Simples chat ponto-a-ponto NFC LLCP

Exemplo de uma comunicação NFC LLCP no Nokia 6212 classic. O usuário pode trocar mensagens com outro aparelho, teclando. Por padrão o MIDlet está no estado de leitura e quando o usuário quer escrever alguma coisa o MIDlet vai para o estado de escrita.

Note que a classe LLCPConnection usada neste exemplo é implementação própria para o Nokia 6212 classic, sendo assim este exemplo pode não funcionar em outros aparelhos.

```
package com.nokia.nfc.sample.app;

import java.io.IOException;

import javax.microedition.io.Connector;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.TextBox;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

import com.nokia.nfc.llcp.LLCPConnection;
import com.nokia.nfc.llcp.LLCPConnectionListener;
import com.nokia.nfc.llcp.LLCPLinkListener;
import com.nokia.nfc.llcp.LLCPManager;

public class LLCPChat extends MIDlet implements CommandListener, LLCPConnectionListener,
LLCPLinkListener {

 /**
  * LLCP manager
  */
 private LLCPManager llcpManager  = null;

 /**
  * LLCP connections
  *
  * @note
  * As conexões de entrada (conexão recebida pelo método connectionOpened()) e
  * e a conexão de saída (criada após a invocação do método linkEstablished())
  * precisam ser criadas.
  */
 private LLCPConnection inbound;
 private LLCPConnection outbound;

 /**
  * Parâmetros de conexão
  *
  * @note
  * O URI da conexao Connection URI e para abrir uma conexão de tipo 2 que utiliza o
identificador do serviço
  * (SID) 7. Observe que o PID mencionado no javadocs do SDK não é válido. Utilize SID
```


(C) Copyright Nokia 2014. All rights reserved.

```
 */
private static final byte SID  = 7;
private static final byte TYPE  = LLCPConnection.TYPE_2;

/**
 * Buffers de dados da conexão
 */
private byte[] sendBuffer  = null;
private byte[] receiveBuffer  = null;

/**
 * Componentes de tela
 */
private Display display   = null;
private Form form   = null;
private TextBox textTextBox  = null;

/**
 * Componentes Commands
 */
private Command writeCmd  = null;
private Command cancelCmd  = null;
private Command exitCmd   = null;

/**
 * "Máquina de Estados" - se a MIDlet está no estado de leitura ou escrita
 */
boolean reading = true;

/*
 *
 */
public LLCPChat() {
 System.out.println("\n\n LLCPChat started");

 /**
  *  Inicializa a interface com o usuário
  */
 display  = Display.getDisplay(this);

 form  = new Form("LLCPChat");

 writeCmd = new Command("Write", Command.ITEM, 1);
 cancelCmd = new Command("Cancel", Command.CANCEL, 1);
 exitCmd  = new Command("Exit", Command.EXIT, 1);

 form.addCommand(writeCmd);
 form.addCommand(exitCmd);
 form.setCommandListener(this);

 display.setCurrent(form);

}

/**
 * Este método é invocado quando a MIDlet está fechando ou entrando no estado de Pause
 */
private void close() {
```


(C) Copyright Nokia 2014. All rights reserved.

```
  System.out.println("close");

  llcpManager.stopListening(TYPE, SID, this);
  llcpManager.removeLinkListener(this);

  inbound = closeConn(inbound);
  outbound = closeConn(outbound);

 }

 /**
  * Este método é invocado pelos handlers de conexão de entrada e saída
  * para fechar as conexões existentes
  *
  * @param conn
  * @return null quando a conexão está fechada
  */
 private final LLCPConnection closeConn(LLCPConnection conn) {
  System.out.println("closeConn");

  if (conn != null) {
   try {
    conn.close();
   } catch (IOException e) {
    System.out.println("Error closing connection : " + e.getMessage());
   }
  }
  return null;
 }

 /*
  * (non-Javadoc)
  * @see
javax.microedition.lcdui.CommandListener#commandAction(javax.microedition.lcdui.Command,
javax.microedition.lcdui.Displayable)
  */
 public void commandAction(Command c, Displayable d) {
  System.out.println("commandAction");

  if (c == writeCmd && d == form) {
   displayWritingTextBox();
  } else if (c == writeCmd && d == textTextBox) {
   write();
  } else if (c == cancelCmd) {
   read();
  } else if (c == exitCmd) {
   notifyDestroyed();
  }
 }

 /*
  * (non-Javadoc)
  * @see
com.nokia.nfc.llcp.LLCPConnectionListener#connectionOpened(com.nokia.nfc.llcp.LLCPConnection)
  */
 public void connectionOpened(LLCPConnection conn) {
  System.out.println("connectionOpened");
  if (inbound == null) {
```


(C) Copyright Nokia 2014. All rights reserved.

```
   inbound = conn;
   handleIn();
  } else {
   closeConn(conn);
  }
 }

 /**
  * Este método é invocado para mostrar na tela o texto num componente TextBox
  */
 private void displayWritingTextBox() {
  System.out.println("writeTb");

  textTextBox = new TextBox("Text to write", null, 255, TextField.ANY);
  textTextBox.addCommand(writeCmd);
  textTextBox.addCommand(cancelCmd);
  textTextBox.setCommandListener(this);

  display.setCurrent(textTextBox);

 }

 /*
  * (non-Javadoc)
  * @see javax.microedition.midlet.MIDlet#destroyApp(boolean)
  */
 protected void destroyApp(boolean unconditional) throws MIDletStateChangeException {
  System.out.println("destroyApp");
  close();
 }

 /**
  * Handler da conexão de entrada
  */
 private final void handleIn() {
  System.out.println("handleIn");

  new Thread() {
   public void run() {

    System.out.println("handleIn begin");

    try {
     if(reading) {
      sendData(inbound, "-1".getBytes());
      byte[] data  = receiveData(inbound);
      handleReceivedData(data);
     } else {
      sendData(inbound, sendBuffer);
      byte[] data = receiveData(inbound);
      if(new String(data).equals("-1")) handleReceivedData(data);
     }
    } catch (Exception e) {
     System.out.println("Error handling incoming connection : " + e.getMessage());
    }

    System.out.println("handleIn end");
```


(C) Copyright Nokia 2014. All rights reserved.

```
    }
   }.start();
  }

  /**
   *
   * @param uid
   */
  private final void handleOut(final String uid) {
   System.out.println("handleOut");

   new Thread() {
    public void run() {

     System.out.println("handleOut begin");

     try {
      outbound = (LLCPConnection) Connector.open("nfc:llcp;type=" + TYPE + ";sid=" + SID
+ ";uid=" + uid);
      System.out.println("Opened outgoing connection");

      if(reading) {
       sendData(outbound, "-1".getBytes());
       byte[] data  = receiveData(outbound);
       handleReceivedData(data);
      } else {
       sendData(outbound, sendBuffer);
       byte[] data = receiveData(outbound);
       if(new String(data).equals("-1")) handleReceivedData(data);
      }

     } catch (IOException e) {
      System.out.println("Error handling outgoing connection : " + e.getMessage());
     }

     System.out.println("handleOut end");

    }
   }.start();
  }

  /**
   *
   * @param data
   */
  private synchronized void handleReceivedData(byte[] data) {
   System.out.println("handleReceivedData");

   /*
    * Recebe apenas a primeira resposta que vem dos handlers da conexão de entrada ou da
conexão de saída
    */
   if(receiveBuffer == null) {
    receiveBuffer = data;
   } else return;

   if(reading) {
    if(!new String(receiveBuffer).equals("-1")) {
```


(C) Copyright Nokia 2014. All rights reserved.

```
    form.append(">> " + new String(receiveBuffer) + "\n");
   }
  } else {
   if(sendBuffer != null) {
    form.append("<< " + new String(sendBuffer) + "\n");
    read();
   }
  }
 }

 /*
  * (non-Javadoc)
  * @see com.nokia.nfc.llcp.LLCPLinkListener#linkEstablished(java.lang.String)
  */
 public void linkEstablished(String uid) {
  System.out.println("linkEstablished");

  receiveBuffer = null;
  handleOut(uid);

 }

 /*
  * (non-Javadoc)
  * @see com.nokia.nfc.llcp.LLCPLinkListener#linkLost()
  */
 public void linkLost() {
  System.out.println("linkLost");

  outbound = closeConn(outbound);
  inbound = closeConn(inbound);

 }

 /**
  * Este método é invocado quando a MIDlet inicia-se ou finaliza o estado de Pause
  */
 private void open() {
  System.out.println("open");

  try {
   llcpManager = LLCPManager.getInstance();
  } catch (Exception e) {
   System.out.println("Failed to get LLCPManager instance : " + e.getMessage());
   return;
  }

  llcpManager.addLinkListener(this);
  llcpManager.startListening(TYPE, SID, this);

 }

 /*
  * (non-Javadoc)
  * @see javax.microedition.midlet.MIDlet#pauseApp()
  */
 protected void pauseApp() {
  System.out.println("pauseApp");
```


(C) Copyright Nokia 2014. All rights reserved.

```
 close();
}

/**
 *
 */
private void read() {
 System.out.println("read");

 sendBuffer = null;
 receiveBuffer = null;

 reading = true;
 display.setCurrent(form);

}

/**
 *
 * @param conn
 * @return
 * @throws IOException
 */
private final byte[] receiveData(LLCPConnection conn) throws IOException {
 System.out.println("receiveData");

 byte[] recv = conn.receive();
 System.out.println("Received " + new String(recv));
 return recv;

}

/**
 *
 * @param conn
 * @param data
 * @throws IOException
 */
private final void sendData(LLCPConnection conn, byte[] data) throws IOException {
 System.out.println("sendData");

 if(data == null) data = "\n".getBytes();

 conn.send(data);
 System.out.println("Sent " + new String(data));
}

/*
 * (non-Javadoc)
 * @see javax.microedition.midlet.MIDlet#startApp()
 */
protected void startApp() throws MIDletStateChangeException {
 System.out.println("startApp");

 open();
}
```


(C) Copyright Nokia 2014. All rights reserved.

```
  /**
   *
   */
  private void write() {
   System.out.println("write");

   reading = false;
   sendBuffer = textTextBox.getString().getBytes();

   // Mostra um alerta
   Alert a = new Alert("Writing", "Touch to write", null, AlertType.INFO);
   a.addCommand(cancelCmd);
   a.setTimeout(Alert.FOREVER);
   a.setCommandListener(this);
   display.setCurrent(a);

  }
 }
```


(C) Copyright Nokia 2014. All rights reserved.