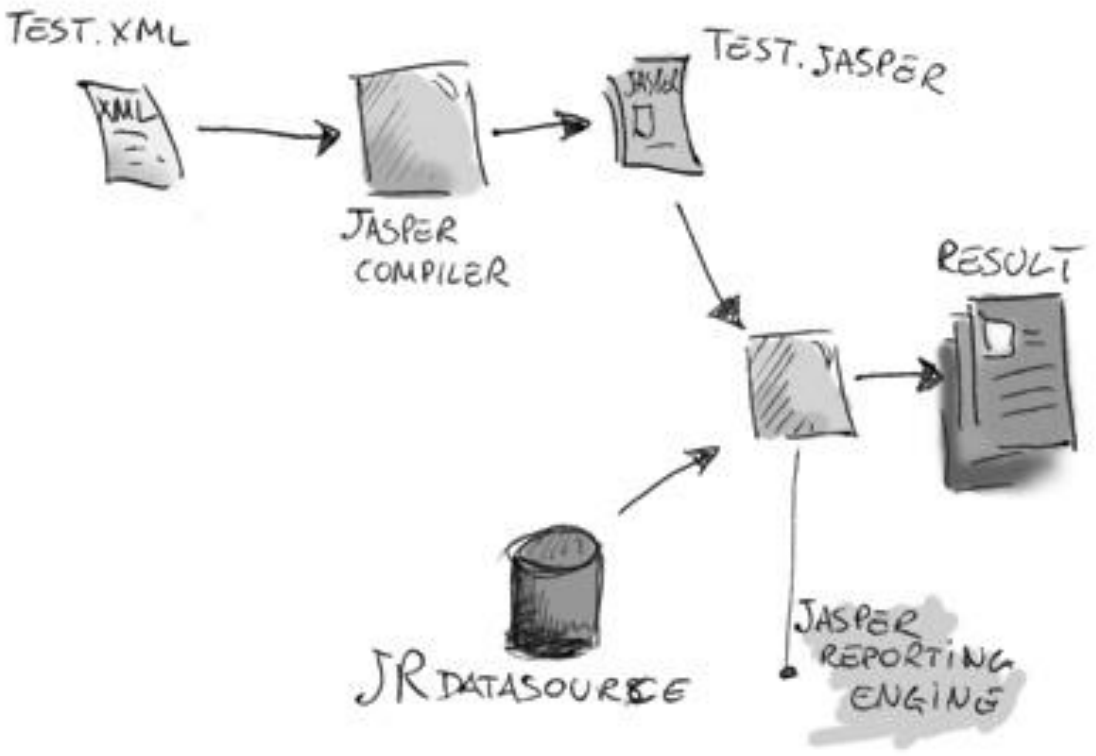
# *Desenvolvimento De Aplicações Para Web*

**1)** Qual a diferença entre **JasperReport** e **Jaspersoft Studio** ?

**Jasper Report -** Biblioteca que Compila os Relatórios.

**Jaspersoft Studio -** Ferramenta que facilita a Geração do Relatório (Parte Visual).

**2)** Explique o que está acontecendo na Imagem.




O Arquivo XML (**Test XML**) - Contém todas as configurações do relatório, usa o **Jasper Compiler**, Cria o Arquivo Executável do relatório (**Test Jasper**), para executar usa o **Jaspersoft Engine**, conecta com banco de dados, e por fim gera o resultado final.

Todas as partes são Obrigatórias para se Executar (Funcionar).

**3)** Para que serve o **SubReport** ?

Ele é responsável por fazer um sub relatório, ou seja, traz os itens de um relatório e anexa em outro relatório, ou seja, um relatório com outro anexado dentro dele.

---

**4)** **JasperReport** foi criado em qual tecnologia, e onde ele pode ser aplicado?

**JasperReport** é criado (feito) em Java e pode ser utilizado em todas as tecnologias.

---

**5)** Porque as configurações relacionada ao **JasperReport** são feitas separadas do arquivo de configuração do projeto?

A conexão é feita a parte, porque pode ser feitos relatórios de qualquer banco de dados.

---

## ❖ *Relatórios*

### controle/RelatorioControle.java

```
public void gerarRelatorioVendas() {
    try {
JasperReport relatorio;
String arquivoJasper = "relVenda.jasper";
FacesContext facesContext = FacesContext.getCurrentInstance();
facesContext.responseComplete();
ServletContext scontext = (ServletContext) facesContext.getExternalContext().getContext();
HashMap p = new HashMap();
p.put( key:"dtInicial",  value:dataInicial);
p.put( key:"dtFinal",  value:dataFinal);
```

**HashMap -** Passa os Parâmetros dos Filtros.

```
JasperPrint jasperPrint = JasperFillManager.fillReport(
( sourceFileName:scontext.getRealPath("/WEB-INF/reports/" + arquivoJasper),
params:p,  connection:Conexao.getConnection());
```

**JasperFillManager -** Arquivo do Relatório, Conexão com banco, Filtros.

```
ByteArrayOutputStream dadosByte = new ByteArrayOutputStream();
JRPdfExporter exporter = new JRPdfExporter();
exporter.setParameter( parameter:JRExporterParameter.JASPER_PRINT,  value:jasperPrint);
exporter.setParameter( parameter:JRExporterParameter.OUTPUT_STREAM,  value:dadosByte);
exporter.exportReport();
```

**JRPdfExporter -** Exporta o **JasperReport** como **PDF**.

```
byte[] bytes = dadosByte.toByteArray();
if (bytes != null && bytes.length > 0) {
    int recorte = arquivoJasper.indexOf( str:".");
    String nomePDF = arquivoJasper.substring( beginIndex:0,  endIndex:recorte);
    HttpServletResponse response = (HttpServletResponse) facesContext.getExternalContext().getResponse();
    response.setContentType( type:"application/pdf");
    response.setHeader( name:"Content-disposition", "inline; filename=\"" + nomePDF + ".pdf\"");
    response.setContentLength( len:bytes.length);
    ServletOutputStream outputStream = response.getOutputStream();
    outputStream.write( b:bytes,  off:0,  len:bytes.length);
    outputStream.flush();
    outputStream.close();
}
```

**Byte** para Frente Gera o **Relatório**.

---

## ❖ *Conexão (Classe de Conexão .Java)*

### util/Conexao.java

```
package util;

import java.io.Serializable;
import java.sql.Connection;
import java.sql.DriverManager;

public class Conexao implements Serializable {

    private static Connection con = null;

    public static Connection getConnection() {
        if (con == null) {
            try {
                String driver = "org.postgresql.Driver";
                String url = "jdbc:postgresql://localhost:5432/erpweb2023";
                String usuario = "postgres";
                String senha = "postgres";
                Class.forName( className:driver);
                con = DriverManager.getConnection(url,  user:usuario,  password:senha);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return con;
    }
    public static void closeConnection() {
        if (con != null) {
            try {
                con.close();
                con = null;
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```