Sistma Gerenciador de FTP
Exercício Computacional I

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 construido um sistema gerenciador de FTP, utilizando JavaFx como Graphical User Interface e o Apache Commons Net 3.6, como 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 JFoenix, essa bibliteca é open sorce e pode ser baixada no github https://github.com/jfoenixadmin/JFoenix Para icones foi utilizado a biblioteca fontawesomefx-8.9

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 na versão 3.6 current.

3. Problematica

O trabalho proposto tem como objetivo criar um Cliente FTP, onde Adicione, Edite, Baixe e faz Upload de arquivos e pastas, porem apenas 5 pastas e 2 arquivos por diretorio sendo que no máximo deve ter 3 níveis. Um problema identificado e referênte a segurança desses pré-requisitos, onde que em um Cliente FTP a segurança da validação desses requisitos e inteiramente do Software Cliente é não do servidor, sendo assim para interece academico as validações devem ser inteiramente do SERVIDOR FTP não do CLIENTE FTP.

4. References

Bibliographic references must be unambiguous and uniform. We recommend giving the author names references in brackets, e.g. [Knuth 1984], [Boulic and Renault 1991], and [Smith and Jones 1999].

The references must be listed using 12 point font size, with 6 points of space before each reference. The first line of each reference should not be indented, while the subsequent should be indented by 0.5 cm.

Referências

Boulic, R. and Renault, O. (1991). 3d hierarchies for animation. In Magnenat-Thalmann, N. and Thalmann, D., editors, New Trends in Animation and Visualization. John Wiley & Sons ltd.

Knuth, D. E. (1984). The TEX Book. Addison-Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In Smith-Jones, A. B., editor, Advances in Computer Science, pages 555–566. Publishing Press.

5. Anexos

A estrutura do código foi dividida em 3 pastas sendo elas 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.

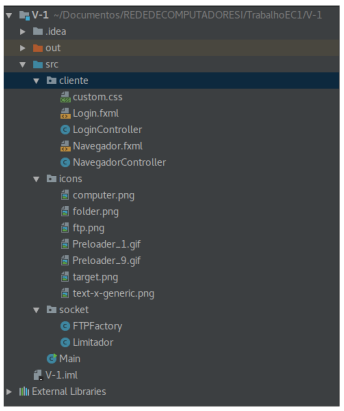

Figura 1. Imagem da Tela de Login.

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.

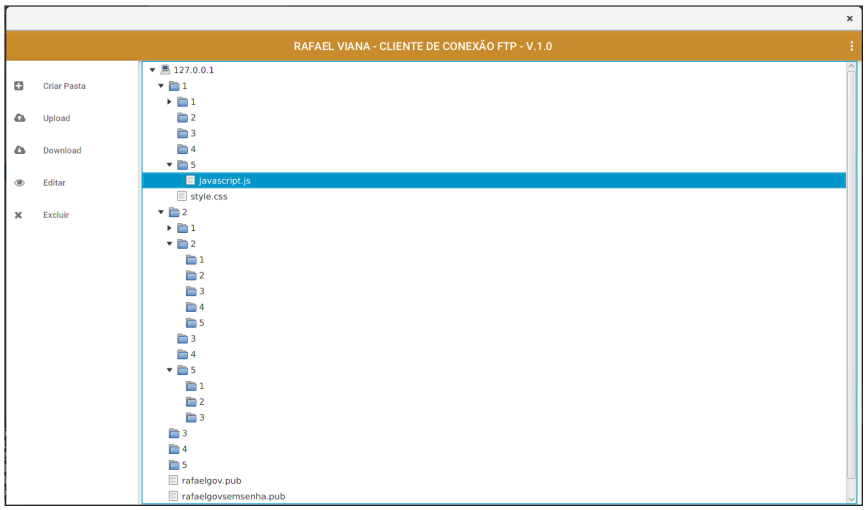

Figura 3. Imagem da Tela de Navegação.

Toda lógica de abastecimento da TreeView e criação dos ItemView utilizados na TreeView estão no código de criação da tela de navegação,o método Navegação inicia a criação dos ItemViws e toda a estrutura da arvore como ADICIONAR, REMOVER, LISTAR, EDITAR, BAIXAR e UPLOAD de arquivos/pastas, o método getNodesForDirectory() cria recursivamente Nodes/ItemView encadeando seus filhos na TreeView, facilitando assim a navegação.

```
1  private void Navegacao() throws IOException {
2      FTPFile files[];
3
4
5      TreeItem<FTPFile> treeRoot;
6
7      files = FTPFactory.getInstance().getFTP().listFiles();
8      Tree.setEditable(true);
9
10     if (files != null && files.length > 0) {
11         files[0].setRawListing(FTPFactory.getInstance().getFTP()
               .getPassiveHost());
12         treeRoot = getNodesForDirectory(files[0], true);
13     } else {
14         FTPFile file = new FTPFile();
15         file.setType(FTPFile.DIRECTORY_TYPE);
16         file.setLink(FTPFactory.getInstance().getFTP().
               printWorkingDirectory());
17         file.setRawListing(FTPFactory.getInstance().getFTP().
               getPassiveHost());
18         treeRoot = new TreeItem<>(file , new ImageView(computador
               ));
19     }
20     Tree.getSelectionModel().select(treeRoot);
21     Tree.setRoot(treeRoot);
22
23
24     btnBaixar.disableProperty().bind(Tree.getSelectionModel().
           selectedItemProperty().isNull()
25     .or(Tree.getSelectionModel().selectedItemProperty().
           isEqualTo(treeRoot)));
26
27
28     btnBaixar.setOnAction(e -> {
29         TreeItem<FTPFile> selected = Tree.getSelectionModel().
               getSelectedItem();
30         if (selected.getValue().isFile()) {
31             JFileChooser local = new JFileChooser();
32             local.setFileSelectionMode(JFileChooser.
                   DIRECTORIES_ONLY);
33             local.setDialogTitle("Escolha um local para salvar")
                   ;
34             local.setFileHidingEnabled(false);
```

```
35              int res = local.showSaveDialog(null);
36              if (res == JFileChooser.APPROVE_OPTION) {
37                  String caminho = String.valueOf(local.
                        getSelectedFile());
38                  try {
39                      FileOutputStream fos = new FileOutputStream(
                            caminho);
40                      if (FTPFactory.getInstance().getFTP().
                            retrieveFile(selected.getValue().getLink
                            (), fos)) {
41                          JOptionPane.showMessageDialog(null, "
                                Arquivo Baixado com Sucesso!" + "\n\n
                                ",
42                          "Sucesso", JOptionPane.
                                INFORMATION_MESSAGE);
43                      }
44
45                  } catch (FileNotFoundException e1) {
46                      e1.printStackTrace();
47                  } catch (Exception ex) {
48                      JOptionPane.showMessageDialog(null, "Erro ao
                            Baixado  Arquivo : " + ex.getMessage(),
                            "Erro", JOptionPane.ERROR_MESSAGE);
49                  }
50              }
51          } else {
52              JOptionPane.showMessageDialog(null, "Somente
                    Arquivos!" + "\n\n",
53              "Erro", JOptionPane.ERROR_MESSAGE);
54          }
55      });
56
57
58      btnUp.setOnAction(e -> {
59          TreeItem<FTPFile> selected = Tree.getSelectionModel().
                getSelectedItem();
60          try {
61              FTPFactory.getInstance().getFTP().
                    changeWorkingDirectory(selected.getValue().
                    getLink());
62              if (limiteNivel() <= 5) {
63                  if (limiteArquivo()) {
64                      JFileChooser fc = new JFileChooser();
65                      fc.setFileSelectionMode(JFileChooser.
                            FILES_ONLY);
66                      int result = fc.showOpenDialog(null);
67                      if (result == JFileChooser.APPROVE_OPTION) {
68                          File arquivo = fc.getSelectedFile();
69                          InputStream isArquivo = null;
```

```
70                          try {
71                              isArquivo = new FileInputStream(
                                    arquivo.getAbsolutePath());
72                          } catch (FileNotFoundException e1) {
73                              e1.printStackTrace();
74                          }
75                          try {
76                              FTPFile f = new FTPFile();
77
78                              if (selected.getValue().isDirectory
                                    ()) {
79                                  FTPFactory.getInstance().getFTP
                                        ().changeWorkingDirectory(
                                        selected.getValue().getLink()
                                        );
80                                  f.setLink(selected.getValue().
                                        getLink() + separador +
                                        arquivo.getName());
81                              } else {
82                                  FTPFactory.getInstance().getFTP
                                        ().changeWorkingDirectory(
                                        selected.getParent().getValue
                                        ().getLink());
83                                  f.setLink(selected.getParent().
                                        getValue().getLink() +
                                        separador + arquivo.getName()
                                        );
84                              }
85
86                              if (FTPFactory.getInstance().getFTP
                                    ().storeFile(arquivo.getName(),
                                    isArquivo)) {
87
88                                  f.setName(arquivo.getName());
89                                  f.setRawListing(arquivo.getName
                                        ());
90                                  f.setType(FTPFile.FILE_TYPE);
91                                  TreeItem<FTPFile> newItem = new
                                        TreeItem<FTPFile>(f, new
                                        ImageView(this.arquivo));
92                                  if (selected.getValue().
                                        isDirectory()) {
93                                      selected.getChildren().add(
                                            newItem);
94                                  } else {
95                                      selected.getParent().
                                            getChildren().add(newItem
                                            );
96                                  }
```

```
97
98                                  JOptionPane.showMessageDialog(
                                        null, "Arquivo Enviado!");
99                              } else {
100                                 JOptionPane.showMessageDialog(
                                        null, "Arquivo não enviado!")
                                        ;
101                             }
102
103
104                         } catch (IOException e1) {
105                             e1.printStackTrace();
106                         }
107
108                     } else {
109                         JOptionPane.showMessageDialog(null, "
                                Arquivo não selecionado!");
110                     }
111                 } else {
112                     JOptionPane.showMessageDialog(null, "Apenas
                            2 arquivos por pasta", "Limite Atingido",
                            JOptionPane.WARNING_MESSAGE);
113                 }
114             } else {
115                 JOptionPane.showMessageDialog(null, "Apenas 3
                        níveis de Diretórios", "Limite Atingido",
                        JOptionPane.WARNING_MESSAGE);
116
117             }
118         } catch (IOException e1) {
119             e1.printStackTrace();
120         }
121     });
122
123
124     btnEditar.setOnAction(e -> {
125         TreeItem<FTPFile> selected = Tree.getSelectionModel().
                getSelectedItem();
126         String novolink = "";
127         if (selected.getParent().getValue() != null) {
128
129             try {
130                 FTPFactory.getInstance().getFTP().
                        changeWorkingDirectory(selected.getParent().
                        getValue().getLink());
131                 novolink = FTPFactory.getInstance().getFTP().
                        printWorkingDirectory();
132             } catch (IOException e1) {
133                 e1.printStackTrace();
```

```
134             }
135             try {
136                 String novoNome = JOptionPane.showInputDialog("
                        Digite um novo nome para " + selected.
                        getValue().getName());
137 
138                 if (FTPFactory.getInstance().getFTP().rename(
                        selected.getValue().getName(), novoNome)) {
139                     selected.getValue().setRawListing(novoNome);
140                     selected.getValue().setLink(novolink +
                            separador + novoNome);
141                     Tree.refresh();
142                 }
143             } catch (IOException e1) {
144                 e1.printStackTrace();
145             }
146         }
147 
148     });
149 
150     btnEditar.disableProperty().bind(Tree.getSelectionModel().
            selectedItemProperty().isNull()
151     .or(Tree.getSelectionModel().selectedItemProperty().
            isEqualTo(treeRoot)));
152 
153 
154     btnExcluir.setOnAction(e -> {
155         TreeItem<FTPFile> selected = Tree.getSelectionModel().
                getSelectedItem();
156         int reply = JOptionPane.showConfirmDialog(null, "Deseja
                deletar esse arquivo ?", "Confirma Exclusão",
                JOptionPane.YES_NO_OPTION);
157         if (reply == JOptionPane.YES_OPTION) {
158             if (FTPFactory.getInstance().Excluir(selected.
                    getValue())) {
159                 System.out.println(selected.getValue().getLink()
                        );
160                 selected.getParent().getChildren().remove(
                        selected);
161                 JOptionPane.showMessageDialog(null, "Excluido ",
                        "OK", JOptionPane.INFORMATION_MESSAGE);
162             } else {
163                 JOptionPane.showMessageDialog(null, "Erro ao
                        excluir arquivo ", "Erro", JOptionPane.
                        WARNING_MESSAGE);
164             }
165         }
166 
167     });
```

```
168
169
170     btnExcluir.disableProperty().bind(Tree.getSelectionModel().
            selectedItemProperty().isNull()
171     .or(Tree.getSelectionModel().selectedItemProperty().
            isEqualTo(treeRoot)));
172
173     TextField textField = new TextField();
174
175
176     EventHandler<ActionEvent> addAction = e -> {
177         try {
178             TreeItem<FTPFile> selected = Tree.getSelectionModel
                    ().getSelectedItem();
179             FTPFactory.getInstance().getFTP().
                    changeWorkingDirectory(selected.getValue().
                    getLink());
180             if (limiteNivel() <= 5) {
181                 if (limitePasta()) {
182                     if (selected == null) {
183                         selected = treeRoot;
184                     }
185                     String text = JOptionPane.showInputDialog("
                            Nome da Pasta");
186                     if (text.isEmpty()) {
187                         text = "NovaPasta";
188                     }
189                     FTPFile f = new FTPFile();
190                     f.setType(FTPFile.DIRECTORY_TYPE);
191                     f.setRawListing(text);
192                     f.setName(text);
193                     TreeItem<FTPFile> newItem = new TreeItem<
                            FTPFile>(f, new ImageView(pasta));
194
195                     if (selected.getValue().isDirectory()) {
196                         f.setLink(selected.getValue().getLink()
                                +separador + text);
197                         System.out.println("Link: " + f.getLink
                                ());
198                         if (FTPFactory.getInstance().getFTP().
                                makeDirectory(f.getLink())) {
199                             selected.getChildren().add(newItem);
200                             Tree.getSelectionModel().select(
                                    newItem);
201                         } else {
202                             JOptionPane.showMessageDialog(null,
                                    "Erro pasta nao pode ser criado
                                    com esse nome.", "Diretório
                                    Existente", JOptionPane.
```

```
                                    WARNING_MESSAGE);
203                         }
204                     } else {
205                         if (selected.getParent().getValue() !=
                                null) {
206                             f.setLink(selected.getParent().
                                    getValue().getLink() + separador
                                    + text);
207                             System.out.println("Link: " + f.
                                    getLink());
208                             if (FTPFactory.getInstance().getFTP
                                    ().makeDirectory(f.getLink())) {
209                                 selected.getParent().getChildren
                                        ().add(newItem);
210                                 Tree.getSelectionModel().select(
                                        newItem);
211                             } else {
212                                 JOptionPane.showMessageDialog(
                                        null, "Erro pasta nao pode
                                        ser criado com esse nome.", "
                                        Diretório Existente",
                                        JOptionPane.WARNING_MESSAGE);
213                             }
214                         }
215
216                     }
217
218                 } else {
219                     JOptionPane.showMessageDialog(null, "Apenas
                            5 pastas por Diretório", "Limite Atingido
                            ", JOptionPane.WARNING_MESSAGE);
220
221                 }
222             } else {
223                 JOptionPane.showMessageDialog(null, "Apenas 3
                        níveis de Diretórios", "Limite Atingido",
                        JOptionPane.WARNING_MESSAGE);
224
225             }
226         } catch (IOException e1) {
227             e1.printStackTrace();
228         }
229     };
230     textField.setOnAction(addAction);
231     btnAdd.setOnAction(addAction);
232
233 }
234
235
```

```
236 public TreeItem<FTPFile> getNodesForDirectory(FTPFile directory,
        boolean v) throws IOException {
237     TreeItem<FTPFile> root;
238     if (v) {
239         directory.setType(FTPFile.DIRECTORY_TYPE);
240         directory.setLink(FTPFactory.getInstance().getFTP().
            printWorkingDirectory());
241         root = new TreeItem<FTPFile>(directory, new ImageView(
            computador));
242 
243     } else {
244         root = new TreeItem<FTPFile>(directory, new ImageView(
            pasta));
245     }
246     FTPFile[] files = FTPFactory.getInstance().getFTP().
        listFiles();
247     for (FTPFile f : files) {
248         System.out.println("Carregando .. " + f.getName());
249         if (f.isDirectory()) {
250             FTPFactory.getInstance().getFTP().
                changeWorkingDirectory(f.getName());
251             f.setLink(FTPFactory.getInstance().getFTP().
                printWorkingDirectory());
252             System.out.println(f.getLink());
253             root.getChildren().add(getNodesForDirectory(f, false
                ));
254         } else {
255             f.setLink(FTPFactory.getInstance().getFTP().
                printWorkingDirectory() + separador + f.getName()
                );
256             root.getChildren().add(new TreeItem<FTPFile>(f, new
                ImageView(this.arquivo)));
257         }
258     }
259     FTPFactory.getInstance().getFTP().changeToParentDirectory();
260     return root;
261 }
```

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

```
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  }
```

Na classe Navegação do pacote cliente 3 métodos fazem parte da lógica empregada na Class Limitador sendo las.

```
1
2   private boolean limiteArquivo() throws IOException {
3       int num = FTPFactory.getInstance().getFTP().listDirectories
            ().length;
4       int numa = FTPFactory.getInstance().getFTP().listFiles().
            length - num;
5       return numa < limite.getMA();
6   }
7
8   private boolean limitePasta() throws IOException {
9       int num_diretorios = FTPFactory.getInstance().getFTP().
            listDirectories().length;
10      return num_diretorios < limite.getMP();
11  }
12
13  private int limiteNivel() throws IOException {
14      int num = FTPFactory.getInstance().getFTP().
            printWorkingDirectory().split("separador").length;
15      return num;
16  }
```