

# Entregable 5: Jaspersoft Informes

## Ejercicio 1:

Plantilla informe JRXML:

The screenshot displays the Jaspersoft Studio IDE with a JRXML report template named 'informe\_clientes.jrxml'. The report is titled 'Listado de Clientes' and is structured as follows:

- Title:** Listado de Clientes
- Page Header:** Fecha: new java.util.Date()
- Column Header:** A table with two columns: Nombre and Telefono.
- Detail 1:** A table with two columns: \$F{nombre} and \$F{telefono}.
- Column Footer:** (Empty)
- Page Footer:** "Page " + \$V{PAGE\_NUMBER}
- Summary:** (Empty)

The report is displayed in the Design view. The bottom panel shows the Report State, which includes the following statistics:

Report State	Value
Compilation Time	0.148 sec
Filling Time	0.755 sec
Report Execution Time	0.931 sec
Export Time	0 sec
Total Pages	1 pages
Processed Records Count	4 records
Fill Size	0 bytes
Data Queried At	Mon Feb 16 16:08:16 CET 2026
Used Data From Snapshot	No
Data Snapshot File	

Compilar JRXML y rellenar:

```

private final String path = "src/main/resources/informe_clientes.jrxml";

public JasperReport readAndCompileFile() {
    JasperReport compiledReport = null;

    try {
        compiledReport = JasperCompileManager.compileReport(path);
        System.out.println("Archivo compilado con éxito mi rey");

    } catch (JRException e) {
        System.out.println("Error al compilar el archivo JRXML: " + e.getMessage());
        e.printStackTrace();
    }
    return compiledReport;
}

public JasperPrint fillFile(JasperReport jasperReport) {
    Map<String, Object> map = new HashMap<>();
    JasperPrint jasperPrint = null;
    try {
        jasperPrint = JasperFillManager.fillReport(jasperReport, map, ConnectionJDBC.connection());
        System.out.println("Archivo llenado con éxito");
    } catch (JRException e) {
        System.out.println("Error al llenar el archivo JRXML; " + e.getMessage());
        e.printStackTrace();
    }
    return jasperPrint;
}

```

## Exportar a PDF:

```

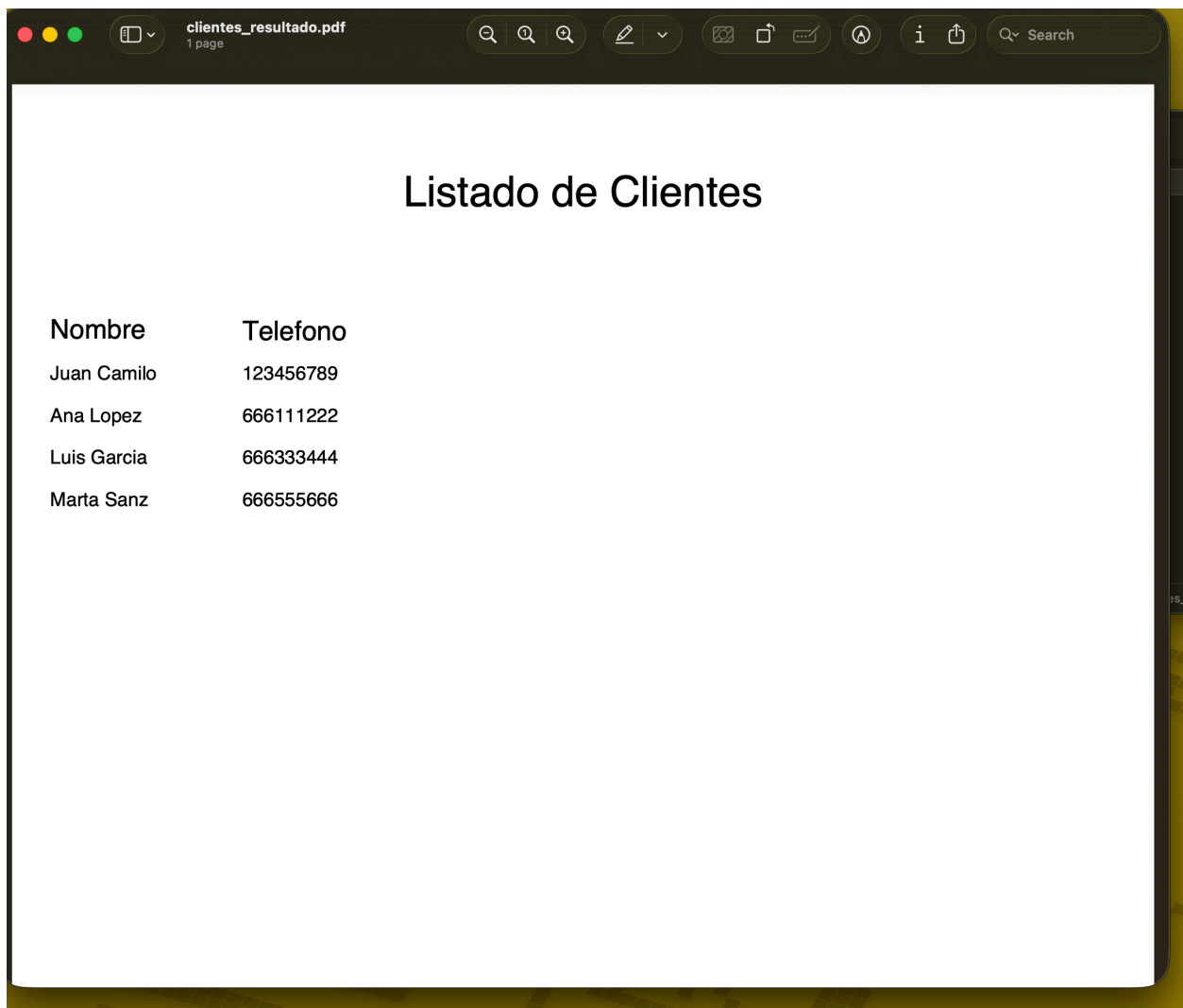
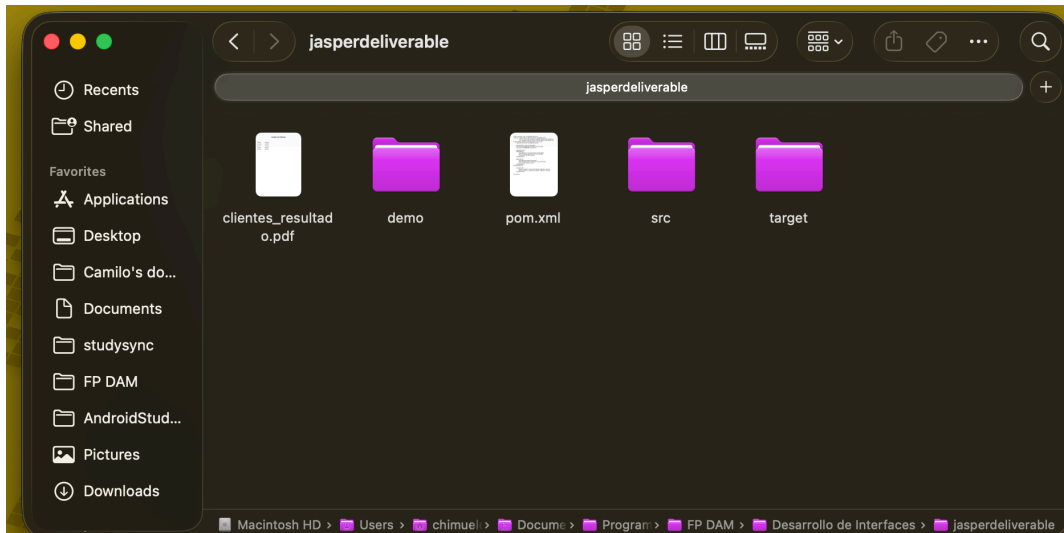
public void exportToPdf(JasperPrint jasperPrint) {

    String rutaSalida = "clientes_resultado.pdf";

    try {
        JasperExportManager.exportReportToPdfFile(jasperPrint, rutaSalida);
        System.out.println(";PDF generado correctamente en: " + rutaSalida + "!");
    } catch (JRException e) {
        System.err.println("Error al exportar a PDF: " + e.getMessage());
        e.printStackTrace();
    }
}

```

Como no se especifica una ruta de salida, el programa lo guarda en la misma carpeta del proyecto:

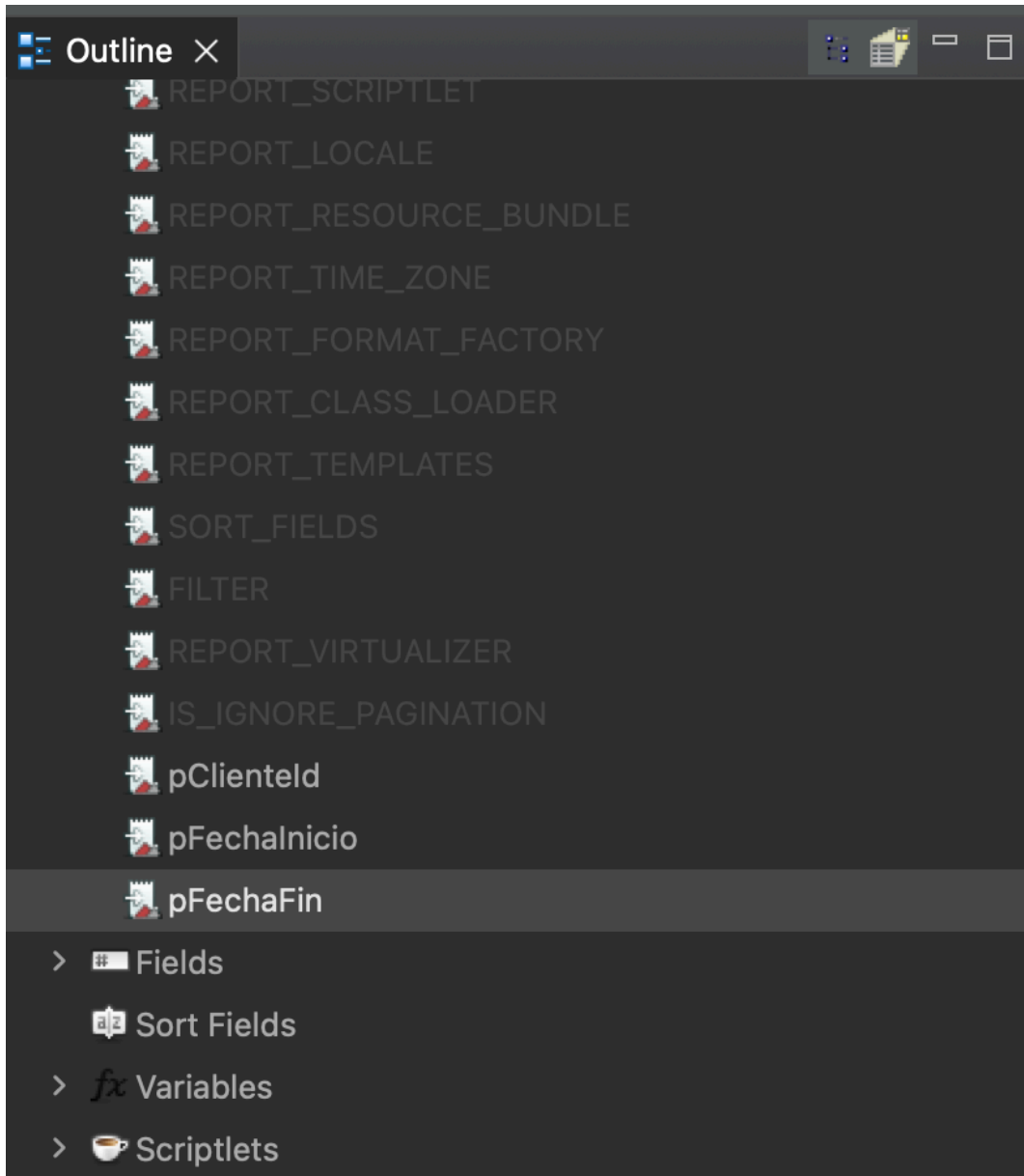


## Ejercicio 2:

### Definir parámetros:

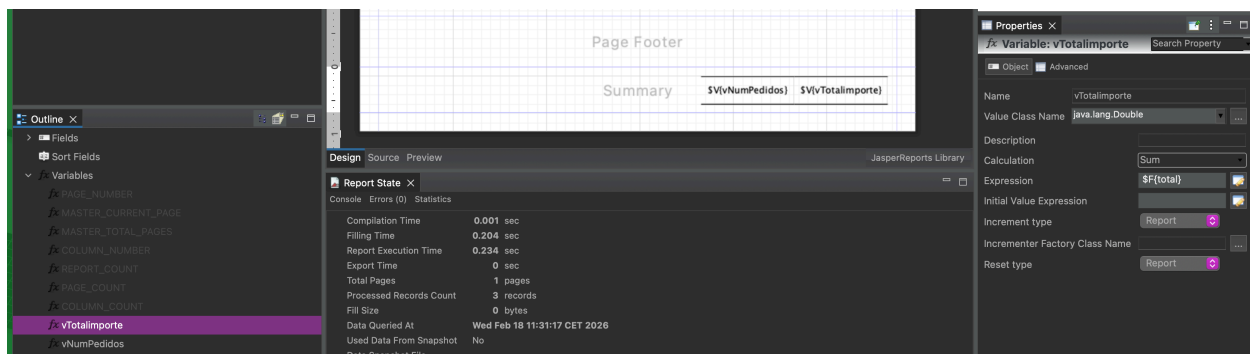
Definí los 3 parametros dichos en el enunciado, estos son los que se usan en ReportService en el metodo

`fillFile()` a través de un `Map<>`



### Creación de Variables:

Se crearon las variables, les agregué una función de `"Sum"` y `"Count"` para evitar hacer esto en código en Java



## Creación del Map<String, Object>

Lo que se pasó como "Object" fue la fecha de inicio y fin, esto se conectó con las variables creadas en JasperSoft

```
public static void main(String[] args) {
    ReportService reportService = new ReportService();

    JasperReport report = reportService.readAndCompileFile();

    if (report != null) {
        for (int i = 1; i <= 3; i++) {
            Map<String, Object> parametros = new HashMap<>();

            if (i == 1) {

                parametros.put("pClienteId", 1);
                parametros.put("pFechaInicio", Date.valueOf("2000-01-01"));
                parametros.put("pFechaFin", Date.valueOf("2099-12-31"));
            } else if (i == 2) {

                parametros.put("pClienteId", 3);
                parametros.put("pFechaInicio", Date.valueOf("2000-01-01"));
                parametros.put("pFechaFin", Date.valueOf("2026-12-31"));
            } else {

                parametros.put("pClienteId", 4);
                parametros.put("pFechaInicio", Date.valueOf("2000-01-01"));
                parametros.put("pFechaFin", Date.valueOf("2026-12-31"));
            }

            // Rellenar y exportar con un nombre diferente cada vez
            JasperPrint impresion = reportService.fillFile(report, parametros);

            if (impresion != null) {
                reportService.exportToPdf(impresion, "pedidos_reporte_" + i +
                ".pdf");
            }
        }
    }
}
```

```
}  
}
```

### Ejecución del programa 3 veces:

Para lograr esto, se creó el bucle for mostrado arriba, donde en cada iteración, crea un nuevo Map con filtros diferentes para cada cliente, y va siguiendo uno a uno el flujo de trabajo:

1. Compilar
2. Rellenar
3. Exportar a PDF