

**CENTRO**

20
20

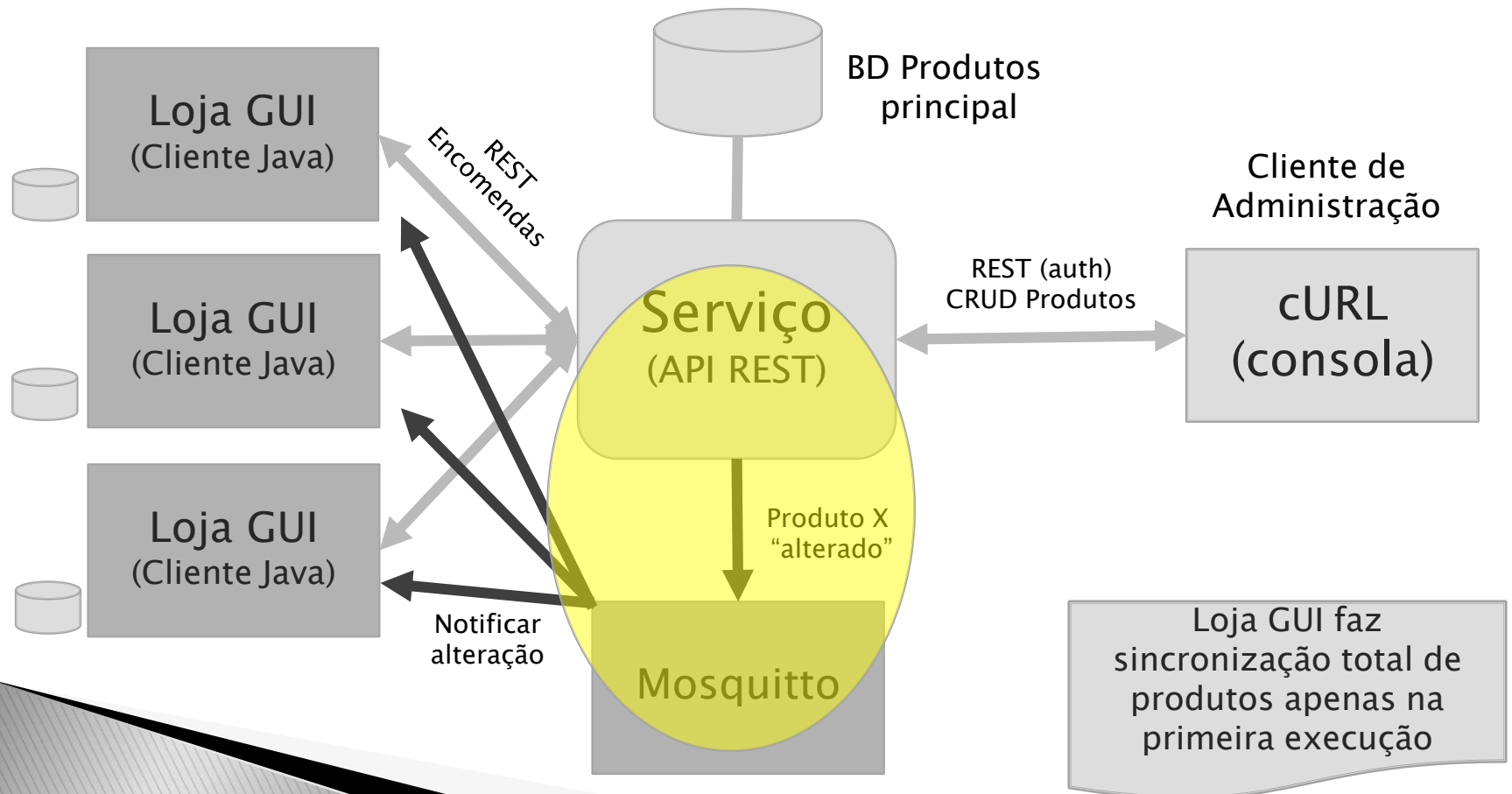


# Serviços e Interoperabilidade de Sistemas

## Parte 3 – Cliente Java: Sincronização da DB

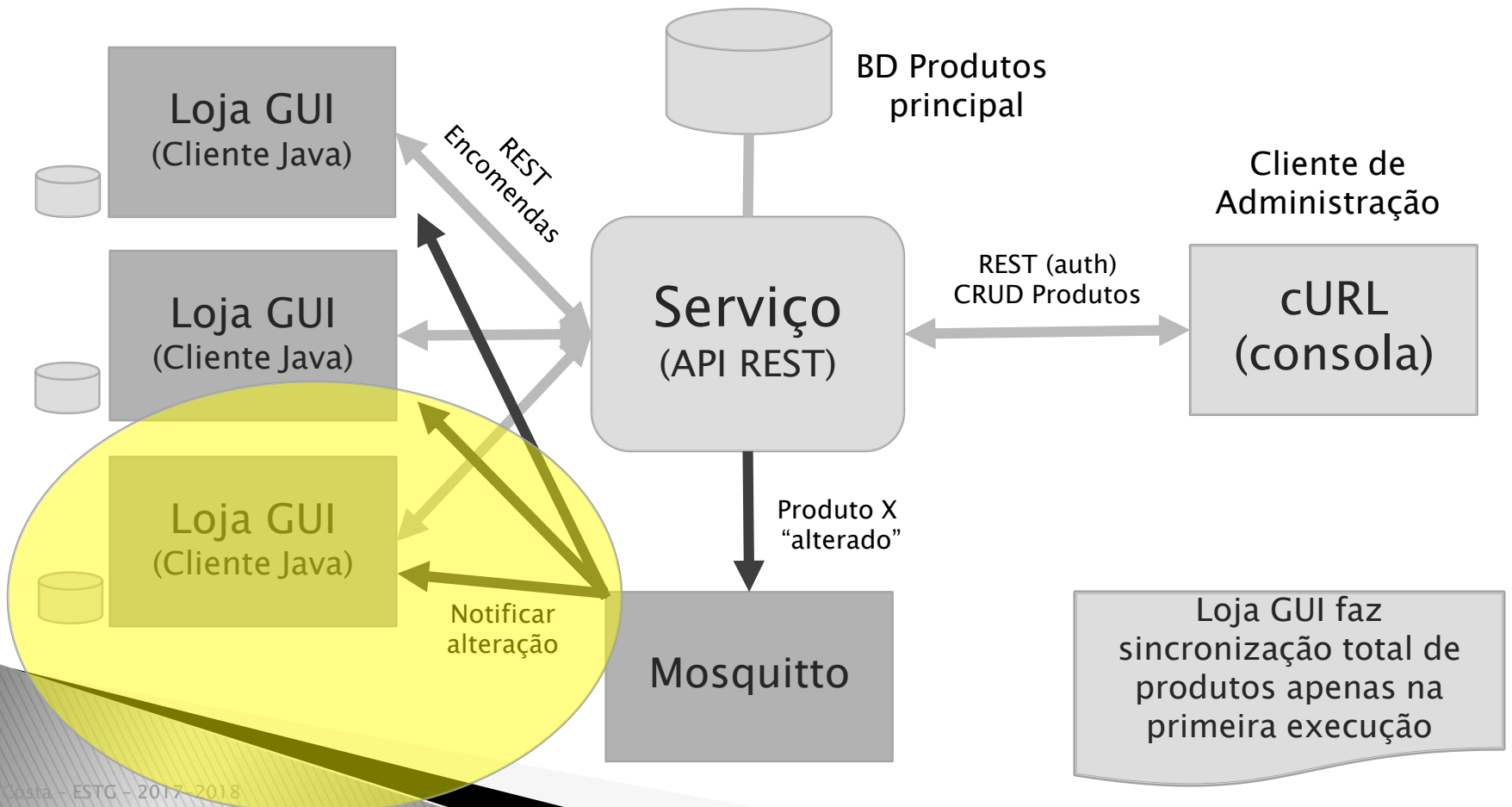
# Loja em “dispositivos móveis”

## ▶ Arquitetura informal de alto nível: última aula



# Loja em “dispositivos móveis”

▶ **Arquitetura informal de alto nível: hoje**



# Loja em “dispositivos móveis”

## ▶ Notificações

- Quando os dados da tabela produtos são alterados...
  - Canal INSERT (registro de produto em JSON)
  - Canal UPDATE (registro de produto em JSON)
  - Canal DELETE (apenas id em JSON)

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

- SQLite – eficiente (e “mono” utilizador)
  - Descarregar e instalar jar sqlite-jdbc
    - <http://www.java2s.com/Code/JarDownload/sqlite/sqlite-jdbc-3.7.2.jar.zip>
- JDialog: Rclick | Events | Window | windowOpened

```
private void formWindowOpened(java.awt.event.WindowEvent evt)
{
    OpenDb();
    SynchDB();
    ShowProducts();
}
```

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
private Connection conn = null;
...

private boolean OpenDb()
{
    Class.forName("org.sqlite.JDBC"); //Driver
    conn = DriverManager.getConnection("jdbc:sqlite:test.db");


    //Criar tabela...
    String sql = "CREATE TABLE IF NOT EXISTS produtos (id
        integer,designacao text,preco real, img text)";
    Statement stmt = conn.createStatement();
    stmt.execute(sql);
}
```

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
private void SynchDB()
{
    int nTotProds=GetProdutosRowCount();
    if(nTotProds < 0)
    {
        JOptionPane.showMessageDialog(null, "Erro na DB local. Impossível
            sincronizar...");
        return;
    }
    else if(nTotProds > 0)
    {
        return; //Já sincronizada
    }

    String strJson = FetchAllProdutos("http://127.0.0.1:8888/api/produtos");
    StoreAllProdutos(strJson);
}
```



Const!

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
private int GetProdutosRowCount()
{
    int nTotRows=0;
    if(conn == null)
    {
        return -1;
    }
    Statement stmt = null;
    stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery( "SELECT count(*) as total FROM
                                     produtos;" );
    if( rs.next() )
    {
        nTotRows = rs.getInt("total");
    }
    rs.close(); stmt.close();
    return nTotRows;
}
```



# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
public class Produto
{
    public long id;           //igual ao json/bd
    public String designacao;
    public double preco;
    public String img;
}
```

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
private String FetchAllProdutos(String path)
{
    String name = "admin";
    String password = "admin";
    String authString = name + ":" + password;
    byte[] authEncBytes = Base64.getEncoder().
        encode(authString.getBytes());
    String authStringEnc = new String(authEncBytes);

    URL url = new URL(path);
    HttpURLConnection conn = (HttpURLConnection)
        url.openConnection();
    conn.setRequestMethod("GET");
    conn.setRequestProperty("Accept", "application/json");
    conn.setRequestProperty("Authorization", "Basic " + authStringEnc);
}
```



java.net

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
if (conn.getResponseCode() != 200)
{
    JOptionPane.showMessageDialog(null, "Impossível contactar o
                                   servidor");
    return "";
}
```

```
BufferedReader br = new BufferedReader(new
    InputStreamReader((conn.getInputStream())));
String strLine = "";
StringBuffer strAllLines=new StringBuffer();
while((strLine = br.readLine()) != null)
{
    strAllLines.append(strLine);
}
conn.disconnect();    return strAllLines.toString();
```

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
private boolean StoreAllProdutos(String str_json_produtos)
{
    if(conn == null) return false;

    Gson gson = new Gson();
    Produto[] arrProdutos = gson.fromJson(str_json_produtos, Produto[].class);
    for(int i=0;i<arrProdutos.length;i++)
    {
        Produto p = (Produto) arrProdutos[i];
        String sql = "Insert into produtos values
            (" + p.id + ", " + p.designacao + ", " + p.preco + ", " + p.img + ")";
        Statement stmt = conn.createStatement();
        stmt.execute(sql);
        stmt.close();
    }
    return true;
}
```

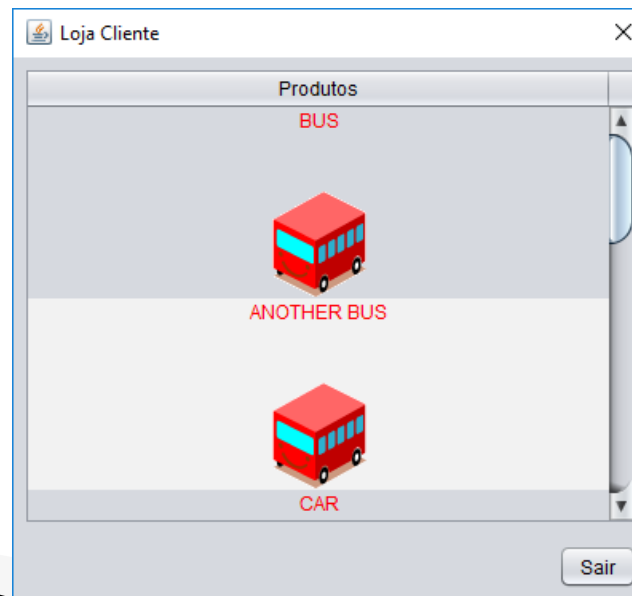
# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

Arrastar uma JTable para a JDialog

**A JTable terá uma CustomCell (designação + imagem)**

Innerclass da JDialog para CustomCell: MultiLabelRenderer



# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
class MultiLabelRenderer implements TableCellRenderer
{
    private JPanel panel;
    private JLabel desigLabel;
    private JLabel imgLabel;

    public MultiLabelRenderer()
    {
        panel = new JPanel(new BorderLayout());

        desigLabel = new JLabel();
        desigLabel.setForeground(Color.RED);
        desigLabel.setHorizontalAlignment(SwingConstants.CENTER);
        desigLabel.setVerticalAlignment(SwingConstants.CENTER);

        imgLabel = new JLabel();
        imgLabel.setForeground(Color.BLUE);
        imgLabel.setHorizontalAlignment(SwingConstants.CENTER);
        imgLabel.setVerticalAlignment(SwingConstants.CENTER);
    }
}
```

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
public Component getTableCellRendererComponent(JTable table, Object
    value, boolean isSelected,boolean hasFocus,
    final int row, final int column)
{
    panel.removeAll();
    if (isSelected)
        panel.setBackground( table.getSelectionBackground() );
    else
        panel.setBackground( table.getBackground() );

    if (value == null) return panel;

    Produto p = (Produto) value;
    desigLabel.setText( p.designacao );

    byte[] btDataFile = Base64.getDecoder().decode(p.img);
    BufferedImage image = ImageIO.read(new
        ByteArrayInputStream(btDataFile));
    if(image != null) imgLabel.setIcon(new ImageIcon(image) );
}
```

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
int columnWidth = table.getColumnModel().  
    getColumn(column).getWidth();  
int redWidth = desigLabel.getPreferredSize().width;  
  
if (redWidth > columnWidth)  
{  
    panel.add(desigLabel);  
}  
else  
{  
    panel.add(desigLabel, BorderLayout.NORTH);  
    panel.add(imgLabel, BorderLayout.SOUTH);  
}  
return panel;  
}  
}
```



# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
private void ShowProducts()
{
    int nCount=0;
    Statement stmt = null;

    stmt = conn.createStatement();
    ResultSet r = stmt.executeQuery("SELECT COUNT(*) AS rowcount FROM
        produtos");
    r.next();
    nCount = r.getInt("rowcount");
    r.close();

    DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
    model.setRowCount(nCount);
    model.setColumnCount(1);
}
```

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
TableColumnModel tcm = jTable1.getColumnModel();
TableColumn tc = tcm.getColumnModel(0);
tc.setHeaderValue( "Produtos" );

((DefaultTableCellRenderer)jTable1.getTableHeader().getDefaultRenderer())
    .setHorizontalAlignment(JLabel.CENTER);

int nRow=0;
ResultSet r = stmt.executeQuery("SELECT * FROM produtos");
while (r.next())
{
    Produto p = new Produto();
    p.id=r.getLong("id");
    p.designacao = r.getString("designacao");
    p.preco = r.getDouble("preco");
    p.img = r.getString("img");

    jTable1.setRowHeight(nRow, 120);
    jTable1.setValueAt(p, nRow, 0);
    nRow++;
}
```


# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
jTable1.setDefaultRenderer(Object.class, new MultiLabelRenderer());  
stmt.close();  
}
```

Alteração a lançar MqttClient (última aula):

```
MqttClient client=new MqttClient("tcp://localhost:1883",  
MqttClient.generateClientId());  
client.setCallback( new MosquittoCallBack(this) );  
client.connect();
```



# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
public class MosquittoCallBack implements MqttCallback
{
    private MainDlg m_cMainDlg=null;

    public MosquittoCallBack(MainDlg main_dlg)
    {
        m_cMainDlg=main_dlg;
    }
    public void connectionLost(Throwable throwable)
    {
        System.out.println("Connection to MQTT broker lost!");
    }
}
```

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
public void messageArrived(String s, MqttMessage mqttMessage) throws
    Exception
{
    System.out.println("Message received:\n\t"+ new
        String(mqttMessage.getPayload())+" "+s );
    m_cMainDlg.UpdateProdutosTable(s, new String(mqttMessage.getPayload() ));
}

public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken)
{
    // not used in this example
}
}
```



# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
public void UpdateProdutosTable(String str_channel, String str_json )
{
    if(str_channel.equals("INSERT"))
    {
        Gson gson = new Gson();
        Produto p = gson.fromJson(str_json, Produto.class);

        String sql = "Insert into produtos values
("+p.id+", "+"+p.designacao+", "+"+p.preco+", "+"+p.img+"")";
        Statement stmt = conn.createStatement();
        stmt.execute(sql);
        stmt.close();
    }
}
```

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
else if(str_channel.equals("UPDATE"))
{
    Gson gson = new Gson();
    Produto p = gson.fromJson(str_json, Produto.class);

    String sql = "update produtos set designacao='"+p.designacao+"',
                preco='"+p.preco+"', img='"+p.img+"' where id='"+p.id;
    Statement stmt = conn.createStatement();
    stmt.execute(sql);
    stmt.close();
}
```

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

```
if(str_channel.equals("DELETE"))
{
    Gson gson = new Gson();
    Produto p = gson.fromJson(str_json, Produto.class);

    String sql = "delete from produtos where id="+p.id;
    Statement stmt = conn.createStatement();
    stmt.execute(sql);
    stmt.close();
}
else
{
    ;
}
ShowProducts();
}
```



# Loja em “dispositivos móveis”

Imagem exemplo  
em base64

R0lGODlhPQBEAPeoAJosM//AwO/AwHVYZ/z595kzAP/s7P+goOXMv8+fhw/v739/f+8PD98fH/8mJl+fn/9ZWb8/PzWlWv///6wWGbImAPgTEMI  
mIN9gUFCEm/gDALULDN8PAD6atYdCTX9gUNKlj8wZAKUsAOzZz+UMAOsJAP/Z2ccMDA8PD/95eX5NWvsJCOVNQPtfX/8zM8+QePLI3  
8MGBr8JCP+zs9myn/8GBqwpAP/GxgwJCPny78lZYLgIAJ8vAP9fX/+MjMUcAN8zM/9wcM8ZGcATEL+QePdZWf/29uc/P9cmJu9MTDImIN+/r  
7+/vz8/P8VNQGNugV8AAF9fX8swMNgTAFIDoICAgPNSUnNWSMQ5MBAQEJE3QPIGAM9AQmQgC9vb6MhJsEdGM8vLx8fH98AANIW  
AMuQeL8fABkTEPPQ0OM5OSYdGF15jo+Pj/+pqcsTE78wMFNGQLYmID4dGPvd3UBAQJmTkP+8vH9QUK+vr8ZWSHhpcJmMLdwcLOGcH  
RQUHxwcK9PT9DQ0O/v70w5MLypoG8wKOuwsP/g4P/Q0lcwKEswKMl8aJ9fX2xjdOtGRs/Pz+Dg4GImIP8gIH0sKEAwKKmTiKZ8aB/f39Wsl  
+LFt8dgUE9PT5x5aHBwcP+AgP+WltdgYMyZfyywz78AAAAAAD///8AAP9mZv///wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AA  
AA  
AA  
AA  
AA  
AA  
AA  
AA  
AAAAA9AEQAAj/AFEJHEiwoMGDCBMqXMiwocAbBww4nEhxoYkUpzJGrMixogkfGUNqlNixJEIDB0SqHGmyJSojM1bKZOmyop0gM3Oe2l  
iTISKMOoPy7GnwY9CjIYcSRYm0aVKSLmE6nfq05QycVLPuhDrxBlCtYJUqNAq2bNWEBj6ZXRuYxZyDRtqwnXvkhACDV+euTeJm1Ki7A73  
qNWtFiF+/gA95Gly2CJLDhwEHMOUAuOpLYDEgBxZ4GRTIC1fDnpkM+fOqD6DDj1aZpITp0dtGCDhr+fVuCu3zlg49ijaokTZTo27uG7Gjn2  
P+hI8+PDPERoUB318bWbfAJ5sUNFcuGRTYUqV/3ogfXp1rWIMc6awJjiAAd2fm4ogXjz56aypOoIde4OE5u/F9x199dlXnnGiHZWEYbGpsAEA  
3QXYnHwEFliKAgswgJ8LPeiUXGwedCAKABACCN+EA1pYIYafLcDhytd51sGAJbo3onOpajiihlO92KHGaUXGwWjUBChjSPiWJuOO/LYIm  
4v1tXfE6J4gCSJEZ7YgRYUNrkji9P55sF/ogxw5ZkSqIDaZBV6aSGYq/1GZplndkckZ98xoICbTcIJGQAZcNmdmUc210hs35nCyJ58fgmIKX5RQG  
OZowxaZwYA+JaoKQswGijBV4C6SiTumpphMspJx9unX4KaimjDv9aaXOEbteBqmuuxgEHoLX6Kqx+yXqqBANsgCtit4FWQAEkrNbpq7HS  
Omtwag5w57GrmlJBASEU18ADjUYb3ADTinIttsGsb1oJffA63bduimuqKB1keqwUhoCSK374wbujvOSu4QG6UvxBRydcPksav++Ca6G8A6Pr1  
x2kVMYHwsVxUALDq/knrhPSOzXG1IUTIoFFqGR7Goi2MAxbv6O2kEG56l7CSIRsEFKFVYovDJoIRTg7sugNRDGqCJzJgcKE0ywc0ELm6KB  
CCJo8DIPFeCWNGcyqNFE06ToAfV0HBRgxsvLThHn1oddQmRjXj5DyAQgJEHSAJMWZwS3HPxT/QMbabI/iBCliMLEJKX2EEkomBAUCxRi4  
2VDADxyTYDVogV+wSCHqmKxEKCDAYFDFj4OmwbY7bDGdDbhtrnTQYOigeChUmc1K3QTnAUfEgGFgAWt88hKA6aCRIXhxnQ1yg3BCa  
yK44EWdkUQcBBYEQChFXfCB776aQsG0BIIQgQgE8qO26X1h8cEUep8ngRBnOy74E9QgRgEAC8SvOfQkh7FDBDmS43PmGoLiKUUEGkME  
C/PJHgxw0xH74yx/3XnaYRJgMB8obxQW6kL9QYEJ0FIFgByfIL7/IQAlvQwEpnAC7DtLNJCKUoO/w45c44GwCXiAFB/OXAATQryUxdN4Lf  
FiwgjCNYg+kYMIEFkCKDs6PKAIJouyGWMS1FSKJOMRB/BoIxYJIUXFUxNwolKEKPAgCBZSQHQ1A2EWDfDEUvLyADj5AChSIQW6gu  
10bE/JG2VnCZGfo4R4d0sdQoBAHhPjhIB94v/wRoRKQWGRHgrhGSQJxCS+0pCZbEhAAOw==

# Loja em “dispositivos móveis”

## ► Cliente Java: Sincronização da DB

