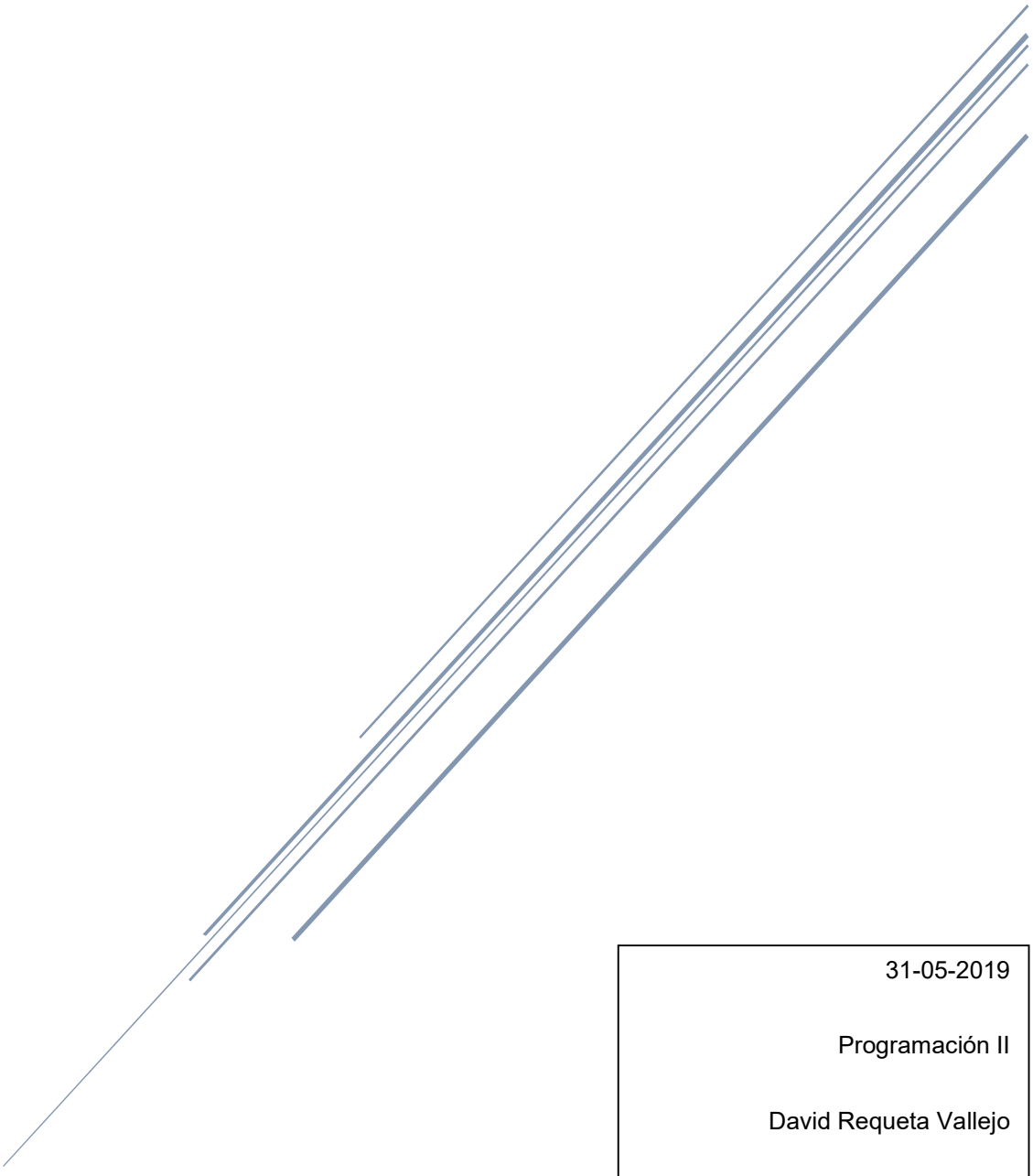
EMPRESA DE ZAPATOS



31-05-2019

Programación II

David Requeta Vallejo

Alex Salazar Herran

ÍNDICE:

General:

En esta memoria reuniremos la documentación sobre nuestra aplicación generada durante el segundo cuatrimestre del primer año de carrera de Industria Digital.

Durante nuestra estancia en la universidad hemos aprendido a desarrollar aplicaciones informáticas de la siguiente manera. Primeramente, aprendimos a estructurar la aplicación internamente en tres paquetes, de tal forma cada uno de los paquetes recogiese parte de las funcionalidades de dicha aplicación.

Por otra parte, aprendimos a gestionar las dependencias entre paquetes de tal forma que la aplicación fuese mas legible y su mantenimiento fuese más sencillo.

A esto tenemos que añadirle otros capos más técnicos como son la creación de pantallas para que el operario interactúe con la aplicación de forma sencilla e intuitiva, tratamiento de excepciones, implementación de interfaces para tratar de otra forma los diferentes datos recogidos e incluso el establecer conexión con una base de datos para el guardado de esos datos. También hemos utilizado diversos recursos de internet, como por ejemplo código adaptado para el guardado de ciertas tablas en ficheros “.xls” y “.pdf”.

Como futuras modificaciones en la aplicación pensamos en el desarrollo de nuevos tipos de informes para que al usuario le resulte más sencillo el obtener los datos. Además, implementaremos otros métodos de introducción de datos y actualización de estos con referencia a los clientes por si el usuario no se sabe algún dato del cliente en el momento de insertarlo en la aplicación.

Código para importar a Excel y PDF:

```java
/**
 * Clase para exportar en PDF o EXCEL
 *
 * @author Alex Salazar
 * @author David Requeta
 */
public class ClsExportar {

    /**
     * Constructor
     *
     * @param table parametro recibido
     * @throws IOException lanza excepcion
     */
    public ClsExportar(JTable table) throws IOException {
        Exportar(table);
    }

    /**
     * Metodo para exportar tablas
     *
     * @param t recibe la tabla
     * @throws IOException lanza excepcion
     */
    private void Exportar(JTable t) throws IOException {
        JFileChooser chooser = new JFileChooser();
        FileNameExtensionFilter filter = new FileNameExtensionFilter("Archivos de Excel", "xls");
        FileNameExtensionFilter filter1 = new FileNameExtensionFilter("Archivos de PDF", "pdf");
        chooser.setFileFilter(filter);
        chooser.setFileFilter(filter1);
        chooser.setDialogTitle("Guardar archivo");
        chooser.setAcceptAllFileFilterUsed(false);
        if (chooser.showSaveDialog(null) == JFileChooser.APPROVE_OPTION) {
            String ruta = chooser.getFileFilter().getDescription();

            if (ruta.equals("Archivos de Excel")) {

                try {
                    ruta = chooser.getSelectedFile().toString().concat(".xls");
                    File archivoXLS = new File(ruta);
                    if (archivoXLS.exists()) {
                        archivoXLS.delete();
                    }
```

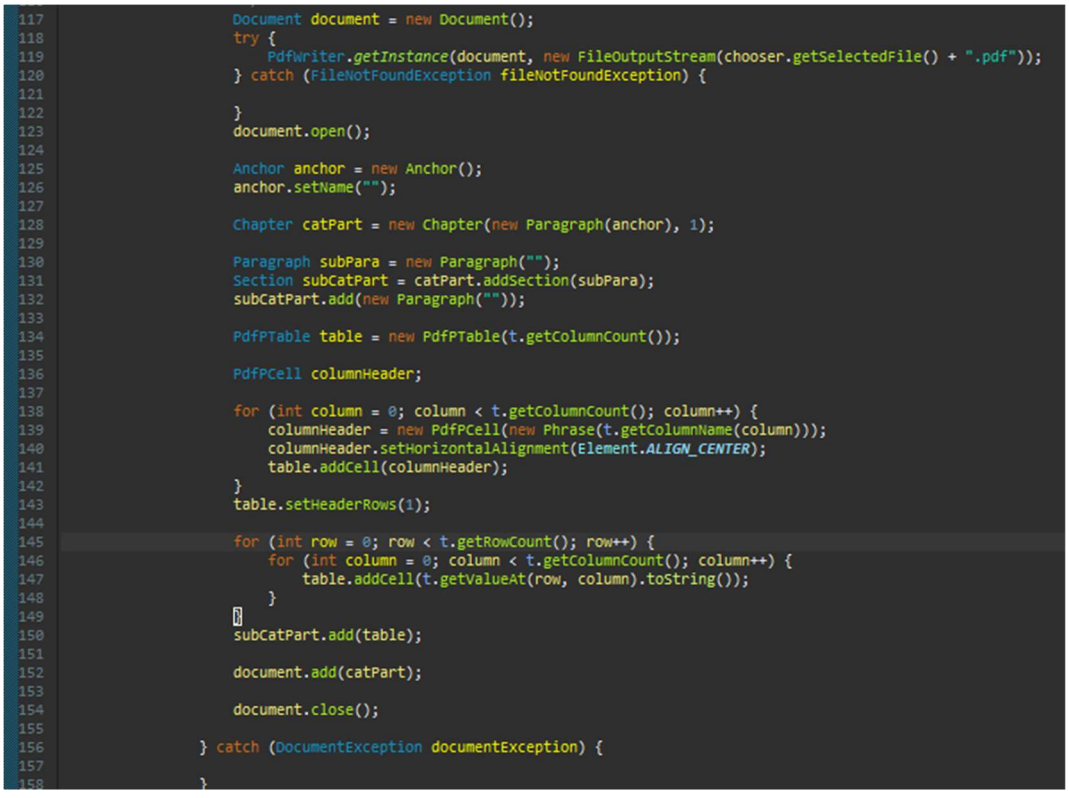
Ilustración 1: Código para importar a Excel y PDF

```
                archivoXLS.createNewFile();
                Workbook libro = new HSSFWorkbook();
                FileOutputStream archivo = new FileOutputStream(archivoXLS);
                Sheet hoja = libro.createSheet("Mi hoja de trabajo 1");
                hoja.setDisplayGridlines(false);
                for (int f = 0; f < t.getRowCount(); f++) {
                    Row fila = hoja.createRow(f);
                    for (int c = 0; c < t.getColumnCount(); c++) {
                        Cell celda = fila.createCell(c);
                        if (f == 0) {
                            celda.setCellValue(t.getColumnName(c));
                        }
                    }
                }
                int filaInicio = 1;
                for (int f = 0; f < t.getRowCount(); f++) {
                    Row fila = hoja.createRow(filaInicio);
                    filaInicio++;
                    for (int c = 0; c < t.getColumnCount(); c++) {
                        Cell celda = fila.createCell(c);
                        if (t.getValueAt(f, c) instanceof Double) {
                            celda.setCellValue(Double.parseDouble(t.getValueAt(f, c).toString()));
                        } else if (t.getValueAt(f, c) instanceof Float) {
                            celda.setCellValue(Float.parseFloat((String) t.getValueAt(f, c)));
                        } else {
                            celda.setCellValue(String.valueOf(t.getValueAt(f, c)));
                        }
                    }
                }
                libro.write(archivo);
                archivo.close();
            } catch (IOException | NumberFormatException e) {
                throw e;
            }
        }

        if (ruta.equals("Archivos de PDF")) {
            try {

                /**
                 * Creamos el documento e indicamos el nombre del fichero.
```

Ilustración 2: Código para importar a Excel y PDF



```
                Document document = new Document();
                try {
                    PdfWriter.getInstance(document, new FileOutputStream(chooser.getSelectedFile() + ".pdf"));
                } catch (FileNotFoundException fileNotFoundException) {

                }
                document.open();

                Anchor anchor = new Anchor();
                anchor.setName("");

                Chapter catPart = new Chapter(new Paragraph(anchor), 1);

                Paragraph subPara = new Paragraph("");
                Section subCatPart = catPart.addSection(subPara);
                subCatPart.add(new Paragraph(""));

                PdfPTable table = new PdfPTable(t.getColumnCount());

                PdfPCell columnHeader;

                for (int column = 0; column < t.getColumnCount(); column++) {
                    columnHeader = new PdfPCell(new Phrase(t.getColumnName(column)));
                    columnHeader.setHorizontalAlignment(Element.ALIGN_CENTER);
                    table.addCell(columnHeader);
                }
                table.setHeaderRows(1);

                for (int row = 0; row < t.getRowCount(); row++) {
                    for (int column = 0; column < t.getColumnCount(); column++) {
                        table.addCell(t.getValueAt(row, column).toString());
                    }
                }
                subCatPart.add(table);

                document.add(catPart);

                document.close();

            } catch (DocumentException documentException) {

            }
```

Ilustración 3: Código para importar a Excel y PDF

Aspectos técnicos:

Contexto y motivación de la aplicación:

Para nuestra aplicación tomamos como referencia una pequeña empresa de zapatos donde nuestro cliente era el único que gestionaba una base de datos de manera algo complicada y costosa en tiempo. Nos solicitaba una solución en la que pudiese emplear menos tiempo para realizar toda la inserción y consulta de datos en su ordenador personal. Además, a ello le quería añadir la posibilidad de actualizar el estado de los pedidos (Entregado o No Entregado) y poder sacar unos informes para saber que debía de hacer durante sus horas de producción.

Estructura de la aplicación desarrollada:

La aplicación esta principalmente estructurada en 9 paquetes con los siguientes nombres (default package, comparadores_y_comprobadores, COMUN, excepciones, LD, LN, LP, PANTALLA_LP y tablas).

En el primer paquete “default package” recogemos la clase principal de la aplicación de la cual se ejecuta. En el segundo paquete hemos reunido todas las clases que nos sirven para comparar ciertos atributos de los objetos generados para poder ordenarlos. Además, tenemos una clase para comprobar (“ClsComprobarDNI_NIF”) que se encarga de validar los DNIs o NIFs introducidos por el cliente. En el paquete de COMUN recogemos una clase y una interfaz que son las encargadas de la gestión de datos con respecto a los paquetes LP. En el siguiente paquete recogemos todo lo relacionado con las posibles excepciones que se puedan dar durante el uso o reprogramación de la aplicación, de tal forma que el cliente no pueda hacer cosas ilógicas que terminen por hacer cascar la aplicación, o que en futuras modificaciones no pueda programarse cosas que provoquen fallos.

Los siguientes paquetes (LD, LN, LP y PANTALLA_LP) son los que reúnen las clases principales que gestionan la aplicación. En LD reunimos la clase principal para la conexión con la base de datos y otra clase donde reunimos las estructuras necesarias para insertar, consultar y borrar datos de tal forma que hacemos mas limpia la programación de la clase ClsDatos. En LN reunimos todas las clases encargadas de gestionar los datos introducidos por el usuario. En este

paquete encontramos la clase principal de la aplicación (ClsGestorLN) que se encarga masivamente de la gestión de la aplicación como por ejemplo ordenar información devolverla o mandarla guardar. El resto de las clases se encargan de generar los objetos de los datos obtenidos. Por último, encontramos LP, PANTALLA_LP y tablas. Estos paquetes reúnen todas las clases encargadas de interactuar con el usuario. De primeras generamos LP que trabajaba contra la consola del entorno de programación. Una vez que pasamos a desarrollar interfaces gráficas para que la interactuación del usuario con la aplicación fuese más sencilla e intuitiva desarrollamos PANTALLA_LP. El paquete tablas reúne todos los modelos de las tablas generadas para mostrarle al usuario la información que nos pida.

Detalles técnicos:

A nivel de técnicas empleadas podemos separarlas en dos partes. Por una parte, consideramos tener las técnicas utilizadas en la programación de la aplicación. Estas reúnen ciertas cosas aprendidas en clase como las que hemos mencionado anteriormente (Excepciones, métodos de ordenamiento, algoritmos de comprobación, etc.). A parte de esas técnicas también hemos utilizado otras como métodos implementados para comprobar objetos repetidos, estructuras de código o algoritmos propios o buscados para desarrollar ciertas funcionalidades.

Código para buscar entre las fechas que nos den:

```
for (ItfProperty a : PedidosNoEntegados) {
    String AComparar = String.valueOf(a.getDateProperty(PROPIEDAD_PEDIDOS_FECHA_DE_ENTREGA));

    FormatearFechaEntregaInicio();
    FormatearFechaEntregaFinal();

    int antes = miFormato.format(FechaDeEntregaInicio).compareTo(AComparar);
    int despues = miFormato.format(FechaDeEntregaFinal).compareTo(AComparar);

    if (antes <= 0 && despues >= 0) {
        PedidosEntreFechas.add(a);
    }

}
```

Ilustración 4: código para buscar entre fechas ciertos pedidos

No obstante, desde nuestro punto de vista lo más interesante que hemos desarrollado es la carga de datos en memoria para que la aplicación sea más rápida que trabajando contra disco. De esta forma el único acceso que realizamos a disco es a la hora de introducir y borrar los datos.

Hemos elegido estas técnicas ya que creemos que son las mas sencillas de implementar, que los procesos que realizan se realizan de forma rápida y que ayudan a que la aplicación sea mas atractiva a nivel de funcionalidad.

Por otra parte, contemplaríamos las técnicas utilizadas a nivel de interfaz gráfica. Creemos que este punto es bastante importante ya que es con lo que el cliente va a operar y por consiguiente tiene que ser agradable, cómoda, sencilla e intuitiva. Para ello, hemos hecho uno de áreas para insertar texto, desplegables para seleccionar número de referencias. Estos desplegables van acompañados de botones o tablas los cuales a través de códigos de programación permiten al usuario tener constantemente la información cerca para poder cotejarla con todo lo que hace y le resulte más cómoda y manejable.

Creemos haber llevado un buen desarrollo a lo largo de la aplicación, habiendo conseguido una aplicación realmente cómoda para su uso, sobre la cual podríamos seguir desarrollando ciertos aspectos para que el día de mañana cumpliese con todas las expectativas que nos pudiésemos o nos pudiesen plantear.

Planificación:

Dentro del apartado de planificación hemos ido recogiendo en un Excel todos los contratiempos que hemos tenido a lo largo del desarrollo de la aplicación, así como las horas invertidas en dichos contratiempos.

Dentro de los motivos de las posibles desviaciones podría decirse que se debe a la falta de experiencia en este lenguaje de programación, ya que muchas de las cosas que se pedían realizar en el proyecto eran nuevas y desconocidas. Esto nos supuso un gran reto a la hora de buscar las posibles soluciones puesto que tenías que investigar por internet, debatir con otros compañeros como lo habían realizado e incluso preguntando al profesor hasta que finalmente dabas con la solución.

Esto nos hizo ser más autónomos a la hora de solucionar problemas en vez de ir a preguntar al profesor directamente. Además, nos permitió ir poco a poco comprendiendo lo que íbamos realizando, llegando así a realizar la aplicación incluso añadiendo cosas extra.

Respecto a los objetivos fijados se podría decir que hemos cumplido con todos ellos en el plazo establecido a pesar de los contratiempos.

Como futuras mejoras estaría la depuración no contemplada durante el desarrollo de la aplicación además de las modificaciones y ampliaciones que el cliente pudiera optar oportunas.

Índice de figuras:

Bibliografía:

Principalmente la información a cerca de los errores o líneas de código implementadas has sido procedentes de las siguientes páginas de internet:

- https://stackoverflow.com/
- https://alud.deusto.es/course/view.php?id=8518
- https://alud.deusto.es/mod/forum/view.php?id=427779

No obstante, también hemos obtenido código y solucionados errores consultando a diferentes compañeros o a nuestro profesor de la asignatura de Programación II.