Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Sistema para la gestión de estadísticas sobre la operación de equipos de cómputo.

Modalidad: Estancia Profesional

Trimestre 2018 – Primavera

Ricardo Marin Tinoco

2142001207

al2142001207@correo.azc.uam.mx

Asesor

M. en C Josué Figueroa González

Profesor Asociado

Departamento de Sistemas

jfgo@correo.azc.uam.mx

Coasesora

M. Celia Martínez Melchor

Jefe de División de Ingeniería Biomédica

División de Ingeniería Biomédica
del Instituto Mexicano del Seguro Social
celia.martinezm@imss.gob.mx

23 de julio de 2018

Declaratoria

Yo, Josué Figueroa González, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Josué Figueroa González

Asesor

Yo, Celia Martínez Melchor, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Celia Martínez Melchor

Coasesora

Yo, Ricardo Marin Tinoco, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Ricardo Marin Tinoco

Alumno

Resumen

En este documento, se presenta el proceso de desarrollo de una aplicación Web para un Sistema de Gestión. Este sistema es capaz de recopilar datos en base a estadísticas y presentarlas en forma de gráficas, en el navegador o en un archivo pdf, de manera que las gráficas se puedan almacenar o desechar a consideración del usuario. El Sistema de Gestión está compuesto por los siguientes módulos:

- Módulo de búsqueda: encargado de buscar información de los equipos de cómputo.
- Módulo de análisis de datos: funciona como intermediario entre los módulos de búsqueda y visualización, recopilando información relevante para enviar al módulo de visualización.
- Módulo de visualización: se encarga de presentar la información en diferentes tipos de gráficas.
- Módulo de reportes: encargado de generar reportes permitiendo añadir las gráficas a dichos.

Además, se describe también las bibliotecas, externas de Java, utilizadas para la creación del proyecto, como jFreeChart y JasperReports. Para finalizar se describe los resultados y conclusiones obtenidos, como realización de este proyecto.

Tabla de Contenido

Tabla de Figuras

Introducción

En la actualidad, en cualquier empresa ya sea pública o privada, el área de cómputo es una de las más importantes, y esto se debe no solo a que estamos en el auge de la tecnología, sino que también a las ventajas que nos brindan los equipos de cómputo porque ayudan en las tareas diarias que se realizan en una institución, además permiten realizar más actividades de mejor forma en un menor tiempo.

Dentro de las dependencias de gobierno esto no es la excepción, los equipos de cómputo y los sistemas de información permiten ofrecer un mejor servicio, del mismo modo que les facilita el llevar un inventario con un control más adecuado. Sin embargo, no es suficiente con tener un manejo del inventario, también es necesario que se lleve un control estadístico de la operación de los equipos de cómputo. Esto se puede obtener gracias a los datos generados en la institución que permiten saber sus áreas de oportunidad a través de indicadores de los cuales se pueden presentar gráficas que muestran la productividad y estado de cada de sus equipos, por ejemplo, cuáles se deben cambiar con mayor frecuencia, cada cuánto se utilizan o en qué momento deben desecharse.

En el presente trabajo se propone crear un software que se encargue de manejar las estadísticas sobre la operación de la infraestructura de cómputo que se utiliza dentro de una dependencia del Instituto Mexicano del Seguro Social (IMSS) la finalidad de presentar los indicadores en un tiempo menor y de manera más clara para su análisis.

Antecedentes

Sistema de Gestión del Repositorio de Software en Tecnologías Abiertas De la Carrera de Ingeniería en Computación de la UAM-Azcapotzalco [1].

Es un software encargado de administrar un repositorio el cual es capaz de realizar búsquedas y generar reportes de manera similar al proyecto propuesto. La diferencia es que el sistema propuesto podrá generar gráficas que muestren las estadísticas de la operación de los equipos de cómputo.

Análisis y gestión de recursos para brindar seguridad en una red empresarial [2].

Se encarga de analizar el conjunto de elementos que forman parte de la red empresarial para implementar políticas de seguridad que disminuyan la vulnerabilidad de la red. La relación es que ambos proyectos permiten consultar estadísticas sobre recursos, la diferencia es que este proyecto maneja otro tipo de datos y genera gráficas con las estadísticas.

Sistema basado en web para la gestión de capital intelectual referente a la organización de las metodologías de las TIC [3].

Este sistema solo se relaciona a través de su módulo de reportes que se encarga de mostrar cuales son los cursos más consultados y la frecuencia de visita de los usuarios ya que su función principal es vender cursos. La diferencia es que el proyecto propuesto tiene dentro de sus funciones generará reportes a partir de un manejo de estadísticas de mayor cantidad de variables, en este caso los equipos de cómputo.

Juice Labs [4].

Es un servicio web que permite crear diferentes tipos de gráficas con la ayuda del software Excel, sin embargo, no permite analizar datos ni generar reportes.

IBM Cognos Analytics on Cloud [5]

Servicio Web enfocada en el área empresarial que brinda la facilidad de llevar un control seguro de datos, así como de generar informes a través de información de valor, la diferencia con este software es que es web y el propuesto no lo es.

IBM Db2 [6]

Este software permite guardar datos como procesarlos de manera metódica a través de modelos o de herramientas como R que permiten el análisis y la representación de datos de forma gráfica [7], la diferencia con el propuesto es que no genera reportes.

Justificación

Actualmente en la dependencia del IMSS en donde se realiza la estancia profesional, cuando se desean consultar estadísticas sobre el uso de equipos, estas se deben consultar en hojas almacenadas en grandes carpetas, lo que toma mucho tiempo y resulta en un trabajo ineficiente y complejo, e incluso, en ocasiones es imposible de realizar, ya que los datos se han perdido o nunca se registraron.

Por este motivo es necesario crear un software que ayude a resolver los problemas relacionados con la consulta de estadísticas sobre la operación del equipo de cómputo. El proyecto propuesto ayudará a que los responsables de la infraestructura del área conozcan cómo se comportan los equipos que tienen a su cargo de manera rápida y clara y esto los ayudará en la toma de decisiones acerca del mantenimiento, compra o retiro de equipo.

Objetivos

Objetivo general

- Diseñar e implementar un software para la gestión de estadísticas sobre el uso de equipos de cómputo.

Objetivos particulares

- Diseñar e implementar un módulo de búsqueda de información de los equipos de cómputo.
- Diseñar e implementar un módulo de análisis de datos.
- Diseñar e implementar un módulo que visualice las estadísticas de los equipos.
- Diseñar e implementar un módulo que genere reportes en base a los datos procesados.

Marco Teórico

1. Sistema de Gestión

Un sistema de gestión es una herramienta que permite optimizar recursos, reducir costos y mejorar la productividad en tu empresa. Los sistemas de gestión sirven para reportar datos en tiempo real que permiten la toma de decisiones para corregir fallos y prevenir la aparición de gastos innecesarios.

Los sistemas de gestión están basados en normas internacionales que permiten controlar distintas facetas en una empresa, como la calidad de su producto o servicio, los impactos ambientales que pueda ocasionar, la seguridad y salud de los trabajadores, la responsabilidad social o la innovación.

Un sistema de gestión está especialmente recomendado a cualquier tipo de organización o actividad orientada a la producción de bienes o servicios, las cuales puedan encontrar en la gestión de sistemas una herramienta útil para mejorar su empresa.

2. Biblioteca JFreeChart

Es una biblioteca de gráficos gratuita basada en Java que facilita la visualización de gráficos de alta calidad e incorporarlos a cualquier aplicación.

La biblioteca JFreeChart incluye:

- Documentación en la cual se describe una amplia gama de tipos de gráficos y la manipulación de los mismos.
- Diseño flexible de fácil extensión, dirigido tanto a aplicaciones del lado del servidor como del lado del cliente.
- Soporte para diferentes tipos de salida, incluidos los componentes Swing y JavaFX, archivos de imagen (incluidos PNG[1] y JPEG[2]) y formatos de archivos de gráficos vectoriales (incluidos PDF[3], EPS[4] y SVG[5]);
- JFreeChart es de software libre y es distribuida bajo los términos de la Licencia Pública General Reducida (LGPL[6]) de GNU[7], que permite el uso en aplicaciones propietarias.

Entre los gráficos principales que maneja la biblioteca JFreeChart se encuentran:

- Gráficos XY (línea, spline y dispersión).
- Gráfico circular.

[1] PNG es la abreviatura del término inglés “Portable Network Graphics”
[2] JPEG es la abreviatura del término inglés “Joint Photographic Experts Group”
[3] PDF es la abreviatura del término inglés “Portable Document Format”
[4] EPS es la abreviatura del término inglés “Encapsulated PostScript"
[5] SVG es la abreviatura del término inglés “Scalable Vector Graphics”
[6] LGPL es la abreviatura del término inglés “Lesser General Public License”
[7] GNU es la abreviatura del término inglés “GNU is Not Unix”

- Diagrama de Gantt.
- Gráficos de barras (horizontal y vertical, apiladas e independientes).
- Valores Simples (termómetro, brújula, indicador de velocidad) que luego se pueden colocar sobre el mapa.

3. Biblioteca JasperReports

La biblioteca JasperReports es un motor de informes de código abierto y es el núcleo de iReport Designer, Jaspersoft Studio y JasperReports Server. Está escrito completamente en Java y es capaz de utilizar datos provenientes de cualquier tipo de fuente de datos y producir documentos que se pueden ver, imprimir o exportar en una variedad de formatos de documentos, incluyendo HTML[8], PDF, Excel, OpenOffice y Word.

Como el sistema JasperReports Library tiene pocas dependencias externas, los únicos dos requisitos son:

- Java JDK[9] 1.6 o superior
- Controlador JDBC[10] 2.1 (si se usa RDBMS[11])

4. Mapeo Objeto Relacional (ORM[12])

El ORM consiste en vincular o mapear entidades de una base de datos relacional con clases de un lenguaje de programación orientado a objetos. Es decir, vincular registros de una Base de Datos con objetos del lenguaje.

Ventajas de utilizar ORM:

- Se hace una abstracción en cuanto a consultas SQL[13]: debido a que se trabajan con objetos y no con registros, permitiendo adaptarse a cualquier manejador de base de datos.
- Se reduce el tiempo de desarrollo: muchas herramientas ORM utilizan la ingeniería inversa para crear archivos de configuración, así como las clases necesarias, para hacer un mapeo a partir de una base de datos relacional.
- Se tienen métodos y clases ya definidos en muchas herramientas ORM, permitiendo extraer, insertar, modificar o eliminar objetos de manera más fácil.

Muchas de las aplicaciones utilizadas hoy en día, están basadas en un lenguaje de programación orientado a objetos y utilizan además una base de datos relacional, dando la posibilidad de utilizar ORM. Existen diferentes frameworks comerciales o de uso libre, como Hibernate, que nos ayudan a utilizar esta técnica de programación.

---

[8] HTML es la abreviatura del término inglés “HyperText Markup Language”
[9] JDK es la abreviatura del término inglés “Java Development Kit”
[10] JDBC es la abreviatura del término inglés “Java Database Connectivity”
[11] RDBMS es la abreviatura del término inglés “Relational Database Management System”
[12] ORM es la abreviatura del término en inglés “Object-Relational mapping”.
[13] SQL es la abreviatura del término en inglés “Structured Query Language”.

5. Hibernate

Hibernate se encarga de asignar clases de Java a las tablas de la base de datos, y de asignar los tipos de datos de Java a los tipos de datos de SQL. Además, proporciona servicios de búsqueda y recuperación de datos. Su principal objetivo es reducir significativamente el tiempo de desarrollo.

Hibernate está enfocado hacia la plataforma Java facilitando el mapeo de atributos mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL

Hibernate está determinado por las siguientes características

- Persistencia idiomática.
- Alto rendimiento.
- Escalabilidad.
- De confianza.
- Extensibilidad.

Desarrollo del proyecto

El Sistema de Gestión se encarga de recaudar información de la base de datos del área de informática con la que ya se cuenta en el IMSS, procesarla y desplegar dicha información en forma de gráficas que pueden ser desechadas o pueden ser almacenadas en forma de archivo pdf.

Los módulos con los que cuenta el sistema son:

- Módulo de búsqueda.
- Módulo de análisis de datos.
- Módulo de visualización.
- Módulo de reportes.

Los componentes del proyecto desarrollado se muestran en la Figura 1.

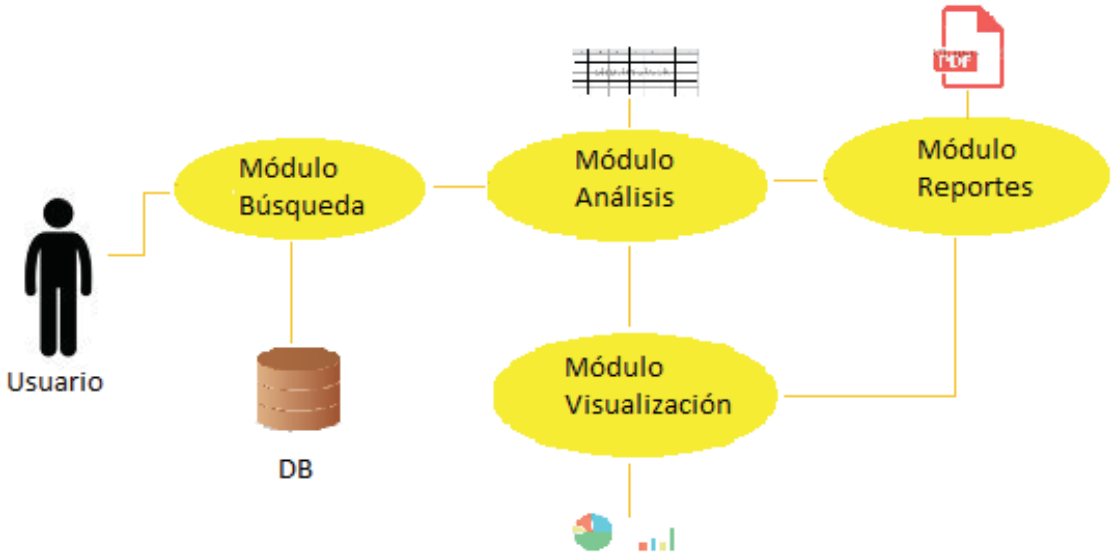

Figura 1. Vista de los Módulos del Sistema

En la Figura 2 se muestra el diagrama Entidad-Relación de la base de datos con la que se cuenta en el IMSS para la extracción y el análisis de los datos.

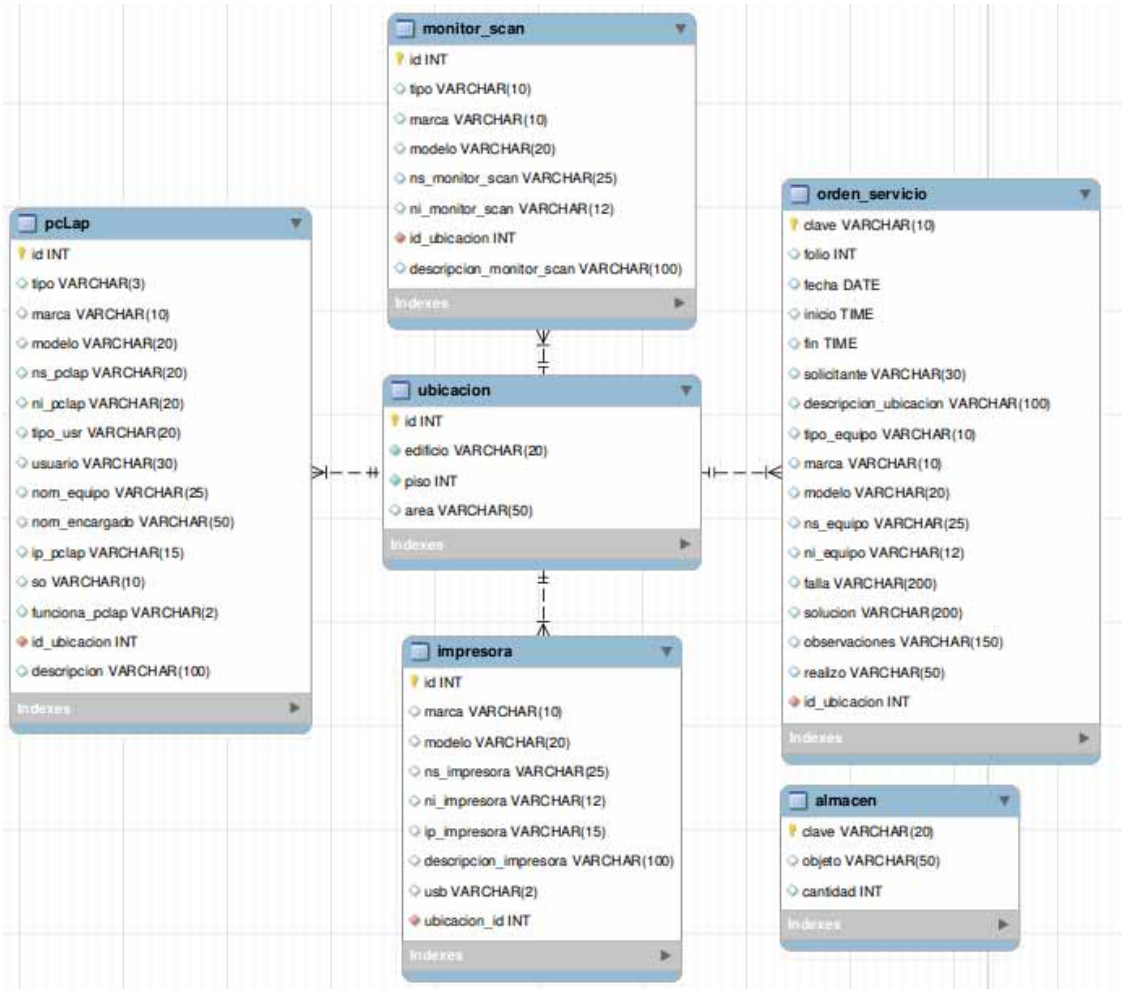

Figura 2. Diagrama Entidad-Relación de la Base de Datos

1. Módulo de búsqueda.

La tarea de este módulo es acceder a la base de datos y recopilar información sobre los datos de los equipos, así como de las ordenes de servicio para su análisis posterior.

En la Figura 3 se muestra una de las funciones del módulo de búsqueda que permite la recopilación de los datos para después enviarlos al módulo de análisis para su procesamiento.

```
public List<OrdenUbicacion> consultarOrdenFecha(String fecha1, String fecha2)
        throws ParseException {
    Session sesion = null;
    List<Object[]> list;
    List<OrdenUbicacion> listaPU = new LinkedList();
    try {
        sesion = CrearConexion.getSessionFactory().openSession();
    } catch (ExceptionInInitializerError ex) {
        System.out.println("No se pudo crear sesion");
        throw new ExceptionInInitializerError(ex);
    }

    String sentencia = "SELECT UBI.edificio, UBI.piso, UBI.area, ORD.descripcionUbicacion, "
            + "ORD.clave, ORD.folio, ORD.fecha, ORD.inicio, ORD.fin, ORD.solicitante, "
            + "ORD.tipoEquipo, ORD.marca, ORD.modelo, ORD.nsEquipo, ORD.niEquipo, "
            + "ORD.falla, ORD.solucion, ORD.observaciones, ORD.realizo "
            + "FROM OrdenServicio AS ORD, Ubicacion AS UBI "
            + "WHERE ORD.idUbicacion = UBI.id "
            + "AND (ORD.fecha BETWEEN :ordfecha1 AND :ordfecha2) "
            + "ORDER BY ORD.fecha DESC";
    Query query = sesion.createQuery(sentencia);
    query.setParameter("ordfecha1", fecha1);
    query.setParameter("ordfecha2", fecha2);
    list = query.list();
    for (int i = 0; i < list.size(); i++) {
        Object[] datosRecuperados = list.get(i);

        OrdenUbicacion orUbi = new OrdenUbicacion();
        orUbi.setEdificio((String) datosRecuperados[0]);
        //orUbi.setPiso((Integer)datosRecuperados[1]);
        String x = (String)datosRecuperados[1];
        orUbi.setPiso(Integer.parseInt(x));
        orUbi.setArea((String) datosRecuperados[2]);
        orUbi.setDescripcionUbicacion((String) datosRecuperados[3]);
        orUbi.setClave((String)datosRecuperados[4]);
        orUbi.setFolio((Integer)datosRecuperados[5]);
        orUbi.setFecha((String) datosRecuperados[6]);
        orUbi.setInicio((String) datosRecuperados[7]);
        orUbi.setFin((String) datosRecuperados[8]);
        orUbi.setSolicitante((String) datosRecuperados[9]);
        orUbi.setTipoEquipo((String) datosRecuperados[10]);
        orUbi.setMarca((String) datosRecuperados[11]);
        orUbi.setModelo((String) datosRecuperados[12]);
        orUbi.setNsEquipo((String) datosRecuperados[13]);
        orUbi.setNiEquipo((String) datosRecuperados[14]);
        orUbi.setFalla((String) datosRecuperados[15]);
        orUbi.setSolucion((String) datosRecuperados[16]);
        orUbi.setObservaciones((String) datosRecuperados[17]);
        orUbi.setRealizo((String) datosRecuperados[18]);

        listaPU.add(orUbi);

    }
    sesion.close();
    return listaPU;
}
```

Figura 3. Función “consultarOrdenFecha” del módulo de búsqueda

2. Módulo de análisis de datos.

En este módulo se examinan los datos de los equipos y generan estadísticas de ellos con el fin de obtener los indicadores requeridos como lo son:

- Las áreas que frecuentemente piden el servicio.
- Los equipos que más fallan.
- Los principales problemas en los equipos.

Todo esto en base a las áreas, equipos y periodos de tiempo.

En la Figura 4 se muestra parte de la función “BuscarUTMMSI” del módulo de análisis que se encarga de ordenar los datos obtenidos de acuerdo con la cantidad de veces que aparece cada uno de ellos y los almacena en un arreglo de cadenas con su valor para después añadirlos a la gráfica deseada

```
// Consulta las ordenes en el rango de fechas y las ordena
// de acuerdo a la clave dada de mayor numero de ocasiones
// en las que aparecen los objetos al menor
public List<OrdenUbicacion> BuscarUTMMSI(String fecha1, String fecha2, int clave)
    ConsultaOrden c = new ConsultaOrden();
    List<OrdenUbicacion> lista = c.consultarOrdenFecha(fecha1, fecha2);
    ArrayList<String> array = new ArrayList<>();
    ArrayList<Integer> valores = new ArrayList<>();
    List<Ordenar> ord = new LinkedList<>();

    String ant, actual;
    Integer e = 1;
    int t = lista.size();

    // Ordena las ordenes de acuerdo a la clave y obtiene
    // cuantas aparece cada uno de los objetos
    switch (clave) {
    case 1:
        ConsultaUbicacion cu = new ConsultaUbicacion();
        List<Ubicacion> listau = new LinkedList<Ubicacion>();

        for (int i = 0; i < t; i++){
            Ubicacion u = new Ubicacion();

            String edificio = lista.get(i).getEdificio();
             int piso = lista.get(i).getPiso();
             String area = lista.get(i).getArea();
             int id = cu.consultarUbicacion(edificio, piso, area);
             System.out.println(edificio + " " + piso + " " + area);
             System.out.println(cu.consultarUbicacion(edificio, piso, area));
             u.setId(id);
             u.setEdificio(edificio);
             u.setPiso(piso);
             u.setArea(area);

             listau.add(u);
         }

         listau.sort(Comparator.comparing(Ubicacion::getId));

         for (int i = 0; i < t; i++)
             System.out.println(listau.get(i).toString());

         ant = "" + listau.get(0).getId();
         array.add(ant);
         for (int i = 1; i < t; i++) {
             actual = "" + listau.get(i).getId();
             if (Busca(array, actual)){
                 int j;
                 for (j = i-1; j < t && (actual.equals("" + listau.get(j).getId())); j++)
                     e++;
                 i = j;
             }
             else{
                 valores.add(e);
                 array.add(actual);
                 e = 1;
             }
         }
         valores.add(e);
         break;
     :
     :
     int tam1 = array.size();
     for (int i = 0; i < tam1; i++){
         Ordenar o = new Ordenar();
         o.setNombre(array.get(i));
         o.setCantidad(valores.get(i));
         ord.add(o);
     }

     // Ordena de mayor a menor los objetos
     ord.sort(Comparator.comparing(Ordenar::getCantidad));
     Collections.reverse(ord);

     // Crea los parametros necesarios para insertar en la grafica
     s = new String[tam1];
     val = new int[tam1];
     for (int i = 0; i < tam1; i++){
         s[i] = ord.get(i).getNombre();
         val[i] = ord.get(i).getCantidad();
     }

     //a.imprimirlista(lista);
     return lista;
 }
```

Figura 4. Función “BuscarUTMMI” del módulo de análisis

3. Módulo de visualización.

El objetivo de este módulo es generar diferentes tipos de gráficas (barras, circulares, entre otras) que se presentarán al usuario en pantalla en base a las estadísticas obtenidas.

Las gráficas podrán ser generadas como un archivo PDF para cuestiones de reporte.

```
// Crea una una grafica circular con n datos y valores de los parametros
public void Circular(String titulo, String s[], int val[], int tam) {
    DefaultPieDataset dataset = new DefaultPieDataset();

    for (int i = 0; i < tam; i++)
        dataset.setValue(s[i], val[i]);

    // title, data, incluir leyenda, true, true
    chart = ChartFactory.createPieChart(titulo, dataset, true, true, true);
}
// Crea una una grafica de barras con n datos y valores de los parametros
public void Barras(String titulo, String categoria, String puntuacion, String s[], int val[], int tam) {

    DefaultCategoryDataset dataset = new DefaultCategoryDataset();

    String aux = new String("");
    float[][] f = new float[1][tam];

    for (int i = 0; i < tam; i++)
        f[0][i] = (float) val[i];

    for (int i = 0; i < tam; i++)
        dataset.addValue(f[0][i], aux, s[i]);

    chart = ChartFactory.createBarChart(titulo, categoria, puntuacion, dataset,
            PlotOrientation.VERTICAL, true, true, false);
}
// Crea una una grafica lineal con los datos y valores de los parametros
public void Lineal(String titulo, String categoria, String puntuacion, String s[], int val[]) {

    String elementos = "Elementos";
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();

    int tam = s.length;
    for (int i = 0; i < tam; i++)
        dataset.addValue(val[i], elementos, s[i]);

    chart = ChartFactory.createLineChart(titulo, categoria, puntuacion, dataset,
            PlotOrientation.VERTICAL, true, true, false);
}
```

Figura 5. Funciones “Circular, Barras, Lineal” correspondientes al módulo de visualización

4. Módulo de reportes.

El módulo de reportes se encarga de generar los reportes necesarios con las estadísticas de un periodo de tiempo en específico para llevar un control más detallado de los equipos y de las ordenes de servicio y así poder usar las gráficas como indicadores que reflejen la producción del área.

En la Figura 6 se muestra la parte del código encargada de generar el archivo pdf en donde se guardar la gráfica y será almacenado por medio de un JFileChooser.

```
ActionListener action = new ActionListener() {
    public void actionPerformed(ActionEvent event){
        String filename = new String();
        JFileChooser savefile = new JFileChooser();
        savefile.setSelectedFile(new File(filename));
        savefile.showSaveDialog(savefile);
        System.out.println(filename);
        int sf = savefile.showSaveDialog(null);
        if(sf == JFileChooser.APPROVE_OPTION){
            File archivo = savefile.getSelectedFile();
            String ruta = archivo.getPath();

            PDFDocument pdfDoc = new PDFDocument();
            pdfDoc.setTitle("Grafica del area de Informatica");
            pdfDoc.setAuthor("IMSS");

            Page page = pdfDoc.createPage(new Rectangle(612, 468));
            PDFGraphics2D g2 = page.getGraphics2D();

            chart.draw(g2, new Rectangle(0, 0, 612, 468));

            System.out.println(filename);
            pdfDoc.writeToFile(new File(ruta + filename + ".pdf"));

            JOptionPane.showMessageDialog(null, "File has been saved","File Saved",JOptionPane.INFORMATION_MESSAGE);
        }else if(sf == JFileChooser.CANCEL_OPTION){
            JOptionPane.showMessageDialog(null, "File save has been canceled");
        }
    }
};
```

Figura 6. Código con el que se almacena la gráfica.

Resultados

El desarrollo del Sistema de Gestión obtuvo como resultado una aplicación de la cual pueden recopilarse cualquier dato de las ordenes de servicio y mostrar indicadores pertinentes a este, por ejemplo, mostrar en forma de gráfica la frecuencia de uso de algún recurso. En las Figuras 7 a 14, se muestran algunos ejemplos de las gráficas resultantes de algunas consultas.

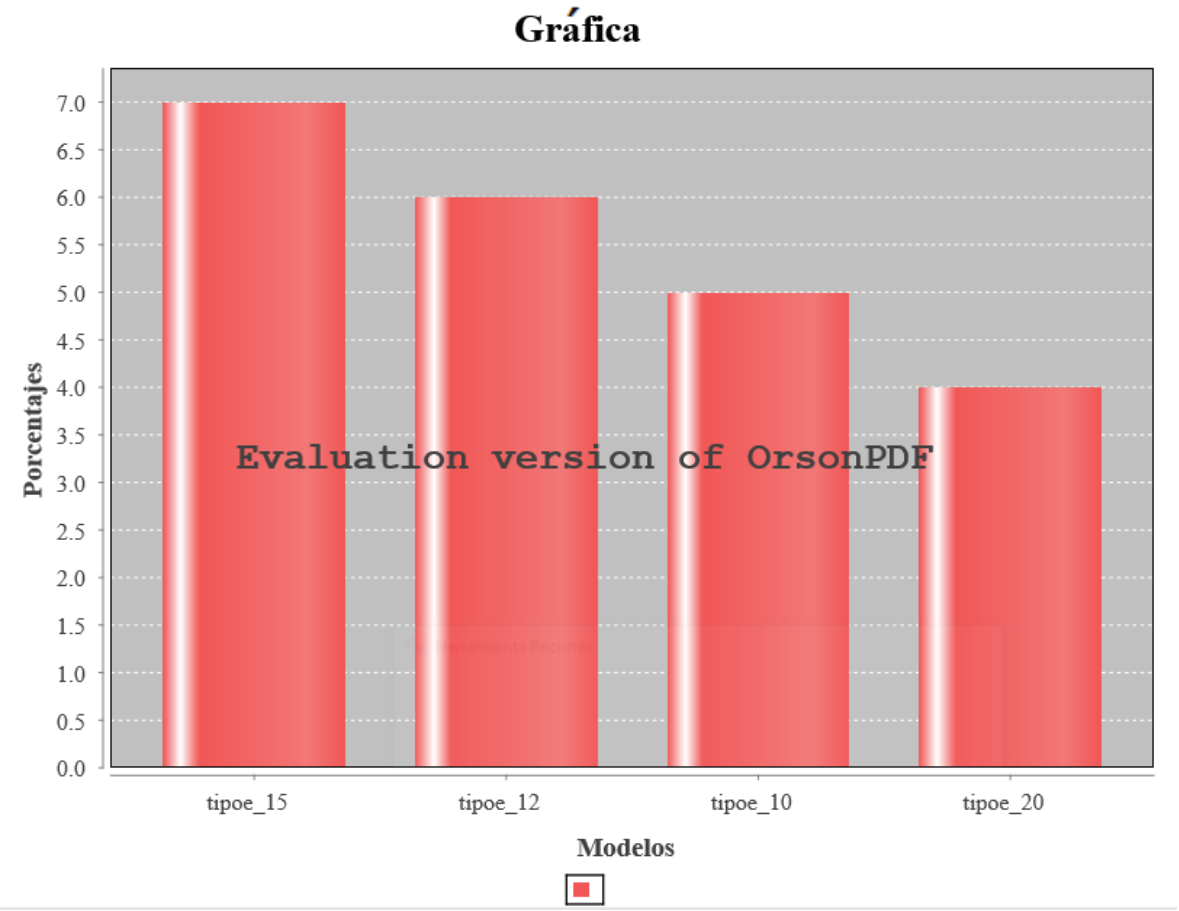

Figura 7. Gráfica de Barras reducida a 4 elementos

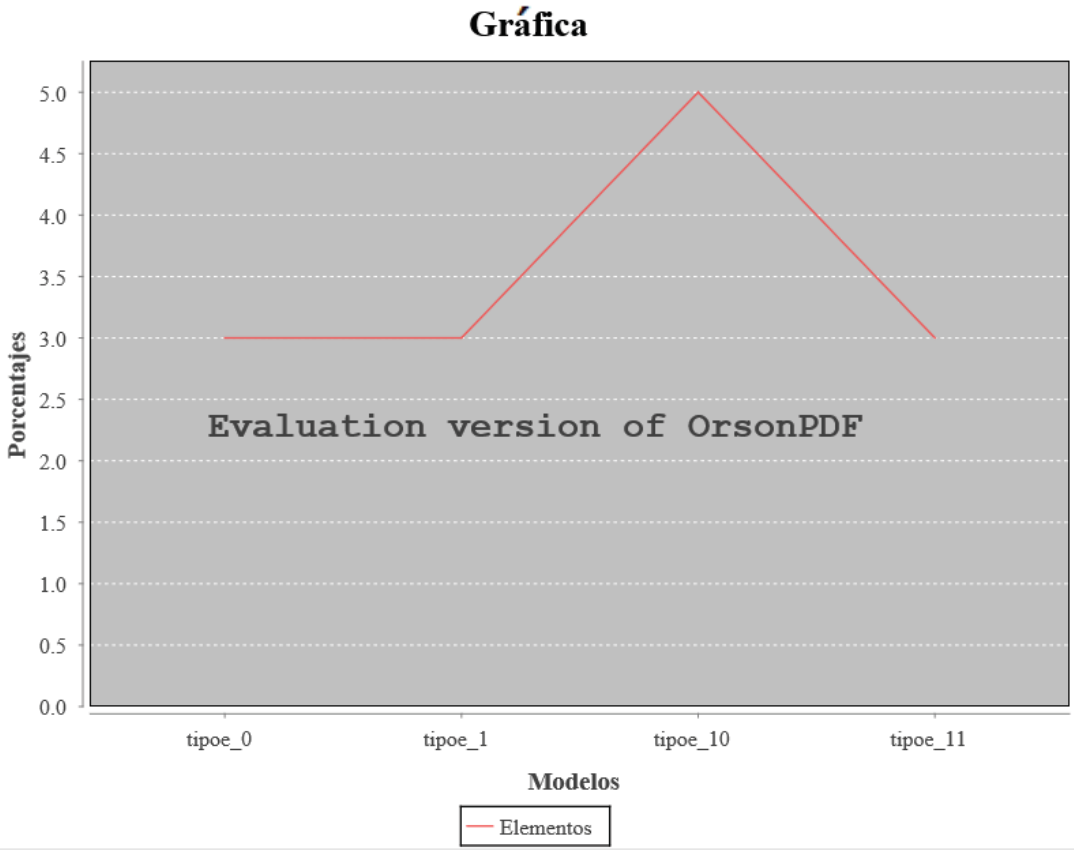

Figura 8. Gráfica Lineal reducida a 4 elementos

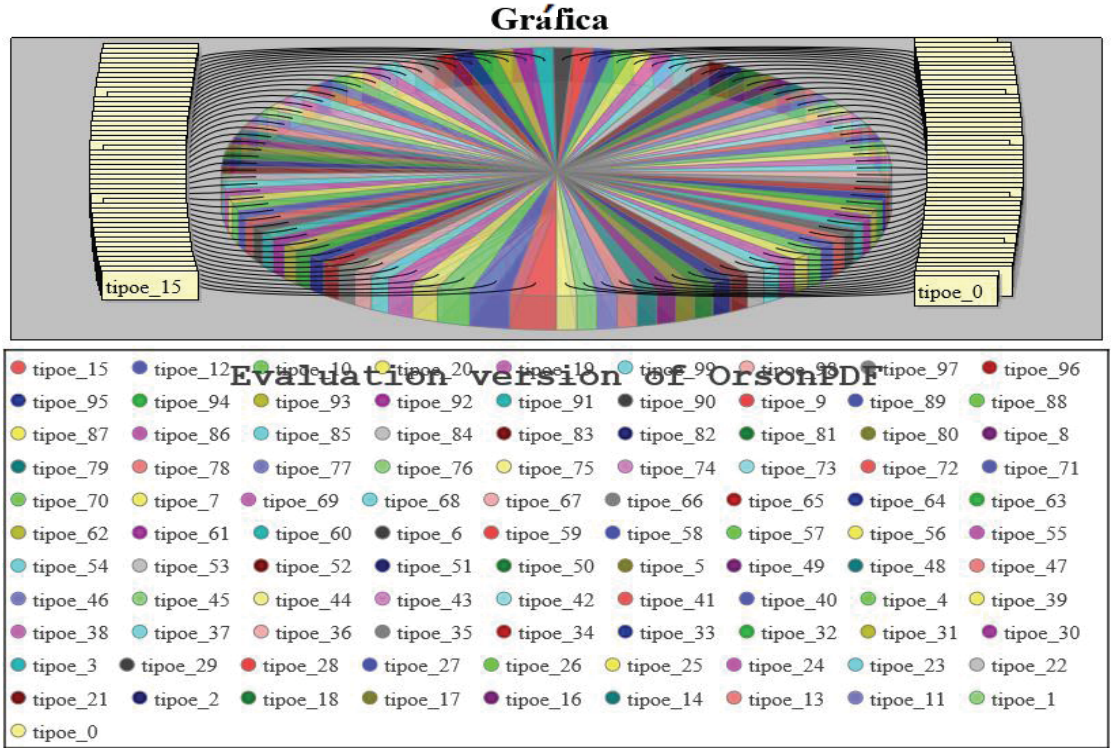

Figura 9. Gráfica Circular 3D completa

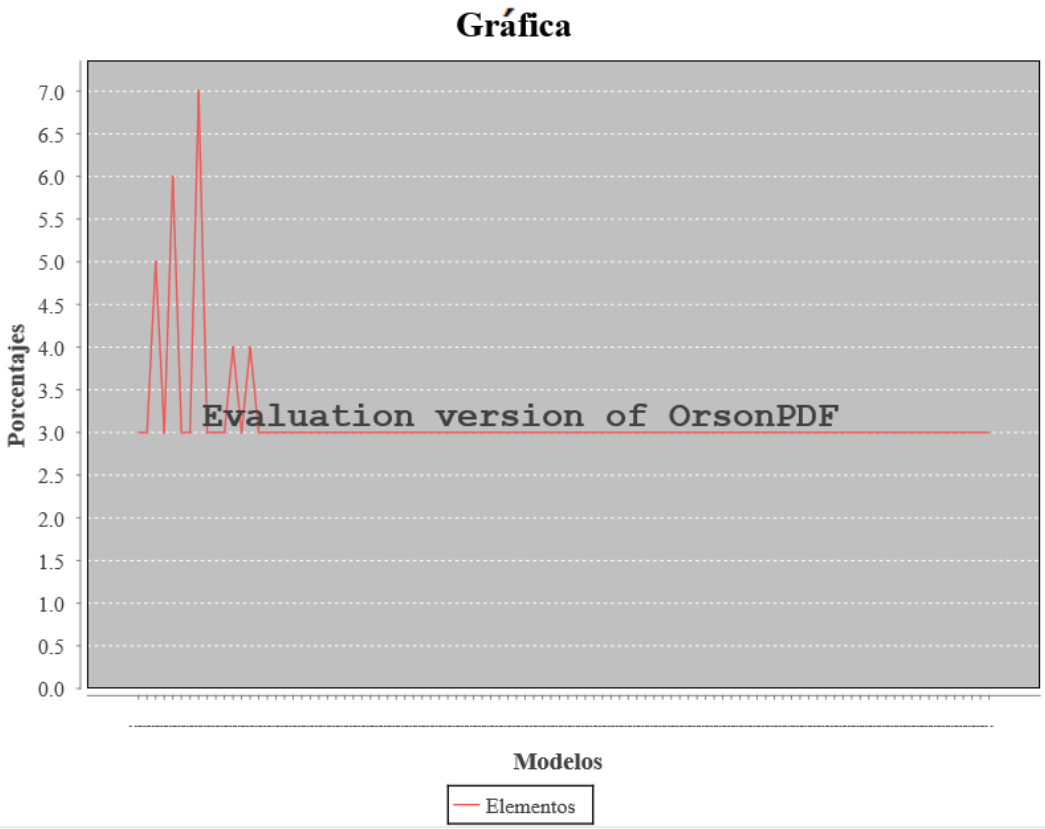

Figura 10. Gráfica Lineal completa

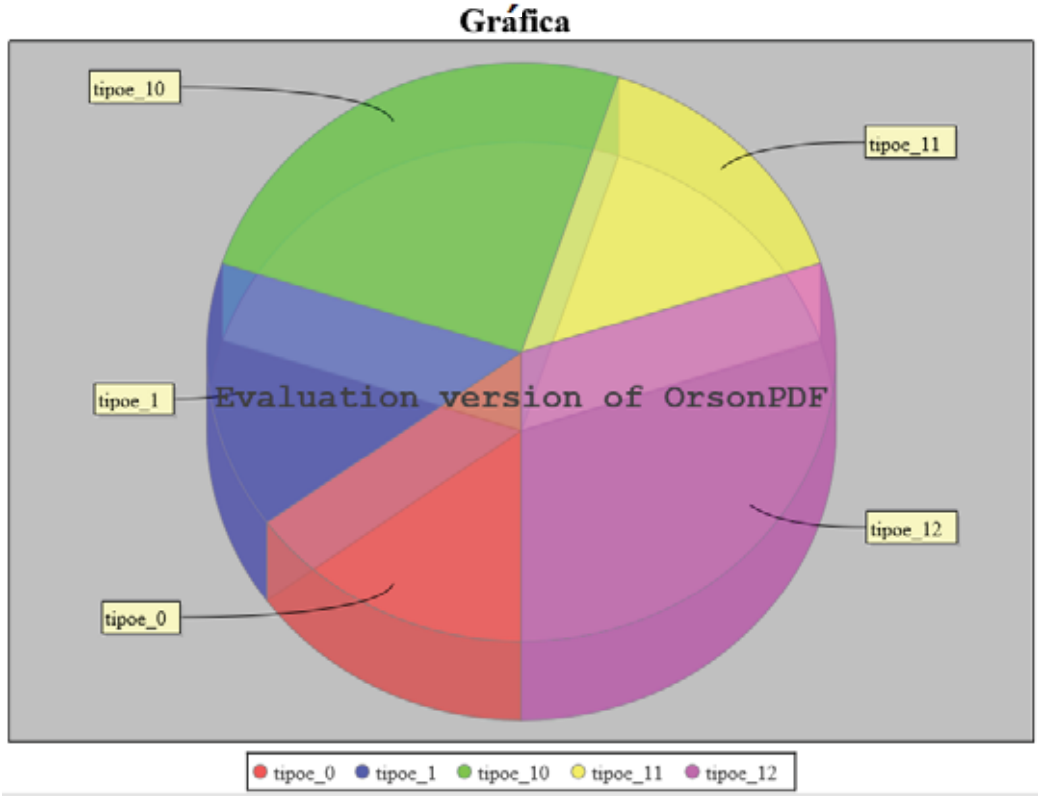

Figura 11. Gráfica Circular 3D reducida a 5 elementos

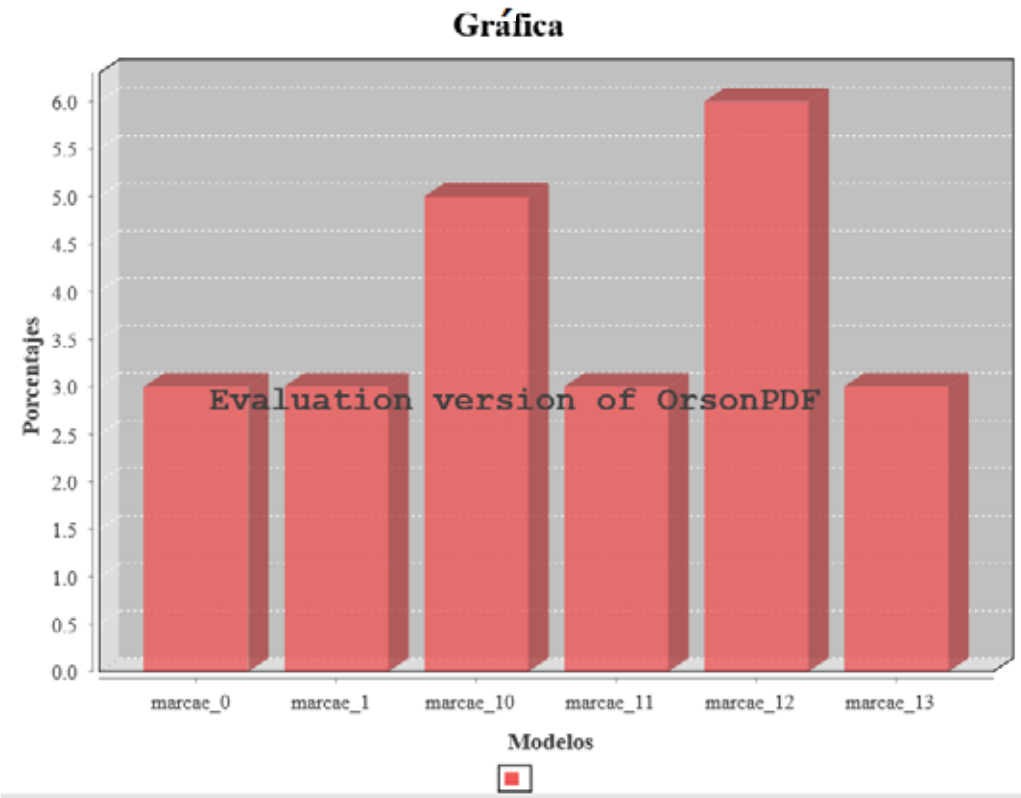

Figura 12. Gráfica de Barras 3D reducida a 6 elementos

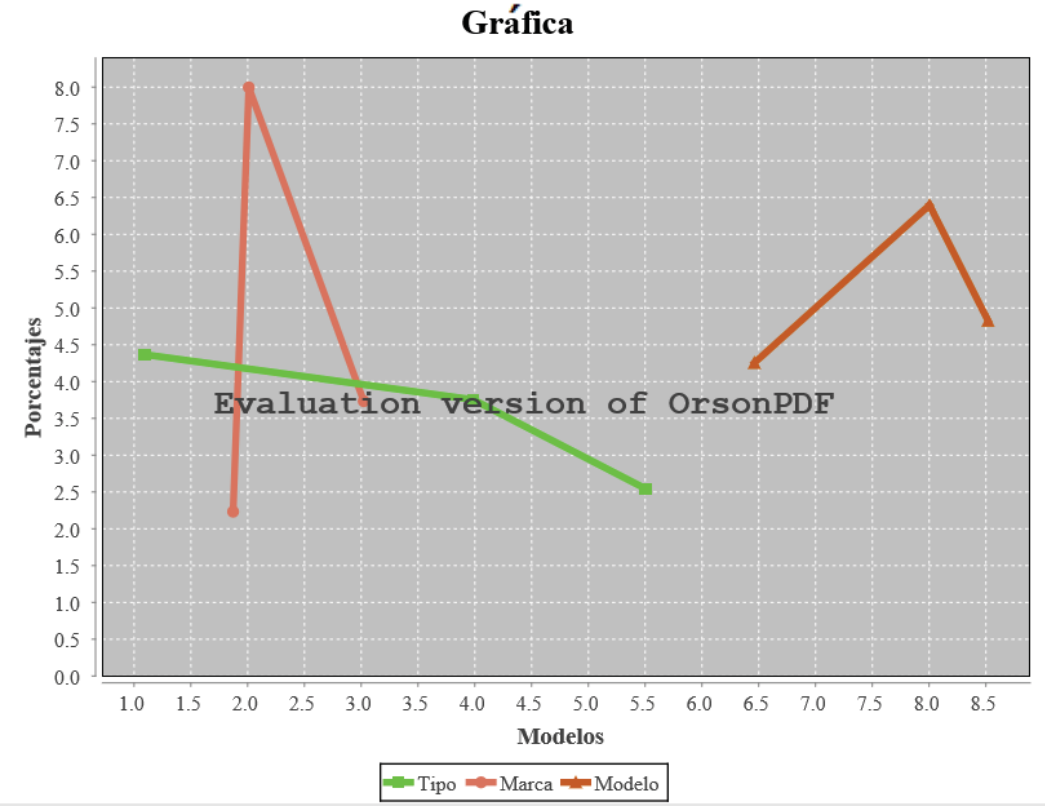

Figura 13. Gráfica XY reducida a 3 elementos

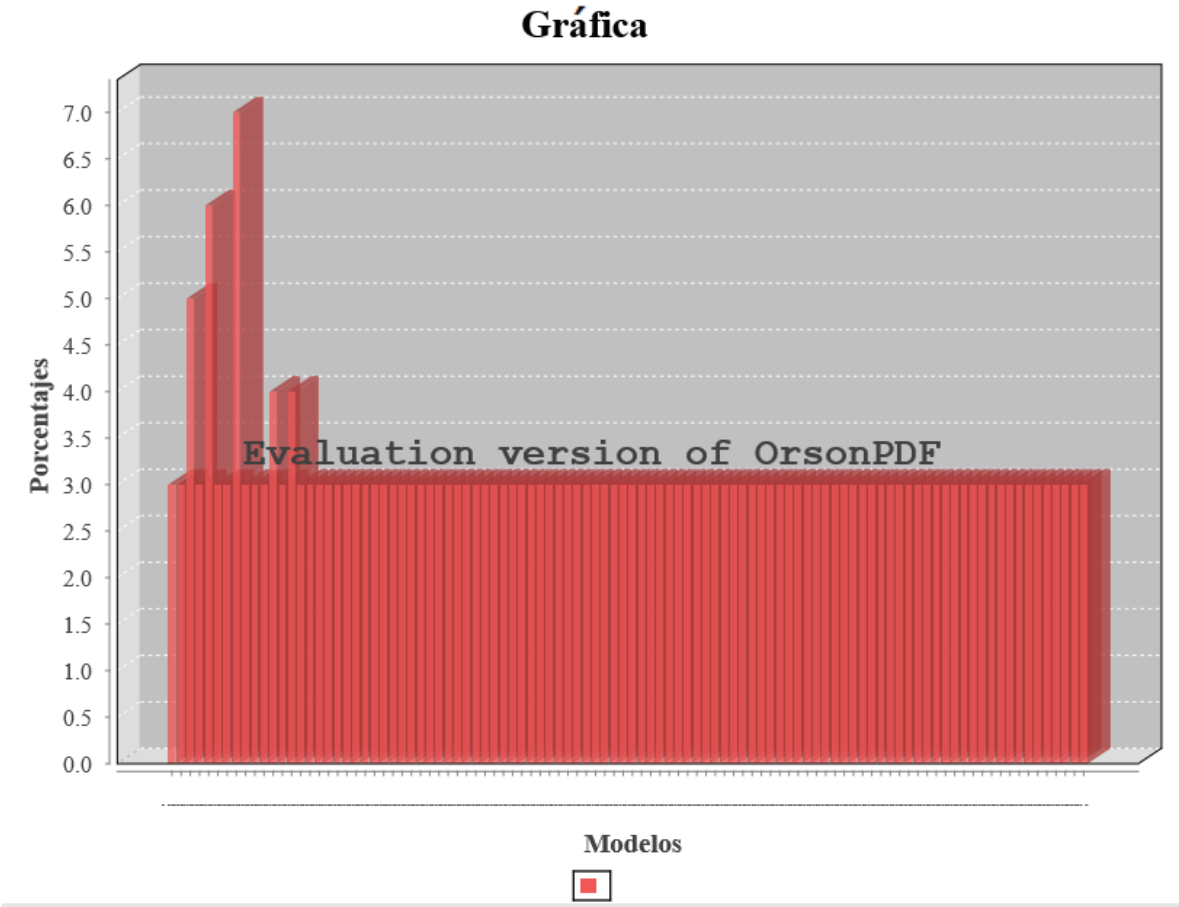

Figura 14. Gráfica de Barras 3D completa

Conclusiones

Para este proyecto, se desarrolló una aplicación de análisis para un Sistema de Información, por medio de cada uno de los módulos propuestos logrando cumplir los objetivos propuestos para el mismo. El SG es capaz de generar gráficas basadas en estadísticas que pueden ser usadas como indicadores de dentro de áreas de oportunidad del instituto.

Actualmente existen sistemas de gestión que realizan la misma tarea, sin embargo, estos son alto costo y están limitados a lo que el programa ofrece sin la posibilidad extenderlos o modificarlos a necesidad del usuario. El SG implementado está adaptado a las necesidades requeridas por instituto permitiendo obtener de este las siguientes ventajas:

- La realización de los indicadores se realiza de forma automática, pues una vez obtenido los datos necesarios, se pueden generar las gráficas correspondientes con solo dar clic en un botón, las veces que se requiera reduciendo tiempo, debido a que, de no existir el proyecto, se tendría que revisar las ordenes de manera manual.
- Se administra de una manera más fácil cada las ordenes de servicio como los recursos a disposición del área.
- El proyecto si se desea puede expandirse a las áreas con las que se tiene contacto directo para regular los recursos y agilizar los procesos entre las mismas.

Referencias Bibliográficas

[1] G. V. Martínez Torres y F. Álvarez Ramírez, “Sistema de Gestión del Repositorio de Software en Tecnologías Abiertas De la Carrera de Ingeniería en Computación de la UAM-Azcapotzalco”, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2011.

[2] D. Avila Castillo, “Análisis y gestión de recursos para brindar seguridad en una red empresarial”, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2015.

[3] S. Calva Calva y J. Medrano Herrera, “Sistema basado en web para la gestión de capital intelectual referente a la organización de las metodologías de las TIC.”, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2010.

[4] Juice Analytics, “Juice Labs - Chart Chooser”, http://labs.juiceanalytics.com/chartchooser/index.html, 2015.

[5] IBM, “IBM Cognos Analytics on Cloud - España”, https://www.ibm.com/es-es/marketplace/business-intelligence, 2017.

[6] IBM, “IBM dashDB | Analytics Business - España”, https://www.ibm.com/analytics/es/es/technology/cloud-data-services/dashdb/, 2017.

[7] The R Foundation, “R: What is R?”, https://www.r-project.org/about.html, 2017.

Apéndice A. Código Fuente

Paquete: web.negocio
Clase: Apoyo
Descripción: Clase encargada de crear un JFrame para desplegar la gráfica con la opción de guardarla en un archivo en formato pdf por medio de un botón.

```
package web.negocio;

import java.awt.BorderLayout;
import java.awt.Rectangle;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;

import com.orsonpdf.PDFDocument;
import com.orsonpdf.PDFGraphics2D;
import com.orsonpdf.Page;

public class Apoyo {

    // Imprime cualquier lista
    public void imprimirLista(List<?> lista) {
        if (lista != null) {
            int tam = lista.size();
            System.out.println("Numero de elementos en lista: " + tam);
            for (int i = 0; i < tam; i++)
                System.out.print(lista.get(i).toString() + "\n");
        }
    }

    // Muestra en un frame la gráfica dada por el objeto JFreeChart
    // El parametro clave sirve escoger que tipo de grafica desea
mostrarse
    public void mostrarGrafica(JFreeChart chart, int clave) {
        ChartPanel panel = new ChartPanel(chart);
        JFrame ventana = new JFrame();
        JButton saveButton = new JButton("Guardar");

        switch (clave) {
        case 1:
            ventana.setTitle("Grafica Circular");
```

```
            break;
        case 2:
            ventana.setTitle("Grafica Circular 3D");
            break;
        case 3:
            panel.setPreferredSize(new java.awt.Dimension(560, 367));
            ventana.setTitle("Grafica Barras");
            break;
        case 4:
            panel.setPreferredSize(new java.awt.Dimension(560, 367));
            ventana.setTitle("Grafica de Barras 3D");
            break;
        case 5:
            ventana.setTitle("Grafica Lineal");
            break;
        case 6:
            ventana.setTitle("Grafica XY");
            break;
        default:
            System.out.println("1 - Crea una grafica de tipo Circular");
            System.out.println("2 - Crea una grafica de tipo Circular
3D");
            System.out.println("3 - Crea una grafica de tipo Barras");
            System.out.println("4 - Crea una grafica de tipo Barras 3D");
            System.out.println("5 - Crea una grafica de tipo Lineal");
            System.out.println("6 - Crea una grafica de tipo XY");
        }

        // Se activa cuando se oprima el boton de guardar
        // y crea un JFileChooser para almacenar la grafica en un archivo
pdf
        ActionListener action = new ActionListener() {
            public void actionPerformed(ActionEvent event){
                String filename = new String();
                JFileChooser savefile = new JFileChooser();
                savefile.setSelectedFile(new File(filename));
                savefile.showSaveDialog(savefile);
                System.out.println(filename);
                int sf = savefile.showSaveDialog(null);
                if(sf == JFileChooser.APPROVE_OPTION){
                    File archivo = savefile.getSelectedFile();
                    String ruta = archivo.getPath();

                    PDFDocument pdfDoc = new PDFDocument();
                    pdfDoc.setTitle("Grafica del area de Informatica");
                    pdfDoc.setAuthor("IMSS");

                    Page page = pdfDoc.createPage(new Rectangle(612,
468));
                    PDFGraphics2D g2 = page.getGraphics2D();

                    chart.draw(g2, new Rectangle(0, 0, 612, 468));

                    System.out.println(filename);
                    pdfDoc.writeToFile(new File(ruta + filename +
".pdf"));

                    JOptionPane.showMessageDialog(null, "File has been
saved","File Saved",JOptionPane.INFORMATION_MESSAGE);
```

```
                }else if(sf == JFileChooser.CANCEL_OPTION){
                    JOptionPane.showMessageDialog(null, "File save has
been canceled");
                }
            }
        };

        saveButton.addActionListener(action);

        ventana.add(panel, BorderLayout.CENTER);
        ventana.add(saveButton, BorderLayout.SOUTH);
        ventana.pack();
        ventana.setVisible(true);
        ventana.setSize(800, 600);
        ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }
}
```

Paquete: web.negocio
Clase: Grafica
Descripción: Clase encargada de generar un objeto JFreeChart con el diseño de la gráfica deseada (Lineal, Circular, Barras, etc.).

```
package web.negocio;

import java.awt.BasicStroke;
import java.awt.Color;
import java.util.Random;


import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PiePlot3D;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.data.general.DefaultPieDataset;

import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

public class Grafica {

    JFreeChart chart;

    public JFreeChart getChart() {
        return chart;
    }

    public void setChart(JFreeChart chart) {
        this.chart = chart;
    }
```

```
    // Crea una una grafica circular con los datos y valores de los
parametros
    public void Circular(String titulo, String s[], int val[]) {
        DefaultPieDataset dataset = new DefaultPieDataset();

        int tam = s.length;
        for (int i = 0; i < tam; i++)
            dataset.setValue(s[i], val[i]);

        // Title, Data, Incluir leyenda
        chart = ChartFactory.createPieChart(titulo, dataset, true, true,
true);
    }

    // Crea una una grafica circular con n datos y valores de los
parametros
    public void Circular(String titulo, String s[], int val[], int tam) {
        DefaultPieDataset dataset = new DefaultPieDataset();

        for (int i = 0; i < tam; i++)
            dataset.setValue(s[i], val[i]);

        // title, data, incluir leyenda, true, true
        chart = ChartFactory.createPieChart(titulo, dataset, true, true,
true);
    }

    // Crea una una grafica circular 3D con los datos y valores de los
parametros
    public void Circular3d(String titulo, String s[], int val[]) {
        DefaultPieDataset dataset = new DefaultPieDataset();

        int tam = s.length;
        for (int i = 0; i < tam; i++)
            dataset.setValue(s[i], val[i]);

        chart = ChartFactory.createPieChart3D(titulo, dataset, true,
true, true);

        PiePlot3D plot = (PiePlot3D) chart.getPlot();
        plot.setStartAngle(270);
        plot.setForegroundAlpha(0.60f);
        plot.setInteriorGap(0.02);
    }

    // Crea una una grafica circular 3D con n datos y valores de los
parametros
    public void Circular3d(String titulo, String s[], int val[], int tam)
{
        DefaultPieDataset dataset = new DefaultPieDataset();

        for (int i = 0; i < tam; i++)
            dataset.setValue(s[i], val[i]);

        chart = ChartFactory.createPieChart3D(titulo, dataset, true,
true, true);

        PiePlot3D plot = (PiePlot3D) chart.getPlot();
        plot.setStartAngle(270);
```

```
        plot.setForegroundAlpha(0.60f);
        plot.setInteriorGap(0.02);
    }

    // Crea una una grafica de barras con los datos y valores de los
parametros
    public void Barras(String titulo, String categoria, String
puntuacion, String s1[], String s2[],
            float val[][]) {

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        int tams1 = s1.length, tams2 = s2.length;

        for (int i = 0; i < tams1; i++)
            for (int j = 0; j < tams2; j++)
                dataset.addValue(val[i][j], s1[i], s2[j]);

        chart = ChartFactory.createBarChart(titulo, categoria,
puntuacion, dataset,
                PlotOrientation.VERTICAL, true, true, false);
    }

    // Crea una una grafica de barras con los datos y valores de los
parametros
    public void Barras(String titulo, String categoria, String
puntuacion, String s[], int val[]) {

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        String aux = new String("");
        int tam = s.length;
        float[][] f = new float[1][tam];

        for (int i = 0; i < tam; i++)
            f[0][i] = (float) val[i];

        for (int i = 0; i < tam; i++)
            dataset.addValue(f[0][i], aux, s[i]);

        chart = ChartFactory.createBarChart(titulo, categoria,
puntuacion, dataset,
                PlotOrientation.VERTICAL, true, true, false);
    }

    // Crea una una grafica de barras con n datos y valores de los
parametros
    public void Barras(String titulo, String categoria, String
puntuacion, String s[], int val[], int tam) {

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        String aux = new String("");
        float[][] f = new float[1][tam];

        for (int i = 0; i < tam; i++)
            f[0][i] = (float) val[i];

        for (int i = 0; i < tam; i++)
```

```
            dataset.addValue(f[0][i], aux, s[i]);

        chart = ChartFactory.createBarChart(titulo, categoria,
puntuacion, dataset,
                PlotOrientation.VERTICAL, true, true, false);
    }

    // Crea una una grafica de barras 3D con los datos y valores de los
parametros
    public void Barras3d(String titulo, String categoria, String
puntuacion, String s1[], String s2[],
            float val[][]) {

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        int tams1 = s1.length, tams2 = s2.length;

        for (int i = 0; i < tams1; i++)
            for (int j = 0; j < tams2; j++)
                dataset.addValue(val[i][j], s1[i], s2[j]);

        chart = ChartFactory.createBarChart3D(titulo, categoria,
puntuacion, dataset,
                PlotOrientation.VERTICAL, true, true, false);
    }

    // Crea una una grafica de barras 3D con los datos y valores de los
parametros
    public void Barras3d(String titulo, String categoria, String
puntuacion, String s[], int val[]) {

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        String aux = new String("");
        int tam = s.length;
        float[][] f = new float[1][tam];

        for (int i = 0; i < tam; i++)
            f[0][i] = (float) val[i];

        for (int i = 0; i < tam; i++)
            dataset.addValue(f[0][i], aux, s[i]);

        chart = ChartFactory.createBarChart3D(titulo, categoria,
puntuacion, dataset,
                PlotOrientation.VERTICAL, true, true, false);
    }

    // Crea una una grafica de barras 3D con n datos y valores de los
parametros
    public void Barras3d(String titulo, String categoria, String
puntuacion, String s[], int val[], int tam) {

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        String aux = new String("");
        float[][] f = new float[1][tam];

        for (int i = 0; i < tam; i++)
```

```
            f[0][i] = (float) val[i];

        for (int i = 0; i < tam; i++)
            dataset.addValue(f[0][i], aux, s[i]);

        chart = ChartFactory.createBarChart3D(titulo, categoria,
puntuacion, dataset,
                PlotOrientation.VERTICAL, true, true, false);
    }

    // Crea una una grafica lineal con los datos y valores de los
parametros
    public void Lineal(String titulo, String categoria, String
puntuacion, String elementos, String s[],
            int val[]) {

        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        int tam = s.length;
        for (int i = 0; i < tam; i++)
            dataset.addValue(val[i], elementos, s[i]);

        chart = ChartFactory.createLineChart(titulo, categoria,
puntuacion, dataset,
                PlotOrientation.VERTICAL, true, true, false);
    }

    // Crea una una grafica lineal con los datos y valores de los
parametros
    public void Lineal(String titulo, String categoria, String
puntuacion, String s[], int val[]) {

        String elementos = "Elementos";
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        int tam = s.length;
        for (int i = 0; i < tam; i++)
            dataset.addValue(val[i], elementos, s[i]);

        chart = ChartFactory.createLineChart(titulo, categoria,
puntuacion, dataset,
                PlotOrientation.VERTICAL, true, true, false);
    }

    // Crea una una grafica lineal con n datos y valores de los
parametros
    public void Lineal(String titulo, String categoria, String
puntuacion, String s[], int val[], int tam) {

        String elementos = "Elementos";
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        for (int i = 0; i < tam; i++)
            dataset.addValue(val[i], elementos, s[i]);

        chart = ChartFactory.createLineChart(titulo, categoria,
puntuacion, dataset,
                PlotOrientation.VERTICAL, true, true, false);
    }
```

```
    // Crea una una grafica XY con los datos y valores de los parametros
    public void XY(String titulo, String categoria, String puntuacion,
String s[], float val[][][],
            int cantidad) {

        XYSeriesCollection dataset = new XYSeriesCollection();

        int tams = s.length;

        for (int i = 0; i < tams; i++) {
            XYSeries xys = new XYSeries(s[i]);
            for (int j = 0; j < cantidad; j++)
                xys.add(val[i][j][0], val[i][j][1]);
            dataset.addSeries(xys);
        }

        chart = ChartFactory.createXYLineChart(titulo, categoria,
puntuacion, dataset,
                PlotOrientation.VERTICAL, true, true, false);

        XYPlot plot = chart.getXYPlot();

        Random rand = new Random();
        int r,g,b;
        XYLineAndShapeRenderer renderer = new XYLineAndShapeRenderer();

        for (int i = 0; i < tams; i++) {
            r = rand.nextInt(256);
            g = rand.nextInt(256);
            b = rand.nextInt(256);
            Color c = new Color(r, g, b);
            renderer.setSeriesPaint(i, c);
            renderer.setSeriesStroke(i, new BasicStroke(4.0f));
        }

        plot.setRenderer(renderer);
    }

    // Funcion de prueba para la Grafica XY
    public void XY(String titulo, String categoria, String puntuacion) {

        Random rand = new Random();
        int tams = 3;

        XYSeries carro = new XYSeries("Tipo");
        carro.add(rand.nextFloat()*10, rand.nextFloat()*10);
        carro.add(rand.nextFloat()*10, rand.nextFloat()*10);
        carro.add(rand.nextFloat()*10, rand.nextFloat()*10);

        XYSeries bici = new XYSeries("Marca");
        bici.add(rand.nextFloat()*10, rand.nextFloat()*10);
        bici.add(rand.nextFloat()*10, rand.nextFloat()*10);
        bici.add(rand.nextFloat()*10, rand.nextFloat()*10);

        XYSeries moto = new XYSeries("Modelo");
        moto.add(rand.nextFloat()*10, rand.nextFloat()*10);
        moto.add(rand.nextFloat()*10, rand.nextFloat()*10);
        moto.add(rand.nextFloat()*10, rand.nextFloat()*10);
```

```
        XYSeriesCollection dataset = new XYSeriesCollection();
        dataset.addSeries(carro);
        dataset.addSeries(bici);
        dataset.addSeries(moto);

        chart = ChartFactory.createXYLineChart(
                        titulo, categoria, puntuacion, dataset,
                        PlotOrientation.VERTICAL, true, true, false);

        XYPlot plot = chart.getXYPlot();

        int r,g,b;
        XYLineAndShapeRenderer renderer = new XYLineAndShapeRenderer();

        for (int i = 0; i < tams; i++) {
            r = rand.nextInt(256);
            g = rand.nextInt(256);
            b = rand.nextInt(256);
            Color c = new Color(r, g, b);
            renderer.setSeriesPaint(i, c);
            renderer.setSeriesStroke(i, new BasicStroke(4.0f));
        }
        plot.setRenderer(renderer);
    }
}
```

Paquete: web.negocio
Clase: Busqueda
Descripción: Clase encargada de buscar, contar y ordenar de acuerdo con algún elemento (ubicación, tipo, marca, modelo, etc.) de las ordenes de servicio para la creación de las gráficas correspondientes.

```
package web.negocio;

import java.text.ParseException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedList;
import java.util.List;

import org.jfree.chart.JFreeChart;

import hibernate.beans.Ubicacion;
import hibernate.compuestas.OrdenUbicacion;
import hibernate.operaciones.consultas.ConsultaOrden;
import hibernate.operaciones.consultas.ConsultaUbicacion;

public class Busqueda {

    String[] s = null;
    int[] val = null;

    // Busca una cadena especifica en el ArrayList
```

```
    private boolean Busca(ArrayList<String> s, String pal) {

        int t = s.size();
        for (int i = 0; i < t; i++) {
            if (s.get(i).equals(pal))
                return true;
        }
        return false;
    }

    // Consulta las ordenes en el rango de fechas y las ordena
    // de acuerdo a la clave dada de mayor numero de ocasiones
    // en las que aparecen los objetos al menor
    public List<OrdenUbicacion> BuscarUTMMSI(String fecha1, String
fecha2, int clave) throws ParseException {
        ConsultaOrden c = new ConsultaOrden();
        List<OrdenUbicacion> lista = c.consultarOrdenFecha(fecha1,
fecha2);
        ArrayList<String> array = new ArrayList<>();
        ArrayList<Integer> valores = new ArrayList<>();
        List<Ordenar> ord = new LinkedList<>();

        String ant, actual;
        Integer e = 1;
        int t = lista.size();

        // Ordena las ordenes de acuerdo a la clave y obtiene
        // cuantas aparece cada uno de los objetos
        switch (clave) {
        case 1:
            ConsultaUbicacion cu = new ConsultaUbicacion();
            List<Ubicacion> listau = new LinkedList<Ubicacion>();

            for (int i = 0; i < t; i++){
                Ubicacion u = new Ubicacion();

                String edificio = lista.get(i).getEdificio();
                int piso = lista.get(i).getPiso();
                String area = lista.get(i).getArea();
                int id = cu.consultarUbicacion(edificio, piso, area);
                System.out.println(edificio + " " + piso + " " + area);
                System.out.println(cu.consultarUbicacion(edificio, piso,
area));
                u.setId(id);
                u.setEdificio(edificio);
                u.setPiso(piso);
                u.setArea(area);

                listau.add(u);
            }

            listau.sort(Comparator.comparing(Ubicacion::getId));

            for (int i = 0; i < t; i++)
                System.out.println(listau.get(i).toString());

            ant = "" + listau.get(0).getId();
            array.add(ant);
            for (int i = 1; i < t; i++) {
```

```
                actual = "" + listau.get(i).getId();
                if (Busca(array, actual)){
                    int j;
                    for (j = i-1; j < t && (actual.equals("" +
listau.get(j).getId())); j++)
                        e++;
                    i = j;
                }
                else{
                    valores.add(e);
                    array.add(actual);
                    e = 1;
                }
            }
            valores.add(e);
            // Comparator<OrdenUbicacion> byTipo =
            // Comparator.comparing(OrdenUbicacion::getTipoEquipo);
            // Collections.sort(lista, byTipo);
            break;
        case 2:

lista.sort(Comparator.comparing(OrdenUbicacion::getTipoEquipo));
            ant = lista.get(0).getTipoEquipo();
            array.add(ant);
            for (int i = 1; i < t; i++) {
                actual = lista.get(i).getTipoEquipo();
                if (Busca(array, actual)){
                    int j;
                    for (j = i-1; j < t &&
(actual.equals(lista.get(j).getTipoEquipo())); j++)
                        e++;
                    i = j;
                }
                else{
                    valores.add(e);
                    array.add(actual);
                    e = 1;
                }
            }
            valores.add(e);
            // Comparator<OrdenUbicacion> byTipo =
            // Comparator.comparing(OrdenUbicacion::getTipoEquipo);
            // Collections.sort(lista, byTipo);
            break;
        case 3:
            lista.sort(Comparator.comparing(OrdenUbicacion::getMarca));
            ant = lista.get(0).getMarca();
            array.add(ant);
            for (int i = 1; i < t; i++) {
                actual = lista.get(i).getMarca();
                if (Busca(array, actual)){
                    int j;
                    for (j = i-1; j < t &&
(actual.equals(lista.get(j).getMarca())); j++)
                        e++;
                    i = j;
                }
                else{
                    valores.add(e);
```

```
                    array.add(actual);
                    e = 1;
                }
            }
            valores.add(e);
            // Comparator<OrdenUbicacion> byMarca =
            // Comparator.comparing(OrdenUbicacion::getMarca);
            // Collections.sort(lista, byMarca);
            break;
        case 4:
            lista.sort(Comparator.comparing(OrdenUbicacion::getModelo));
            ant = lista.get(0).getModelo();
            array.add(ant);
            for (int i = 1; i < t; i++) {
                actual = lista.get(i).getModelo();
                if (Busca(array, actual)){
                    int j;
                    for (j = i-1; j < t &&
(actual.equals(lista.get(j).getModelo())); j++)
                        e++;
                    i = j;
                }
                else{
                    valores.add(e);
                    array.add(actual);
                    e = 1;
                }
            }
            valores.add(e);
            // Comparator<OrdenUbicacion> byModelo =
            // Comparator.comparing(OrdenUbicacion::getModelo);
            // Collections.sort(lista, byModelo);
            break;
        case 5:

lista.sort(Comparator.comparing(OrdenUbicacion::getNsEquipo));
            ant = lista.get(0).getNsEquipo();
            array.add(ant);
            for (int i = 1; i < t; i++) {
                actual = lista.get(i).getNsEquipo();
                if (Busca(array, actual)){
                    int j;
                    for (j = i-1; j < t &&
(actual.equals(lista.get(j).getNsEquipo())); j++)
                        e++;
                    i = j;
                }
                else{
                    valores.add(e);
                    array.add(actual);
                    e = 1;
                }
            }
            valores.add(e);
            // Comparator<OrdenUbicacion> bySerie =
            // Comparator.comparing(OrdenUbicacion::getNsEquipo);
            // Collections.sort(lista, bySerie);
            break;
        case 6:
```

```
lista.sort(Comparator.comparing(OrdenUbicacion::getNiEquipo));
            ant = lista.get(0).getNiEquipo();
            array.add(ant);
            for (int i = 1; i < t; i++) {
                actual = lista.get(i).getNiEquipo();
                if (Busca(array, actual)){
                    int j;
                    for (j = i-1; j < t &&
(actual.equals(lista.get(j).getNiEquipo())); j++)
                        e++;
                    i = j;
                }
                else{
                    valores.add(e);
                    array.add(actual);
                    e = 1;
                }
            }
            valores.add(e);
            // Comparator<OrdenUbicacion> byInventario =
            // Comparator.comparing(OrdenUbicacion::getNiEquipo);
            // Collections.sort(lista, byInventario);
            break;
        default:
            System.out.println("1 - El dato ingresado en el campo de
busqueda lo busca en ubicacion");
            System.out.println("2 - El dato ingresado en el campo de
busqueda lo busca en tipo");
            System.out.println("3 - El dato ingresado en el campo de
busqueda lo busca en marca");
            System.out.println("4 - El dato ingresado en el campo de
busqueda lo busca en modelo");
            System.out.println("5 - El dato ingresado en el campo de
busqueda lo busca en numero de serie");
            System.out.println("6 - El dato ingresado en el campo de
busqueda lo busca en numero de inventario");
        }

        int tam1 = array.size();
        for (int i = 0; i < tam1; i++){
            Ordenar o = new Ordenar();
            o.setNombre(array.get(i));
            o.setCantidad(valores.get(i));
            ord.add(o);
        }

        // Ordena de mayor a menor los objetos
        ord.sort(Comparator.comparing(Ordenar::getCantidad));
        Collections.reverse(ord);

        // Crea los parametros necesarios para insertar en la grafica
        s = new String[tam1];
        val = new int[tam1];
        for (int i = 0; i < tam1; i++){
            s[i] = ord.get(i).getNombre();
            val[i] = ord.get(i).getCantidad();
        }
```

```
        //a.imprimirlista(lista);
        return lista;
    }

    // Crea una grafica segun el tipo requerido. Asignandole el titulo,
    // la categoria y la puntuacion permitiendo mostrar la grafica
    // con todos los datos o con una cantidad de datos en especifico
    public JFreeChart Grafica(String titulo, String categoria, String
puntuacion, int tipo, boolean completa, int cantidad) {

        Grafica g = new Grafica();
        JFreeChart chart;

        switch (tipo) {
        case 1:
            if (completa)
                g.Circular(titulo, s, val);
            else
                g.Circular(titulo, s, val, cantidad);
            break;
        case 2:
            if (completa)
                g.Circular3d(titulo, s, val);
            else
                g.Circular3d(titulo, s, val, cantidad);
            break;
        case 3:
            if (completa)
                g.Barras(titulo, categoria, puntuacion, s, val);
            else
                g.Barras(titulo, categoria, puntuacion, s, val,
cantidad);
            break;
        case 4:
            if (completa)
                g.Barras3d(titulo, categoria, puntuacion, s, val);
            else
                g.Barras3d(titulo, categoria, puntuacion, s, val,
cantidad);
            break;
        case 5:
            if (completa)
                g.Lineal(titulo, categoria, puntuacion, s, val);
            else
                g.Lineal(titulo, categoria, puntuacion, s, val,
cantidad);
            break;
        case 6:
            g.XY(titulo, categoria, puntuacion);
            break;
        default:
            System.out.println ("1 - Crea una grafica de tipo Circular");
            System.out.println ("2 - Crea una grafica de tipo Circular
3D");
            System.out.println ("3 - Crea una grafica de tipo Barras");
            System.out.println ("4 - Crea una grafica de tipo Barras
3D");
            System.out.println ("5 - Crea una grafica de tipo Lineal");
            System.out.println ("6 - Crea una grafica de tipo XY");
```

```
        }
        chart = g.getChart();
        return chart;
    }
}
```

Paquete: hibernate.operaciones.consultas
Clase: ConsultaOrden
Descripción: Clase encargada de consultar las ordenes de la base de datos en un rango de fechas dado permitiendo consultarlas por alguno de sus elementos (ubicación, tipo, marca, modelo, etc.).

```
package hibernate.operaciones.consultas;

import hibernate.compuestas.OrdenUbicacion;
import hibernate.conexion.CrearConexion;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.LinkedList;
import java.util.List;

import org.hibernate.Session;
import org.hibernate.query.Query;

public class ConsultaOrden {

    public List<OrdenUbicacion> consultarOrdenFecha(String fecha1, String
fecha2)
            throws ParseException {
        Session sesion = null;
        List<Object[]> list;
        List<OrdenUbicacion> listaPU = new LinkedList();
        try {
            sesion = CrearConexion.getSessionFactory().openSession();
        } catch (ExceptionInInitializerError ex) {
            System.out.println("No se pudo crear sesion");
            throw new ExceptionInInitializerError(ex);
        }

        String sentencia = "SELECT UBI.edificio, UBI.piso, UBI.area,
ORD.descripcionUbicacion, "
                + "ORD.clave, ORD.folio, ORD.fecha, ORD.inicio, ORD.fin,
ORD.solicitante, "
                + "ORD.tipoEquipo, ORD.marca, ORD.modelo, ORD.nsEquipo,
ORD.niEquipo, "
                + "ORD.falla, ORD.solucion, ORD.observaciones,
ORD.realizo "
                + "FROM OrdenServicio AS ORD, Ubicacion AS UBI "
```

```
                + "WHERE ORD.idUbicacion = UBI.id "
                + "AND (ORD.fecha BETWEEN :ordfecha1 AND :ordfecha2) "
                + "ORDER BY ORD.fecha DESC";
        Query query = sesion.createQuery(sentencia);
        query.setParameter("ordfecha1", fecha1);
        query.setParameter("ordfecha2", fecha2);
        list = query.list();

        for (int i = 0; i < list.size(); i++) {
            Object[] datosRecuperados = list.get(i);

            OrdenUbicacion orUbi = new OrdenUbicacion();
            orUbi.setEdificio((String) datosRecuperados[0]);
            //orUbi.setPiso((Integer)datosRecuperados[1]);
            String x = (String)datosRecuperados[1];
            orUbi.setPiso(Integer.parseInt(x));
            orUbi.setArea((String) datosRecuperados[2]);
            orUbi.setDescripcionUbicacion((String) datosRecuperados[3]);
            orUbi.setClave((String)datosRecuperados[4]);
            orUbi.setFolio((Integer)datosRecuperados[5]);
            orUbi.setFecha((String) datosRecuperados[6]);
            orUbi.setInicio((String) datosRecuperados[7]);
            orUbi.setFin((String) datosRecuperados[8]);
            orUbi.setSolicitante((String) datosRecuperados[9]);
            orUbi.setTipoEquipo((String) datosRecuperados[10]);
            orUbi.setMarca((String) datosRecuperados[11]);
            orUbi.setModelo((String) datosRecuperados[12]);
            orUbi.setNsEquipo((String) datosRecuperados[13]);
            orUbi.setNiEquipo((String) datosRecuperados[14]);
            orUbi.setFalla((String) datosRecuperados[15]);
            orUbi.setSolucion((String) datosRecuperados[16]);
            orUbi.setObservaciones((String) datosRecuperados[17]);
            orUbi.setRealizo((String) datosRecuperados[18]);

            listaPU.add(orUbi);

        }
        sesion.close();
        return listaPU;
    }

    public List<OrdenUbicacion> consultarOrdenTipo(String tipo, String
fecha1, String fecha2) throws ParseException {
        Session sesion = null;
        List<Object[]> list;
        List<OrdenUbicacion> listaOrden = new LinkedList();
        try {
            sesion = CrearConexion.getSessionFactory().openSession();
        } catch (ExceptionInInitializerError ex) {
            System.out.println("No se pudo crear sesion");
            throw new ExceptionInInitializerError(ex);
        }

        String sentencia = "SELECT UBI.edificio, UBI.piso, UBI.area,
ORD.descripcionUbicacion, ORD.clave, ORD.folio, ORD.fecha, ORD.inicio,
ORD.fin,"
                + " ORD.solicitante, ORD.tipoEquipo, ORD.marca,
ORD.modelo, ORD.nsEquipo, ORD.niEquipo, ORD.falla, ORD.solucion,"
```

```
                + " ORD.observaciones, ORD.realizo FROM OrdenServicio AS
ORD, Ubicacion AS UBI "
                + " WHERE ORD.idUbicacion = UBI.id"
                + " AND ORD.tipoEquipo = :ordtipo "
                + " AND (ORD.fecha BETWEEN :ordfecha1 AND :ordfecha2) "
                + " ORDER BY ORD.fecha DESC";
        Query query = sesion.createQuery(sentencia);
        query.setParameter("ordtipo", tipo);
        query.setParameter("ordfecha1", fecha1);
        query.setParameter("ordfecha2", fecha2);
        list = query.list();

        for (int i = 0; i < list.size(); i++) {
            Object[] datosRecuperados = list.get(i);

            OrdenUbicacion orUbi = new OrdenUbicacion();
            orUbi.setEdificio((String) datosRecuperados[0]);
            String x = (String)datosRecuperados[1];
            orUbi.setPiso(Integer.parseInt(x));
            orUbi.setArea((String) datosRecuperados[2]);
            orUbi.setDescripcionUbicacion((String) datosRecuperados[3]);
            orUbi.setClave((String)datosRecuperados[4]);
            orUbi.setFolio((Integer)datosRecuperados[5]);
            orUbi.setFecha((String)datosRecuperados[6]);
            orUbi.setInicio((String) datosRecuperados[7]);
            orUbi.setFin((String) datosRecuperados[8]);
            orUbi.setSolicitante((String) datosRecuperados[9]);
            orUbi.setTipoEquipo((String) datosRecuperados[10]);
            orUbi.setMarca((String) datosRecuperados[11]);
            orUbi.setModelo((String) datosRecuperados[12]);
            orUbi.setNsEquipo((String) datosRecuperados[13]);
            orUbi.setNiEquipo((String) datosRecuperados[14]);
            orUbi.setFalla((String) datosRecuperados[15]);
            orUbi.setSolucion((String) datosRecuperados[16]);
            orUbi.setObservaciones((String) datosRecuperados[17]);
            orUbi.setRealizo((String) datosRecuperados[18]);

            listaOrden.add(orUbi);

        }
        sesion.close();
        return listaOrden;
    }

    public List<OrdenUbicacion> consultarOrdenMarca( String marca, String
fecha1, String fecha2)
        throws ParseException {
    Session sesion = null;
    List<Object[]> list;
    List<OrdenUbicacion> listaOrden = new LinkedList();
    try {
        sesion = CrearConexion.getSessionFactory().openSession();
    } catch (ExceptionInInitializerError ex) {
        System.out.println("No se pudo crear sesion");
        throw new ExceptionInInitializerError(ex);
    }
```

```
    String sentencia = "SELECT UBI.edificio, UBI.piso, UBI.area,
ORD.descripcionUbicacion, ORD.clave, ORD.folio, ORD.fecha, ORD.inicio,
ORD.fin,"
            + " ORD.solicitante, ORD.tipoEquipo, ORD.marca, ORD.modelo,
ORD.nsEquipo, ORD.niEquipo, ORD.falla, ORD.solucion,"
            + " ORD.observaciones, ORD.realizo FROM OrdenServicio AS ORD,
Ubicacion AS UBI "
            + " WHERE ORD.idUbicacion = UBI.id"
            + " AND ORD.marca = :ordmarca"
            + " AND (ORD.fecha BETWEEN :ordfecha1 AND :ordfecha2) "
            + " ORDER BY ORD.fecha DESC";
    Query query = sesion.createQuery(sentencia);
    query.setParameter("ordmarca", marca);
    query.setParameter("ordfecha1", fecha1);
    query.setParameter("ordfecha2", fecha2);
    list = query.list();

    for (int i = 0; i < list.size(); i++) {
        Object[] datosRecuperados = list.get(i);

        OrdenUbicacion orUbi = new OrdenUbicacion();
        orUbi.setEdificio((String) datosRecuperados[0]);
        String x = (String)datosRecuperados[1];
        orUbi.setPiso(Integer.parseInt(x));
        orUbi.setArea((String) datosRecuperados[2]);
        orUbi.setDescripcionUbicacion((String) datosRecuperados[3]);
        orUbi.setClave((String)datosRecuperados[4]);
        orUbi.setFolio((Integer)datosRecuperados[5]);
        orUbi.setFecha((String)datosRecuperados[6]);
        orUbi.setInicio((String) datosRecuperados[7]);
        orUbi.setFin((String) datosRecuperados[8]);
        orUbi.setSolicitante((String) datosRecuperados[9]);
        orUbi.setTipoEquipo((String) datosRecuperados[10]);
        orUbi.setMarca((String) datosRecuperados[11]);
        orUbi.setModelo((String) datosRecuperados[12]);
        orUbi.setNsEquipo((String) datosRecuperados[13]);
        orUbi.setNiEquipo((String) datosRecuperados[14]);
        orUbi.setFalla((String) datosRecuperados[15]);
        orUbi.setSolucion((String) datosRecuperados[16]);
        orUbi.setObservaciones((String) datosRecuperados[17]);
        orUbi.setRealizo((String) datosRecuperados[18]);

        listaOrden.add(orUbi);

    }
    sesion.close();
    return listaOrden;
}

    public List<OrdenUbicacion> consultarOrdenModelo( String modelo,
String fecha1, String fecha2)
            throws ParseException {
        Session sesion = null;
        List<Object[]> list;
        List<OrdenUbicacion> listaOrden = new LinkedList();
        try {
            sesion = CrearConexion.getSessionFactory().openSession();
        } catch (ExceptionInInitializerError ex) {
            System.out.println("No se pudo crear sesion");
```

```
            throw new ExceptionInInitializerError(ex);
        }

        String sentencia = "SELECT UBI.edificio, UBI.piso, UBI.area,
ORD.descripcionUbicacion, ORD.clave, ORD.folio, ORD.fecha, ORD.inicio,
ORD.fin,"
                           + " ORD.solicitante, ORD.tipoEquipo,
ORD.marca, ORD.modelo, ORD.nsEquipo, ORD.niEquipo, ORD.falla,
ORD.solucion,"
                           + " ORD.observaciones, ORD.realizo FROM
OrdenServicio AS ORD, Ubicacion AS UBI "
                           + " WHERE ORD.idUbicacion = UBI.id"
                           + " AND ORD.modelo=:ordmodelo"
                           + " AND (ORD.fecha BETWEEN :ordfecha1 AND
:ordfecha2) "
                           + " ORDER BY ORD.fecha DESC";
        Query query = sesion.createQuery(sentencia);
        query.setParameter("ordmodelo", modelo);
        query.setParameter("ordfecha1", fecha1);
        query.setParameter("ordfecha2", fecha2);
        list = query.list();

        for (int i = 0; i < list.size(); i++) {
            Object[] datosRecuperados = list.get(i);

            OrdenUbicacion orUbi = new OrdenUbicacion();
            orUbi.setEdificio((String) datosRecuperados[0]);
            String x = (String)datosRecuperados[1];
            orUbi.setPiso(Integer.parseInt(x));
            orUbi.setArea((String) datosRecuperados[2]);
            orUbi.setDescripcionUbicacion((String) datosRecuperados[3]);
            orUbi.setClave((String)datosRecuperados[4]);
            orUbi.setFolio((Integer)datosRecuperados[5]);
            orUbi.setFecha((String)datosRecuperados[6]);
            orUbi.setInicio((String) datosRecuperados[7]);
            orUbi.setFin((String) datosRecuperados[8]);
            orUbi.setSolicitante((String) datosRecuperados[9]);
            orUbi.setTipoEquipo((String) datosRecuperados[10]);
            orUbi.setMarca((String) datosRecuperados[11]);
            orUbi.setModelo((String) datosRecuperados[12]);
            orUbi.setNsEquipo((String) datosRecuperados[13]);
            orUbi.setNiEquipo((String) datosRecuperados[14]);
            orUbi.setFalla((String) datosRecuperados[15]);
            orUbi.setSolucion((String) datosRecuperados[16]);
            orUbi.setObservaciones((String) datosRecuperados[17]);
            orUbi.setRealizo((String) datosRecuperados[18]);

            listaOrden.add(orUbi);

        }
        sesion.close();
        return listaOrden;
    }

    public List<OrdenUbicacion> consultarOrdenNs(String ns, String
fecha1, String fecha2)
            throws ParseException {
        Session sesion = null;
        List<Object[]> list;
```

```
        List<OrdenUbicacion> listaOrden = new LinkedList();
        try {
            sesion = CrearConexion.getSessionFactory().openSession();
        } catch (ExceptionInInitializerError ex) {
            System.out.println("No se pudo crear sesion");
            throw new ExceptionInInitializerError(ex);
        }

        String sentencia = "SELECT UBI.edificio, UBI.piso, UBI.area,
ORD.descripcionUbicacion, ORD.clave, ORD.folio, ORD.fecha, ORD.inicio,
ORD.fin,"
                + " ORD.solicitante, ORD.tipoEquipo, ORD.marca,
ORD.modelo, ORD.nsEquipo, ORD.niEquipo, ORD.falla, ORD.solucion,"
                + " ORD.observaciones, ORD.realizo FROM OrdenServicio AS
ORD, Ubicacion AS UBI "
                + " WHERE ORD.idUbicacion = UBI.id"
                + " AND ORD.nsEquipo = :ns"
                + " AND (ORD.fecha BETWEEN :ordfecha1 AND :ordfecha2) "
                + " ORDER BY ORD.fecha DESC";
        Query query = sesion.createQuery(sentencia);
        query.setParameter("ns", ns);
        query.setParameter("ordfecha1", fecha1);
        query.setParameter("ordfecha2", fecha2);
        list = query.list();

        for (int i = 0; i < list.size(); i++) {
            Object[] datosRecuperados = list.get(i);

            OrdenUbicacion orUbi = new OrdenUbicacion();
            orUbi.setEdificio((String) datosRecuperados[0]);
            String x = (String)datosRecuperados[1];
            orUbi.setPiso(Integer.parseInt(x));
            orUbi.setArea((String) datosRecuperados[2]);
            orUbi.setDescripcionUbicacion((String) datosRecuperados[3]);
            orUbi.setClave((String)datosRecuperados[4]);
            orUbi.setFolio((Integer)datosRecuperados[5]);
            orUbi.setFecha((String)datosRecuperados[6]);
            orUbi.setInicio((String) datosRecuperados[7]);
            orUbi.setFin((String) datosRecuperados[8]);
            orUbi.setSolicitante((String) datosRecuperados[9]);
            orUbi.setTipoEquipo((String) datosRecuperados[10]);
            orUbi.setMarca((String) datosRecuperados[11]);
            orUbi.setModelo((String) datosRecuperados[12]);
            orUbi.setNsEquipo((String) datosRecuperados[13]);
            orUbi.setNiEquipo((String) datosRecuperados[14]);
            orUbi.setFalla((String) datosRecuperados[15]);
            orUbi.setSolucion((String) datosRecuperados[16]);
            orUbi.setObservaciones((String) datosRecuperados[17]);
            orUbi.setRealizo((String) datosRecuperados[18]);

            listaOrden.add(orUbi);

        }
        sesion.close();
        return listaOrden;
    }

    public List<OrdenUbicacion> consultarOrdenNi(String ni, String
fecha1, String fecha2)
```

```
            throws ParseException {
        Session sesion = null;
        List<Object[]> list;
        List<OrdenUbicacion> listaOrden = new LinkedList();
        try {
            sesion = CrearConexion.getSessionFactory().openSession();
        } catch (ExceptionInInitializerError ex) {
            System.out.println("No se pudo crear sesion");
            throw new ExceptionInInitializerError(ex);
        }

        String sentencia = "SELECT UBI.edificio, UBI.piso, UBI.area,
ORD.descripcionUbicacion, ORD.clave, ORD.folio, ORD.fecha, ORD.inicio,
ORD.fin,"
                + " ORD.solicitante, ORD.tipoEquipo, ORD.marca,
ORD.modelo, ORD.nsEquipo, ORD.niEquipo, ORD.falla, ORD.solucion,"
                + " ORD.observaciones, ORD.realizo FROM OrdenServicio AS
ORD, Ubicacion AS UBI "
                + " WHERE ORD.idUbicacion = UBI.id"
                + " AND ORD.niEquipo = :ni"
                + " AND (ORD.fecha BETWEEN :ordfecha1 AND :ordfecha2) "
                + " ORDER BY ORD.fecha DESC";
        Query query = sesion.createQuery(sentencia);
        query.setParameter("ni", ni);
        query.setParameter("ordfecha1", fecha1);
        query.setParameter("ordfecha2", fecha2);
        list = query.list();

        for (int i = 0; i < list.size(); i++) {
            Object[] datosRecuperados = list.get(i);

            OrdenUbicacion orUbi = new OrdenUbicacion();
            orUbi.setEdificio((String) datosRecuperados[0]);
            String x = (String)datosRecuperados[1];
            orUbi.setPiso(Integer.parseInt(x));
            orUbi.setArea((String) datosRecuperados[2]);
            orUbi.setDescripcionUbicacion((String) datosRecuperados[3]);
            orUbi.setClave((String)datosRecuperados[4]);
            orUbi.setFolio((Integer)datosRecuperados[5]);
            orUbi.setFecha((String)datosRecuperados[6]);
            orUbi.setInicio((String) datosRecuperados[7]);
            orUbi.setFin((String) datosRecuperados[8]);
            orUbi.setSolicitante((String) datosRecuperados[9]);
            orUbi.setTipoEquipo((String) datosRecuperados[10]);
            orUbi.setMarca((String) datosRecuperados[11]);
            orUbi.setModelo((String) datosRecuperados[12]);
            orUbi.setNsEquipo((String) datosRecuperados[13]);
            orUbi.setNiEquipo((String) datosRecuperados[14]);
            orUbi.setFalla((String) datosRecuperados[15]);
            orUbi.setSolucion((String) datosRecuperados[16]);
            orUbi.setObservaciones((String) datosRecuperados[17]);
            orUbi.setRealizo((String) datosRecuperados[18]);

            listaOrden.add(orUbi);

        }
        sesion.close();
        return listaOrden;
    }
```

```
    public OrdenUbicacion getOrdenporClave(String clave, String fecha1,
String fecha2)throws ParseException {
        Session sesion = null;
        List<Object[]> list;
        OrdenUbicacion orUbi = new OrdenUbicacion();
        try {
            sesion = CrearConexion.getSessionFactory().openSession();
        } catch (ExceptionInInitializerError ex) {
            System.out.println("No se pudo crear sesion");
            throw new ExceptionInInitializerError(ex);
        }

        String sentencia = "SELECT UBI.edificio, UBI.piso, UBI.area,
ORD.descripcionUbicacion, "
                + "ORD.clave, ORD.folio, ORD.fecha, ORD.inicio, ORD.fin,
ORD.solicitante, "
                + "ORD.tipoEquipo, ORD.marca, ORD.modelo, ORD.nsEquipo,
ORD.niEquipo, "
                + "ORD.falla, ORD.solucion, ORD.observaciones,
ORD.realizo "
                + "FROM OrdenServicio AS ORD, Ubicacion AS UBI "
                + "WHERE ORD.idUbicacion = UBI.id "
                + "AND ORD.clave = :clave "
                + "AND (ORD.fecha BETWEEN :ordfecha1 AND :ordfecha2) "
                + "ORDER BY ORD.fecha DESC";
        Query query = sesion.createQuery(sentencia);
        query.setParameter("clave", clave);
        query.setParameter("ordfecha1", fecha1);
        query.setParameter("ordfecha2", fecha2);
        list = query.list();

        for (int i = 0; i < list.size(); i++) {
            Object[] datosRecuperados = list.get(i);

            orUbi.setEdificio((String) datosRecuperados[0]);
            String x = (String)datosRecuperados[1];
            orUbi.setPiso(Integer.parseInt(x));
            orUbi.setArea((String) datosRecuperados[2]);
            orUbi.setDescripcionUbicacion((String) datosRecuperados[3]);
            orUbi.setClave((String)datosRecuperados[4]);
            orUbi.setFolio((Integer)datosRecuperados[5]);
            orUbi.setFecha((String) datosRecuperados[6]);
            orUbi.setInicio((String) datosRecuperados[7]);
            orUbi.setFin((String) datosRecuperados[8]);
            orUbi.setSolicitante((String) datosRecuperados[9]);
            orUbi.setTipoEquipo((String) datosRecuperados[10]);
            orUbi.setMarca((String) datosRecuperados[11]);
            orUbi.setModelo((String) datosRecuperados[12]);
            orUbi.setNsEquipo((String) datosRecuperados[13]);
            orUbi.setNiEquipo((String) datosRecuperados[14]);
            orUbi.setFalla((String) datosRecuperados[15]);
            orUbi.setSolucion((String) datosRecuperados[16]);
            orUbi.setObservaciones((String) datosRecuperados[17]);
            orUbi.setRealizo((String) datosRecuperados[18]);

        }
        sesion.close();
        return orUbi;
```

```
    }


    public OrdenUbicacion getOrdenporClave(String clave)
            throws ParseException {
        Session sesion = null;
        List<Object[]> list;
        OrdenUbicacion orUbi = new OrdenUbicacion();
        try {
            sesion = CrearConexion.getSessionFactory().openSession();
        } catch (ExceptionInInitializerError ex) {
            System.out.println("No se pudo crear sesion");
            throw new ExceptionInInitializerError(ex);
        }

        String sentencia = "SELECT UBI.edificio, UBI.piso, UBI.area,
ORD.descripcionUbicacion, "
                + "ORD.clave, ORD.folio, ORD.fecha, ORD.inicio, ORD.fin,
ORD.solicitante, "
                + "ORD.tipoEquipo, ORD.marca, ORD.modelo, ORD.nsEquipo,
ORD.niEquipo, "
                + "ORD.falla, ORD.solucion, ORD.observaciones,
ORD.realizo "
                + "FROM OrdenServicio AS ORD, Ubicacion AS UBI "
                + "WHERE ORD.idUbicacion = UBI.id "
                + "AND ORD.clave = :clave "
                + "ORDER BY ORD.fecha DESC";
        Query query = sesion.createQuery(sentencia);
        query.setParameter("clave", clave);
        list = query.list();

        for (int i = 0; i < list.size(); i++) {
            Object[] datosRecuperados = list.get(i);

            orUbi.setEdificio((String) datosRecuperados[0]);
            String x = (String)datosRecuperados[1];
            orUbi.setPiso(Integer.parseInt(x));
            orUbi.setArea((String) datosRecuperados[2]);
            orUbi.setDescripcionUbicacion((String) datosRecuperados[3]);
            orUbi.setClave((String)datosRecuperados[4]);
            orUbi.setFolio((Integer)datosRecuperados[5]);
            orUbi.setFecha((String) datosRecuperados[6]);
            orUbi.setInicio((String) datosRecuperados[7]);
            orUbi.setFin((String) datosRecuperados[8]);
            orUbi.setSolicitante((String) datosRecuperados[9]);
            orUbi.setTipoEquipo((String) datosRecuperados[10]);
            orUbi.setMarca((String) datosRecuperados[11]);
            orUbi.setModelo((String) datosRecuperados[12]);
            orUbi.setNsEquipo((String) datosRecuperados[13]);
            orUbi.setNiEquipo((String) datosRecuperados[14]);
            orUbi.setFalla((String) datosRecuperados[15]);
            orUbi.setSolucion((String) datosRecuperados[16]);
            orUbi.setObservaciones((String) datosRecuperados[17]);
            orUbi.setRealizo((String) datosRecuperados[18]);

        }
        sesion.close();
        return orUbi;
    }
```