent().getChildren().add(newItem);
215 Tree.getSelectionModel().select(newItem);
216 } else {
217 JOptionPane.showMessageDialog(null, "Erro pasta nao pode ser
       criado com esse nome.", "Diretório Existente", JOptionPane.
       WARNING_MESSAGE);
218 }
219 }
220
221 }
222
223 } else {
224 JOptionPane.showMessageDialog(null, "Apenas 5 pastas por
       Diretório", "Limite Atingido", JOptionPane.WARNING_MESSAGE);
225
226 }
227 } else {
228 JOptionPane.showMessageDialog(null, "Apenas 3 níveis de
       Diretórios", "Limite Atingido", JOptionPane.WARNING_MESSAGE);
229
230 }
231 } catch (IOException e1) {
232 e1.printStackTrace();
233 }
234 };
235 textField.setOnAction(addAction);
236 btnAdd.setOnAction(addAction);
237
238 }
239 public TreeItem<FTPFile> CarregarFiles(FTPFile directory,
       boolean v) throws IOException {
240 TreeItem<FTPFile> root;
241
242 if (v) {
243 directory.setType(FTPFile.DIRECTORY_TYPE);
244 directory.setLink(FTPFactory.getInstance().getFTP().
```

```
    printWorkingDirectory());
245 root = new TreeItem<FTPFile>(directory, new ImageView(computador
    ));
246
247 } else {
248 root = new TreeItem<FTPFile>(directory, new ImageView(pasta));
249 }
250 root.setExpanded(true);
251 FTPFile[] files = FTPFactory.getInstance().getFTP().listFiles();
252 for (FTPFile f : files) {
253 System.out.println("Carregando .. " + f.getName());
254 if (f.isDirectory()) {
255 FTPFactory.getInstance().getFTP().changeWorkingDirectory(f.
    getName());
256 f.setLink(FTPFactory.getInstance().getFTP().
    printWorkingDirectory());
257 root.getChildren().add(CarregarFiles(f, false));
258 } else {
259 f.setLink(FTPFactory.getInstance().getFTP().
    printWorkingDirectory() + separador + f.getName());
260 root.getChildren().add(new TreeItem<FTPFile>(f, new ImageView(
    this.arquivo)));
261 }
262 }
263 FTPFactory.getInstance().getFTP().changeToParentDirectory();
264 return root;
265 }
```

Um dos objetivos do trabalho era limitar o cliente FTP, fazendo com que usuários que utilizar o sistema só poderam criar 5 pastas e 2 arquivos por diretório, sendo que poderá criar no máximo 3 níveis de diretórios.

A classe Limitador do pacote socket e responsável por fazer a armazenagem da quantidade máxima de pastas e arquivos. Sendo utilizada na classe navegação do pacote cliente, em conjunto dos métodos limiteArquivo(), limitePasta() e limiteNivel().

```
1  public class Limitador {
2
3      private int p;
4      private int a;
5
6      public Limitador(int p, int a) {
7          this.p = p;
8          this.a = a;
9      }
10
11     public int getMP() {
12         return p;
13     }
```

```
14
15      public int getMA() {
16          return a;
17      }
18
19  }
20
21  private boolean limiteArquivo() throws IOException {
22      int num = FTPFactory.getInstance().getFTP().listDirectories
            ().length;
23      int numa = FTPFactory.getInstance().getFTP().listFiles().
            length - num;
24      return numa < limite.getMA();
25  }
26
27  private boolean limitePasta() throws IOException {
28      int num_diretorios = FTPFactory.getInstance().getFTP().
            listDirectories().length;
29      return num_diretorios < limite.getMP();
30  }
31
32  private int limiteNivel() throws IOException {
33      int num = FTPFactory.getInstance().getFTP().
            printWorkingDirectory().split("/").length;
34      return num;
35  }
```

Para deletar uma pasta deve-se utilizar o método removeDirectory(Caminho), ou para deletar um arquivo utiliza-se o método deleteFile(Caminho) do FTPClient Apache. Porém para deletar uma pasta que contém N sub pastas deve-se percorrer recursivamente as pastas, até a folha mais baixa e vim apagando debaixo para cima cada pasta ou arquivo de cada pasta recursivamente pelo método DeletarRecursivo(TreeItem). Para renomear um arquivo ou pasta utiliza-se o método rename(caminho,novonome) da Apache. Porém para mudar os links das sub pastas de um diretório

```
1
2       public boolean DeletarRecursivo(TreeItem<FTPFile> a) {
3       boolean flag = false;
4
5       if (a.getChildren().isEmpty()) {
6       if (FTPFactory.getInstance().Excluir(a.getValue())) {
7       return true;
8       }
9       } else {
10      for (TreeItem<FTPFile> iterator: a.getChildren()){
11      DeletarRecursivo(iterator);
12      }
13      if (FTPFactory.getInstance().Excluir(a.getValue())) {
14      return true;
```

```
15      }
16      }
17      return false;
18      }
19
20
21      public void RenameRecursivao(TreeItem<FTPFile> a) {
22      String novolink;
23      for (Iterator<TreeItem<FTPFile>> iterator = a.getChildren().
           iterator(); iterator.hasNext(); ) {
24      TreeItem<FTPFile> c = iterator.next();
25      novolink = a.getValue().getLink() + separador + c.getValue()
           .getName();
26      c.getValue().setLink(novolink);
27      if (!c.getChildren().isEmpty()) {
28      RenameRecursivao(c);
29      }
30
31      }
32      }
```