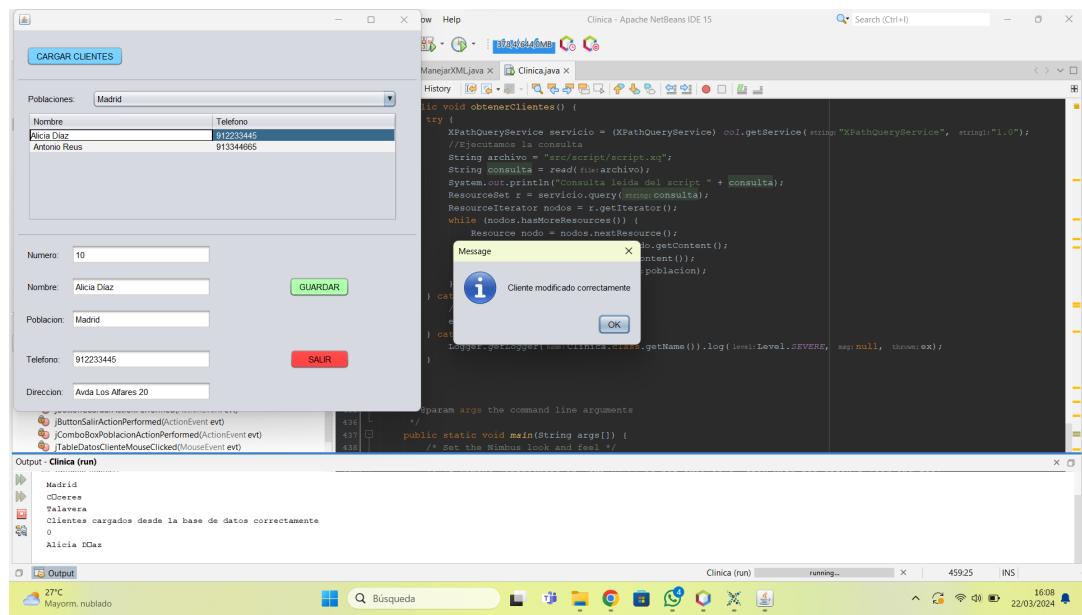
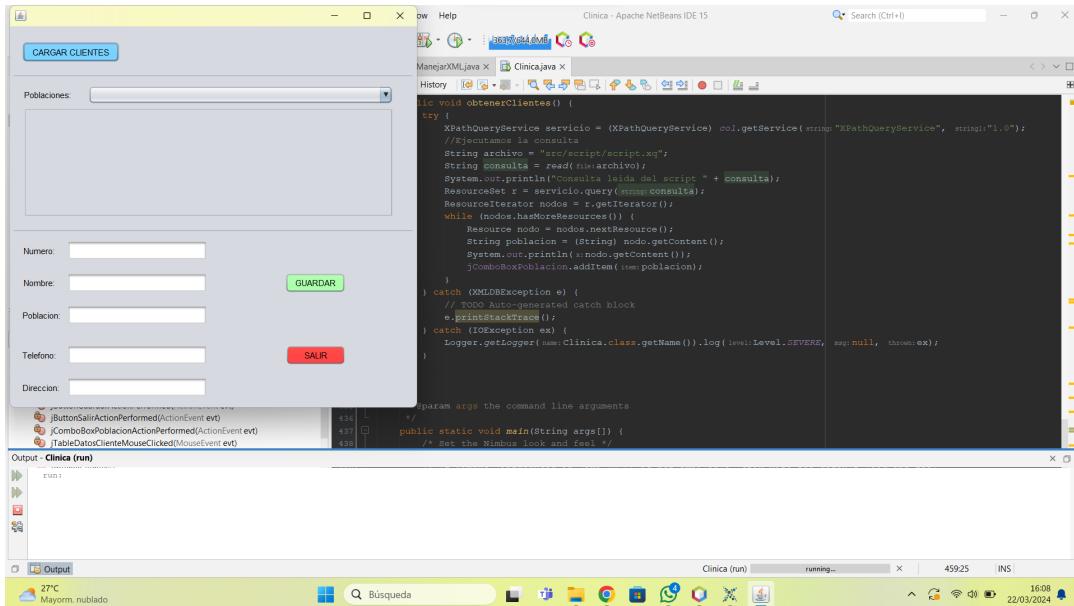


Tarea 6 para Acceso a Datos



Diego Manuel Carrasco Castañares

Hay que entregar un documento en pdf, con explicación y los pantallazos de la ejecución (muy importante) de la tarea. Si no se entrega, la nota tendrá una bajada de 2 puntos.

Manejo de bases de datos con el SGBD eXist. Esta tarea tiene dos partes.

Parte 1: 4 puntos

Para esta tarea crearemos, utilizando eXist-db (XQuery), una base de datos llamada estantería, con los 2 ficheros .xml que se adjunta. (0,5 puntos cada apartado)

1. Muestra o devuelve todas las personas.
2. Muestra los nombres de todas las personas.
3. Muestra los datos de la persona cuyo dni es 11111111
4. Muestra las ciudades que están asociadas a cada persona sin utilizar path absoluto.
5. Muestra nombre y apellidos de los autores de los libros
6. Muestra los libros con año mayor a 1990
7. Muestra el segundo libro.
8. Muestra las personas que viven en Madrid

Parte 2: (6 puntos)

Apartado 3.1 de la unidad eXist

Al igual que en los ejemplos puestos en el apartado recursos de la unidad. Se trata de que realices las consultas a la colección clínica y las ejecutes desde una aplicación Java con entorno gráfico. Las consultas que realices que sean constantes, es decir, que no requieran de ningún parámetro, debes guardarlas en un script.xq, (1 punto) que llamarás desde tu programa. (en el proyecto de ejemplo manejarXML.zip existe un método llamado read(String file) que devuelve un String y que puede ser utilizado en este ejercicio para leer los script.xq).

Para realizar los siguientes apartados deberás crear un interfaz. Al principio puede ser así, luego vais abriendo campos o los ponéis todos al principio.

Las consultas y/o modificaciones son:

1. Filtrar los clientes por población. Nuestro entorno gráfico dispondrá de un botón llamado cargar clientes, al pulsarlo se nos mostrará una lista desplegable en la que aparecerán todas las poblaciones de todos los clientes (sin repetición (1,5 puntos)

Cargar clientes

Poblaciones

Número

Nombre

Población

Teléfono

Dirección

Salir Guardar

Cargar clientes

Poblaciones

- Madrid
- Cáceres
- Talavera

Número

Nombre

Población

Teléfono

Dirección

Salir Guardar

2. Cuando seleccionemos una población de la lista desplegable se nos mostrarán el nombre y el teléfono de todos los clientes de la población seleccionada (1,5 puntos).

Nombre	Teléfono
Pilar Martín	927233490

Cargar clientes

Poblaciones Cáceres

Número

Nombre

Población

Teléfono

Dirección

Salir Guardar

3. En entorno gráfico me permitirá seleccionar un cliente de todos los que se han mostrado y que previamente habíamos filtrado por población. De este cliente mostraremos sus datos (1 punto) y podremos modificarlos y guardar estas modificaciones realizadas en la base de datos. (1,5 puntos).

Que funcione la opción Salir (0,5 puntos).

The screenshot shows a Java Swing application window titled "Clínica - Tarea 6 de AD". At the top, there is a button labeled "Cargar clientes". Below it, a dropdown menu shows "Poblaciones Cáceres". A table displays one client record:

Nombre	Teléfono
Pilar Martín	927233490

Below the table, there are five input fields with their corresponding values:

Número	20
Nombre	Pilar Martín
Población	Cáceres
Teléfono	927233490
Dirección	Avda de Madrid 6

At the bottom of the window are two buttons: "Salir" and "Guardar".

Criterios de puntuación. Total 10 puntos.

Se tendrá en cuenta que:

El funcionamiento correcto de los programas que se piden.

Cada apartado tiene su puntuación y descripción del mismo.

Recursos necesarios para realizar la Tarea.

Netbeans y eXist-db

Consejos y recomendaciones.

Realiza previamente todos los ejemplos de la unidad.

Indicaciones de entrega.

Una vez realizada la tarea elaborarás un único documento donde figuren el proyecto y un documento pdf donde figure la explicación y ejecución del proyecto. El envío se realizará a través de la plataforma de la forma establecida para ello, y el archivo se nombrará siguiendo las siguientes pautas:

apellido1_apellido2_nombre_ADxx_Tareaxxx_Entregaxx

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna Begoña Sánchez Mañas para la primera unidad del MP de AD, debería nombrar esta tarea como...

sanchez_manas_begona_AD06_Tarea06_Entrega01

INDICE:

Parte 1.

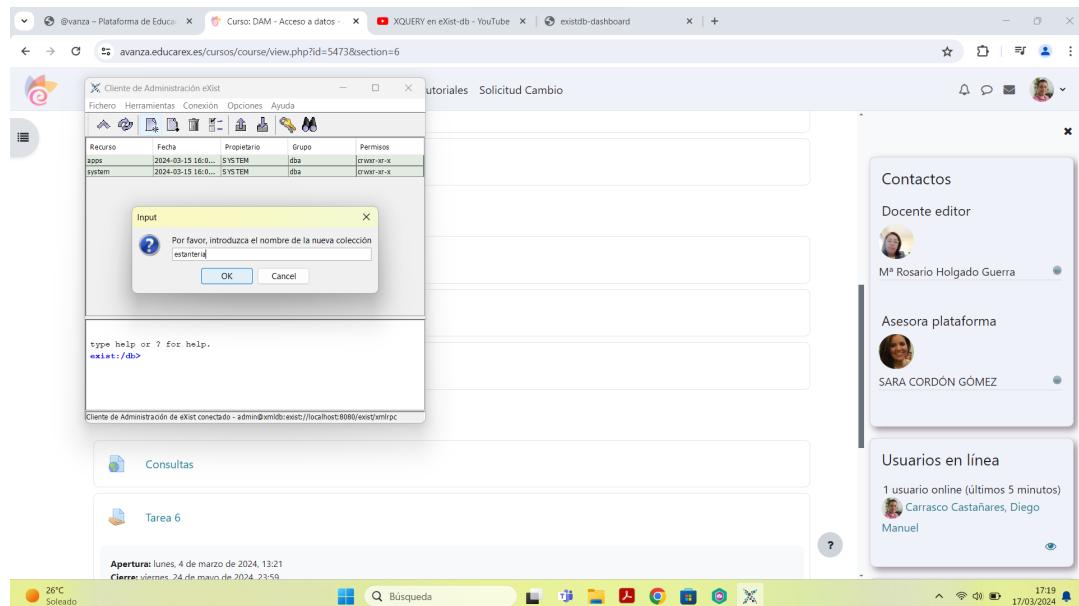
1.0.- Creación de la colección.....	07
1.1.- Muestra o devuelve todas las personas.....	08
1.2.- Muestra los nombres de todas las personas.....	09
1.3.- Muestra los datos de la persona cuyo dni es 11111111.....	09
1.4.- Muestra las ciudades sin path absoluto.....	10
1.5.- Muestra nombre y apellidos de los autores.....	10
1.6.- Muestra los libros con año mayor a 1990.....	11
1.7.- Muestra el segundo libro.....	11
1.8.- Muestra las personas que viven en Madrid.....	12

Parte 2.

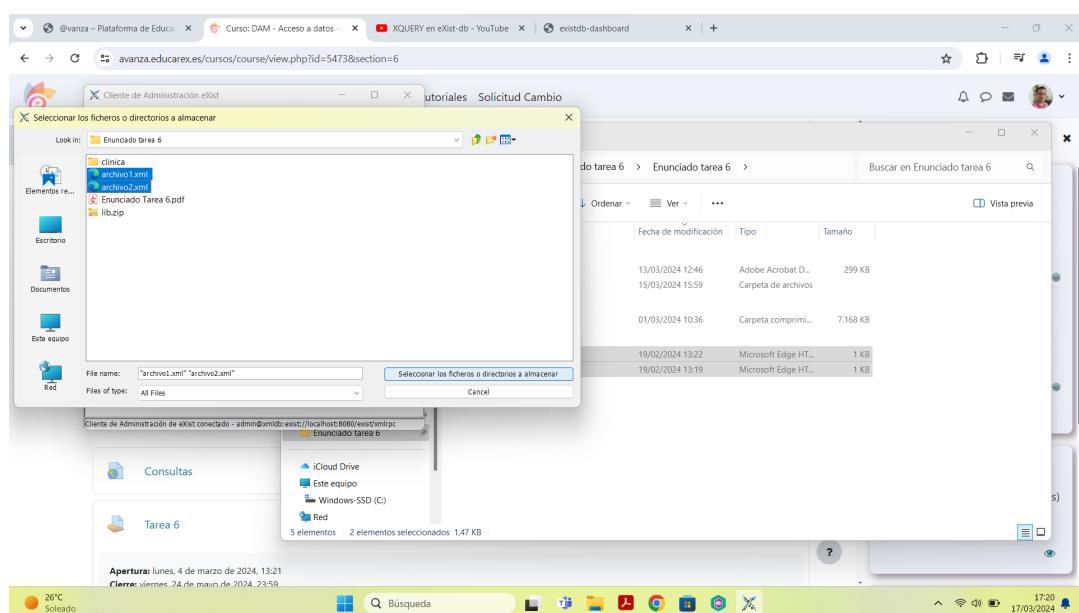
2.1.- Botón salir.....	14
2.2.- Mostrar el nombre y el teléfono de los clientes de población seleccionada.....	14
2.3.- Seleccionar un cliente de la tabla y modificar los datos.....	15
2.4.- Botón guardar.....	16
2.5.- Filtrar los clientes por población.....	17

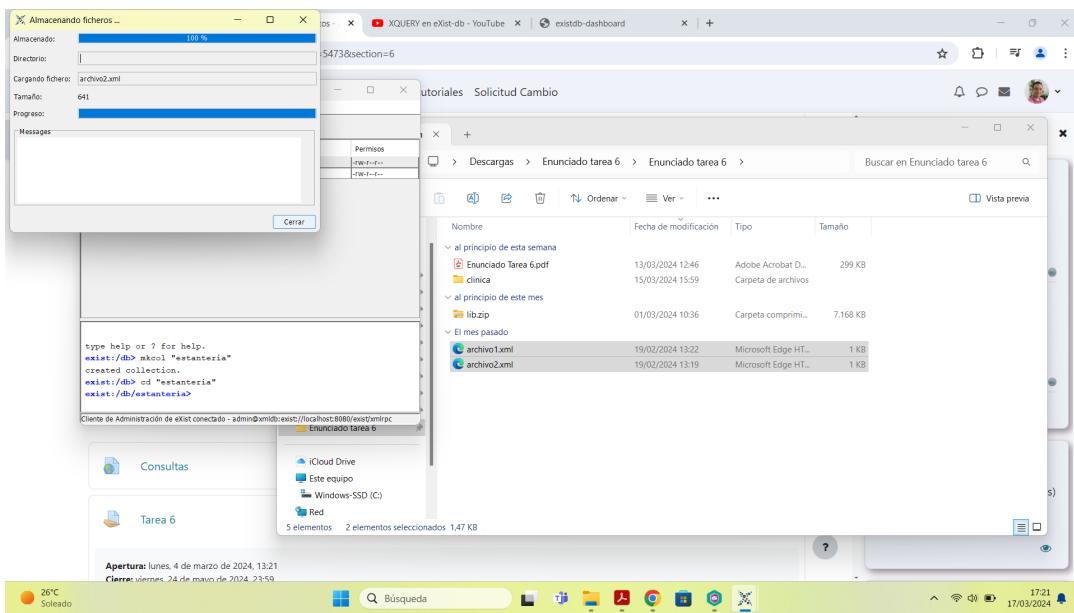
Lo primero que haremos será crear la colección llamada *estantería*.

Para ello abrimos eXist, abriremos el cliente de administración java, clicamos en añadir colección, le damos un nombre y guardamos.



Tras ello abrimos la colección creada y clicamos en añadir archivos, seleccionamos los dos archivos xml dados en la tarea y, una vez cargados, cerramos.





Con esto ya tenemos creada la colección y añadidos los archivos xml.

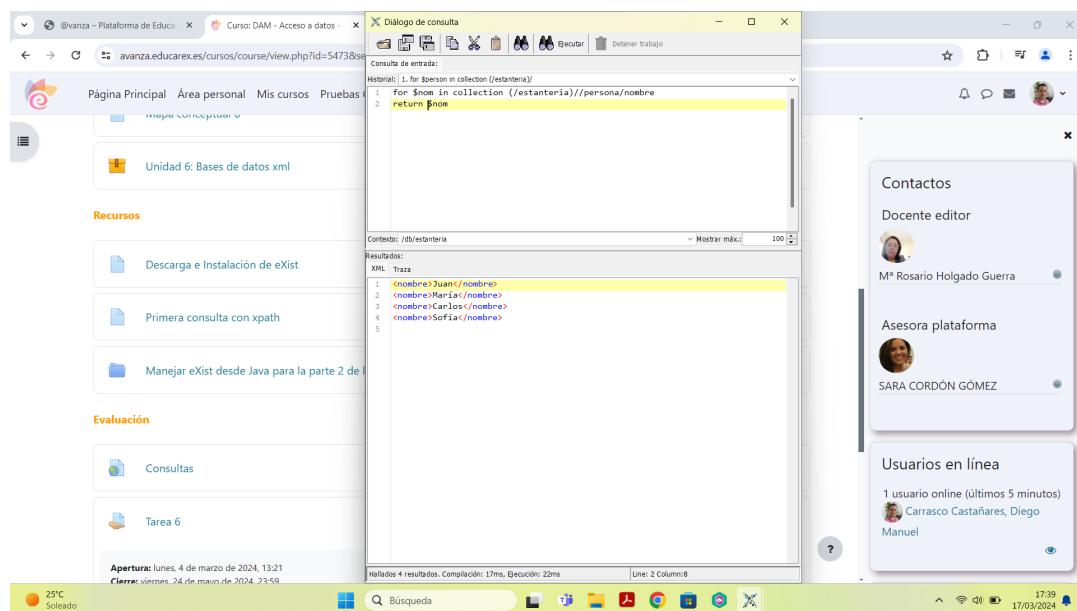
Ahora pasaremos a las consultas.

1.1.- Muestra o devuelve todas las personas.

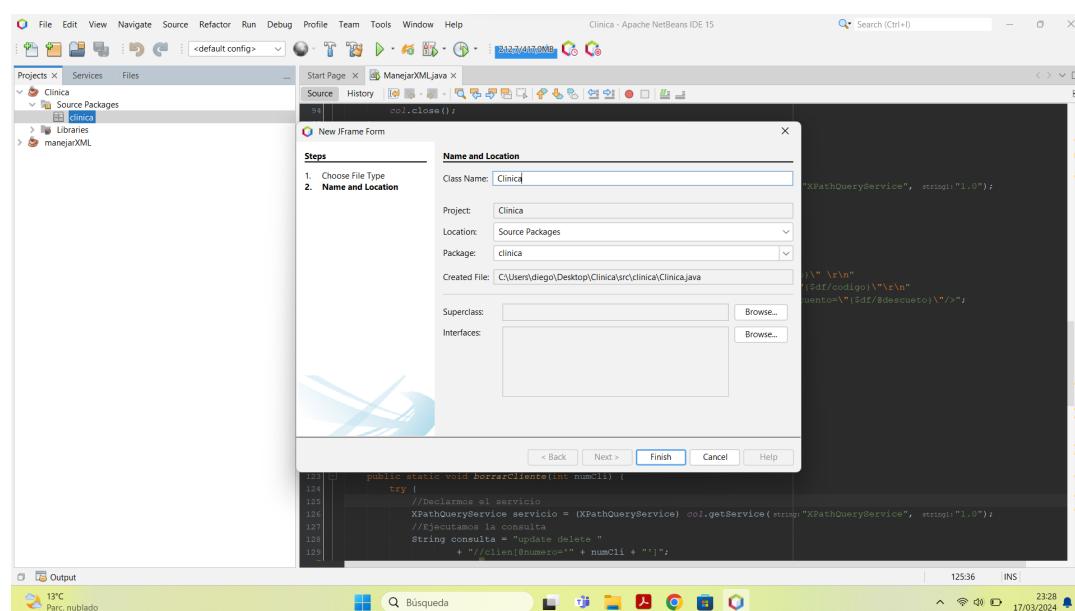
The screenshot shows a web-based XQuery editor interface:

- Plataforma de Educacion**: A sidebar with links to "Página Principal", "Área personal", "Mis cursos", "Pruebas", "Unidad 6: Bases de datos xml", "Recursos", "Descarga e Instalación de eXist", "Primera consulta con xpath", "Manejar eXist desde Java para la parte 2 de", "Evaluación", "Consultas", and "Tarea 6".
- Curso: DAM - Acceso a datos**: A browser tab showing the course page.
- Diálogo de consulta**: A central window with the following details:
 - Consulta de entrada:** `1. for $person in collection ('/estanteria')///persona
2. return $person`
 - Contexto:** /objestanteria
 - Resultados:** XML Traza (showing 25 numbered XML snippets representing person data)
 - Apertura:** lunes, 4 de marzo de 2024, 13:21
 - Cierre:** viernes, 24 de mayo de 2024, 23:59
- Contactos**: A sidebar listing "Docente editor" (M^a Rosario Holgado Guerra) and "Asesora plataforma" (SARA CORDÓN GÓMEZ).
- Usuarios en línea**: A sidebar showing "1 usuario online (últimos 5 minutos)" (Diego Manuel Carrasco Castañares).
- Barra de herramientas**: Standard browser toolbar.
- Estado del sistema**: Status bar at the bottom right showing date (17/03/2024), time (17:28), battery level, and signal strength.

1.2.- Muestra los nombres de todas las personas.



1.3.- Muestra los datos de la persona cuyo dni es 111111111.



1.4.- Muestra las ciudades sin path absoluto.

```

@vanza - Plataforma de Educación Curso: DAM - Acceso a datos - avanza.educarex.es/cursos/course/view.php?id=547385
Página Principal Área personal Mis cursos Pruebas Unidad 6: Bases de datos xml
Recursos Descarga e Instalación de eXist Primera consulta con xpath Manejar eXist desde Java para la parte 2 de I
Evaluación Consultas Tarea 6
Apertura: lunes, 4 de marzo de 2024, 13:21 Cierre: viernes, 24 de mayo de 2024, 23:59
25°C Soleado
    
```

Consulta de consulta

Consulta de entrada:

```

Historial: 1. for $persona in collection (/estanteria)
1   for $persona in /personas/persona
2   return concat("Nombre: ", $persona/nombre/text(), ", Ciudad: ", $persona/ciudad/text())
3

```

Contexto: /db/estanteria

Resultados: XML Traza

```

1 Nombre: Juan, Ciudad: Madrid
2 Nombre: María, Ciudad: Barcelona
3 Nombre: Carlos, Ciudad: Madrid
4 Nombre: Sofía, Ciudad: Valencia

```

Hallados 4 resultados. Compilación: 18ms, Ejecución: 22ms Line: 3 Column:1

Búsqueda

1.5.- Muestra nombre y apellidos de los autores

```

@vanza - Plataforma de Educación Curso: DAM - Acceso a datos - avanza.educarex.es/cursos/course/view.php?id=547385
Página Principal Área personal Mis cursos Pruebas Unidad 6: Bases de datos xml
Recursos Descarga e Instalación de eXist Primera consulta con xpath Manejar eXist desde Java para la parte 2 de I
Evaluación Consultas Tarea 6
Apertura: lunes, 4 de marzo de 2024, 13:21 Cierre: viernes, 24 de mayo de 2024, 23:59
24°C Parc. soleado
    
```

Consulta de consulta

Consulta de entrada:

```

Historial: 1. for $autor in collection (/estanteria) //autor
1   for $autor in /personas/autor
2   return concat("Nombre y apellidos: ", $autor/nombre/text(), " ", $autor/apellidos/text())
3

```

Contexto: /db/estanteria

Resultados: XML Traza

```

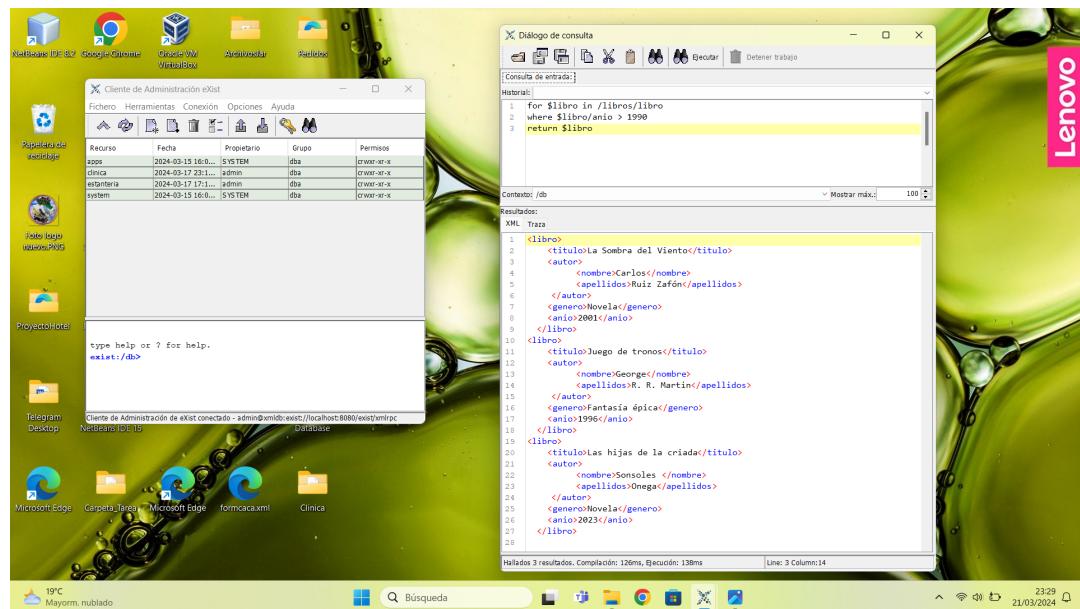
1 Nombre y apellidos: Carlos Ruiz Zafón
2 Nombre y apellidos: George R. R. Martin
3 Nombre y apellidos: Sonsoles Omega
4 Nombre y apellidos: Gabriel García Márquez

```

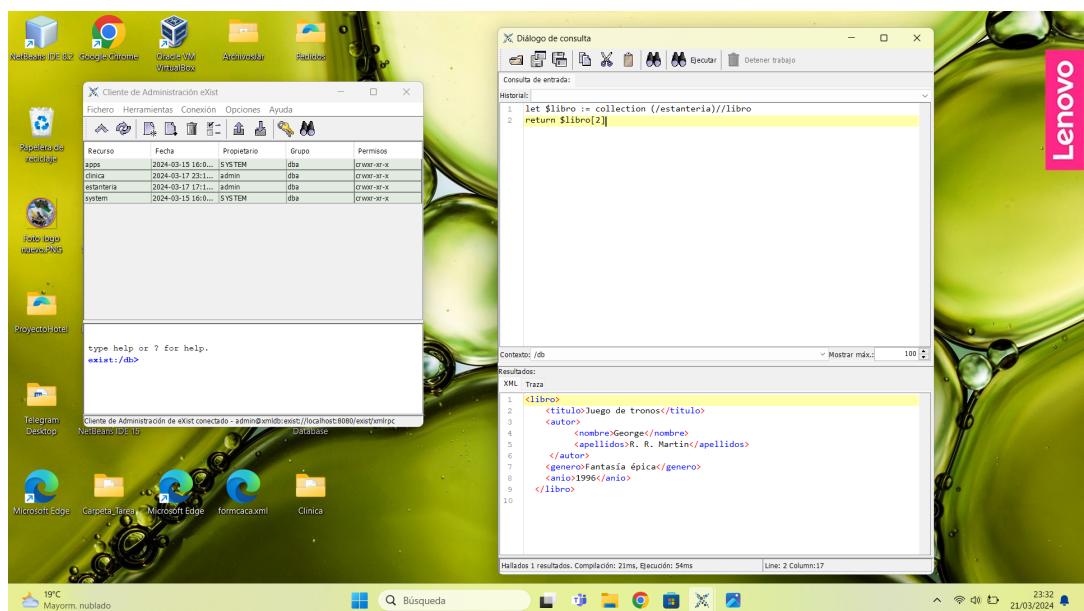
Hallados 4 resultados. Compilación: 22ms, Ejecución: 14ms Line: 2 Column:34

Búsqueda

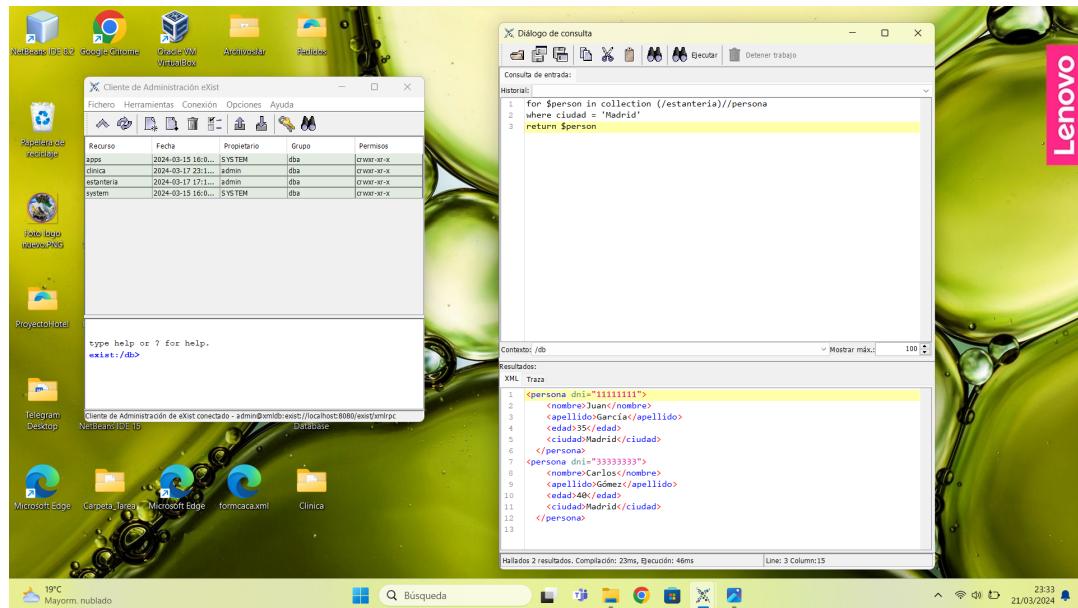
1.6.- Muestra los libros con año mayor a 1990.



1.7.- Muestra el segundo libro.



1.8.- Muestra las personas que viven en Madrid.

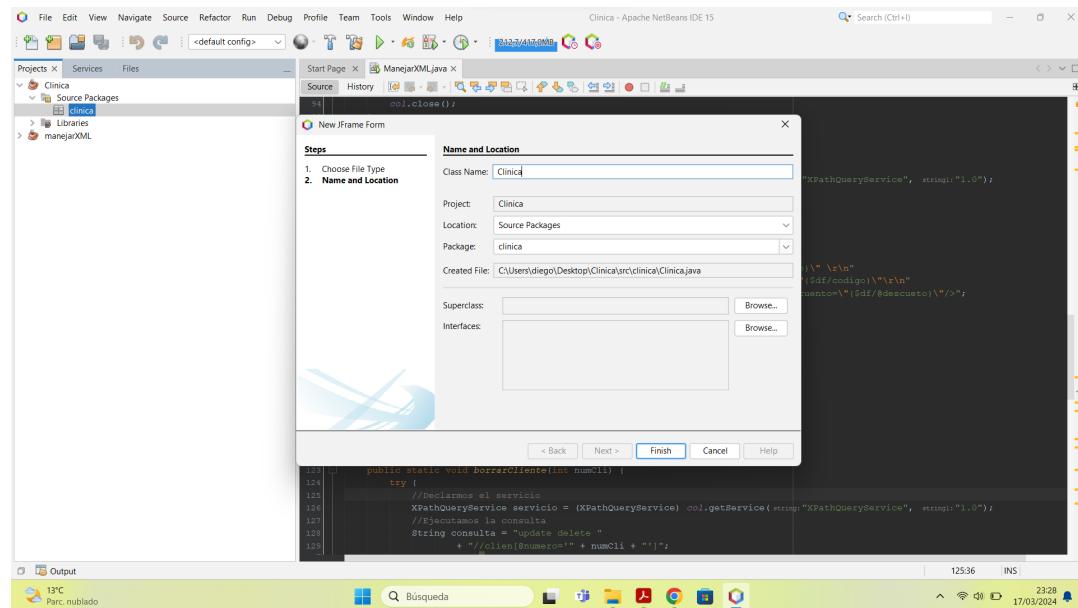


Con esto damos por finalizado la parte de uno de la tarea.

Ahora pasaremos a la parte dos.

Para esta segunda parte de la tarea empezaremos creando el proyecto en NetBeans, al cual llamaremos Clinica e importaremos las librerías indicadas en la tarea.

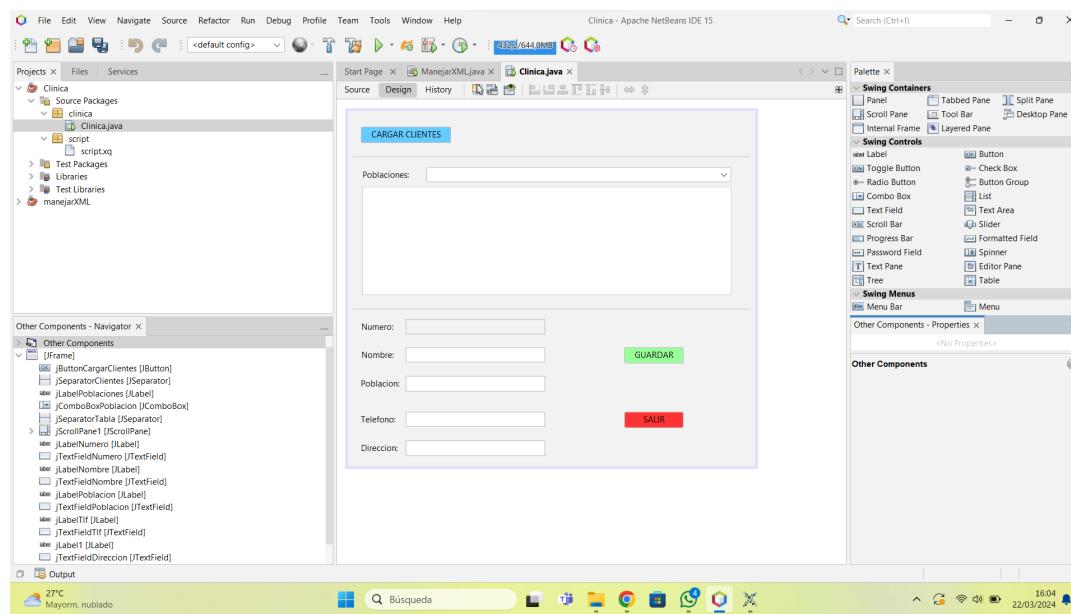
Tras ello crearemos un nuevo JFrame Form.



A continuación vamos a componer el JFrame.

Estará compuesto por un botón, que se encargará de realizar la conexión con la base de datos y cargar los clientes en un ComboBox, un ComboBox que mostrará los clientes extraídos de la base de datos, una tabla que nos mostrará el nombre y el teléfono

de los clientes seleccionados en el ComboBox, varios jTextField que almacenaran, de forma individual cada dato extraido del cliente (numero, nombre, población, teléfono y dirección) y desde donde modificaremos los datos que queramos que sean modificados en la base de datos, un botón que será el encargado de actualizar en la base de datos los datos modificados en los jTextField y un botón de salir que se encargará de cerrar la conexión con la base de datos y cerrar el programa tras un mensaje de confirmación.



En el código del JFrame empezaremos componiendo el método constructor el cual establecerá la propiedad Visible de la tabla a false (para que no se muestre al arrancar el programa), contendrá un string con el driver a utilizar para la conexión, creará una instancia a la base de datos y su conexión con el puerto, el usuario y la contraseña que pusimos.

```

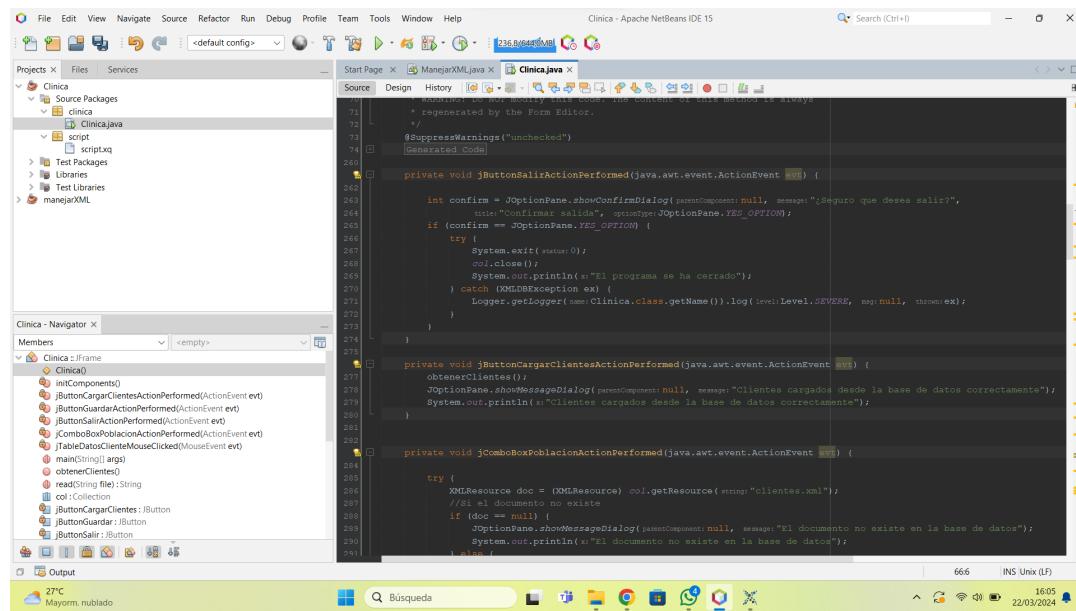
public class Clinica extends javax.swing.JFrame {
    /**
     * Creates new form Clinica
     */
    public static Collection col = null;
    public Clinica() {
        try {
            initComponents();
            jTableDatosClinica.setVisible(false);
            final String driver = "org.existxmldb.DatabaseImpl";
            // initialize database driver
            Class cl = Class.forName(driver);
            Database database = (Database) cl.getDeclaredConstructor().newInstance();
            database.setProperty("create-database", "true");
            DatabaseManager.registerDatabase(database);

            // Conectarnos a una colección
            col = DatabaseManager.getCollection(url: "xmldb:exist://localhost:8080/exist/xmldb/db/clinica",
                name: "clinica");
            if (col == null) {
                Logger.getLogger("Clinica.class.getName()").log(Level.SEVERE, null, ex);
            }
        } catch (NoSuchMethodException ex) {
            Logger.getLogger("Clinica.class.getName()").log(Level.SEVERE, null, ex);
        } catch (SecurityException ex) {
            Logger.getLogger("Clinica.class.getName()").log(Level.SEVERE, null, ex);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger("Clinica.class.getName()").log(Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            Logger.getLogger("Clinica.class.getName()").log(Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            Logger.getLogger("Clinica.class.getName()").log(Level.SEVERE, null, ex);
        } catch (IllegalArgumentException ex) {
            Logger.getLogger("Clinica.class.getName()").log(Level.SEVERE, null, ex);
        } catch (InvocationTargetException ex) {
            Logger.getLogger("Clinica.class.getName()").log(Level.SEVERE, null, ex);
        } catch (XMldbException ex) {
            ...
        }
    }
}

```

El botón salir contendrá un int, que guardará la opción retornada por el mensaje de confirmación y un if, que si se confirma la salida, cerrara la conexión a la base de datos y el programa.

El botón cargarClientes llamará al método obtenerClientes (que mas adelante detallaremos) y mostrara un mensaje si se han cargado correctamente.



El el evento del ComboBox se creara una instancia al documento mediante el XMLResource y mediante un if y la instancia creada comprobará si existe el documento, en cuyo caso extraerá la población seleccionada del item y la guardará en un variable.

Creará otro string con la consulta a la colección para extraer los cliente que tienen como población la seleccionada en el ComboBox.

Creará un servicio para lanzar la consulta, un ResourceSet que guardará el resultado devuelto por la consulta, definirá el modelo de la tabla y el nombre de las columnas (nombre y teléfono en este caso).

Creará un ResourceIterator para iterar sobre el resultado, que mediante un while creará un Resource, un array de tipo string donde guardará el resultado obtenido y añadirá, mediante el método adRow, el resultado a la tabla para posteriormente, hacerla visible.

```

private void jcComboBoxPoblacionActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        XMLResource doc = (XMLResource) col.getResource("clientes.xml");
        //Si el documento no existe
        if (doc == null) {
            JOptionPane.showMessageDialog(parentComponent, null, "El documento no existe en la base de datos");
            System.out.println("El documento no existe en la base de datos");
        } else {
            String itemSelect = (String) jcComboBoxPoblacion.getSelectedItem();
            String consulta = "for $cliente in doc('clientes.xml')/cliente\n" +
                "where $cliente/poblacion ='" + itemSelect + "'\n" +
                "return concat($cliente/nombre/text(), ', ', $cliente/tlf/text())";
            //Solicitamos el servicio para consulta
            XPathQueryService servicio = (XPathQueryService) col.getService("XPathQueryService", string("1.0"));
            // Ejecutamos la consulta
            ResourceSet resultado = servicio.query(string(consulta));
            //Definimos el modelo de la tabla
            DefaultTableModel modelo = new DefaultTableModel();
            modelo.addColumn(columnName:"Nombre");
            modelo.addColumn(columnName:"Telefono");
            //Iteramos los resultados
            ResourceIterator i = resultado.getIterator();
            while (i.hasMoreResources()) {
                String[] cliente = i.next().getContent();
                //Obtenemos los datos del cliente
                String[] cliente = ((String) r.getContent()).split(repr(":"));
                modelo.addRow(newRow:cliente);
            }
            jTableDatosCliente.setModel(modelo);
            jTableDatosCliente.setVisible(sFlag: true);
            jTableDatosCliente.getTableHeader().setVisible(sFlag: true);
            jTableDatosCliente.setEnabled(enabled: true);
        }
    } catch (XMldbException ex) {
        JOptionPane.showMessageDialog(parentComponent, null, "Error: " + ex + "\nInténtalo nuevamente", title: "Error");
    } catch (XMLDBException ex) {
        Logger.getLogger(Clinica.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

El evento de la tabla contendrá un int donde se guardará la fila seleccionada, un if que comprobará si se ha seleccionado alguna fila, en cuyo caso, creará un objeto para almacenar la fila seleccionada, creará un XPathQueryService, creará un string que almacenará la consulta a la colección en función de la fila seleccionada y creará un ResourceSet que almacenará el resultado devuelto pro la consulta.

Tras ello, mediante un if comprobará si la longitud de ResourceSet es mayor a 0, en cuyo caso creará un ResourceIterator y, mediante un while, lanzará una consultas a la colección para extraer cada uno de los datos tales como el número, nombre, población, teléfono y dirección y los mostrará en el jTextField correspondiente.

Cada consulta estará compuesta por un Resource, el string que contendrá la consulta, el ResourceSet que lanzará la consulta y el sexText correspondiente.

```

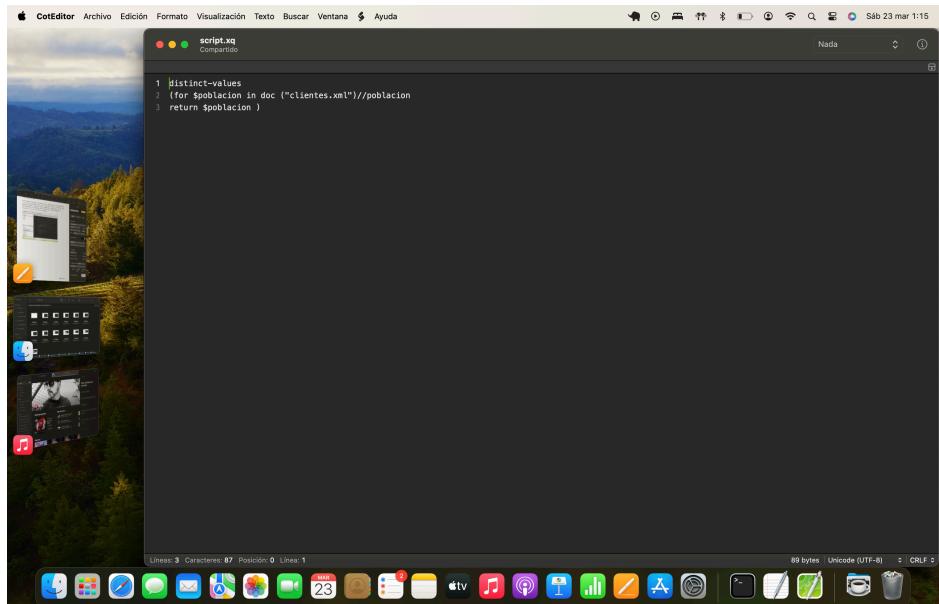
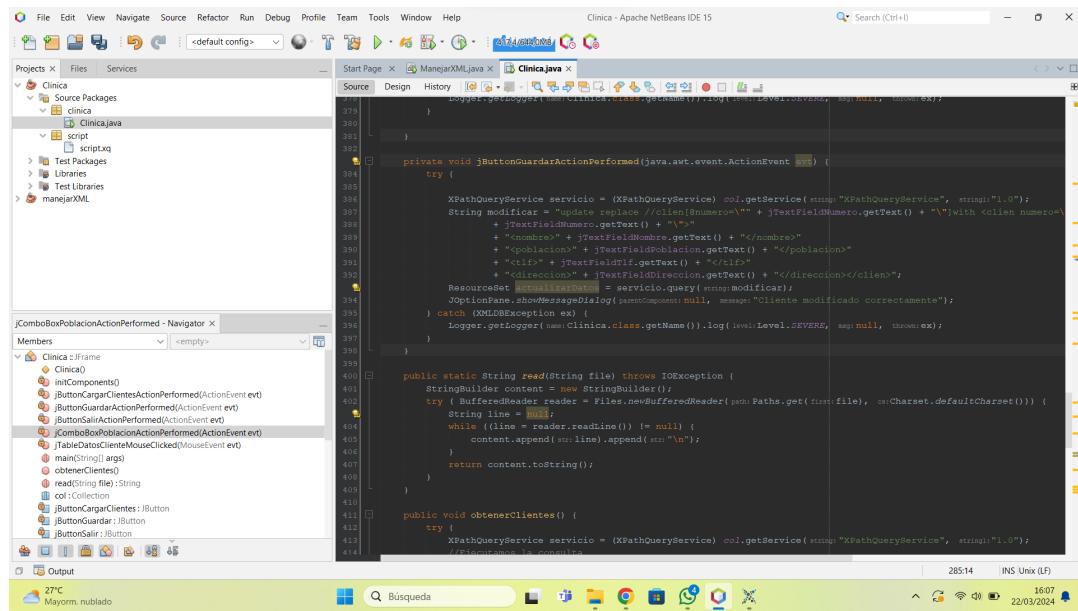
private void jcComboBoxPoblacionActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        XMLResource poblacion = queryService.query(string(buscarPoblacion));
        jTextFieldPoblacion.setText(poblacion.getIterator().nextResource().getContent().toString());
        String buscarNombre = "for $cliente in collection('clinica')//cliente\n" +
            "where $cliente/nombre ='" + filaselect + "'\n" +
            "return string($cliente/nombre)";
        ResourceSet nombre = queryService.query(string(buscarNombre));
        jTextFieldNombre.setText(nombre.getIterator().nextResource().getContent().toString());
        String buscarDireccion = "for $cliente in collection('clinica')//cliente\n" +
            "where $cliente/direccion ='" + filaselect + "'\n" +
            "return string($cliente/direccion)";
        ResourceSet direccion = queryService.query(string(buscarDireccion));
        jTextFieldDireccion.setText(direccion.getIterator().nextResource().getContent().toString());
        String buscarNumero = "for $cliente in collection('clinica')//cliente\n" +
            "where $cliente/nombre ='" + filaselect + "'\n" +
            "return string($cliente/nombre)";
        ResourceSet numero = queryService.query(string(buscarNumero));
        jTextFieldNumero.setText(numero.getIterator().nextResource().getContent().toString());
    } catch (HeadlessException ex) {
        JOptionPane.showMessageDialog(parentComponent, null, "Error: " + ex + "\nInténtalo nuevamente", title: "Error");
    } catch (XMldbException ex) {
        Logger.getLogger(Clinica.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

El evento del botón guardar contará con XParthQueryService, un string que contendrá la consulta la colección que tendrá la función de actualizar cada campo recogido del jTextField, un ResourceSet que lanzará dicha consulta y un JOptionPane que mostrará si se ha realizado la actualización de los datos.

Dispondremos además, de un método read que será el encargado de leer el script que contendrá la culta para recuperar todo los clientes por países.

Dicho método recibirá un archivo File, contendrá un StringBuilder, un string que guardará cada línea leída del archivo, un while, que mientras queden líneas por leer, guardará, concatenado, en el StringBuilder el contenido del fichero para posteriormente retornar un string con dicho contenido.

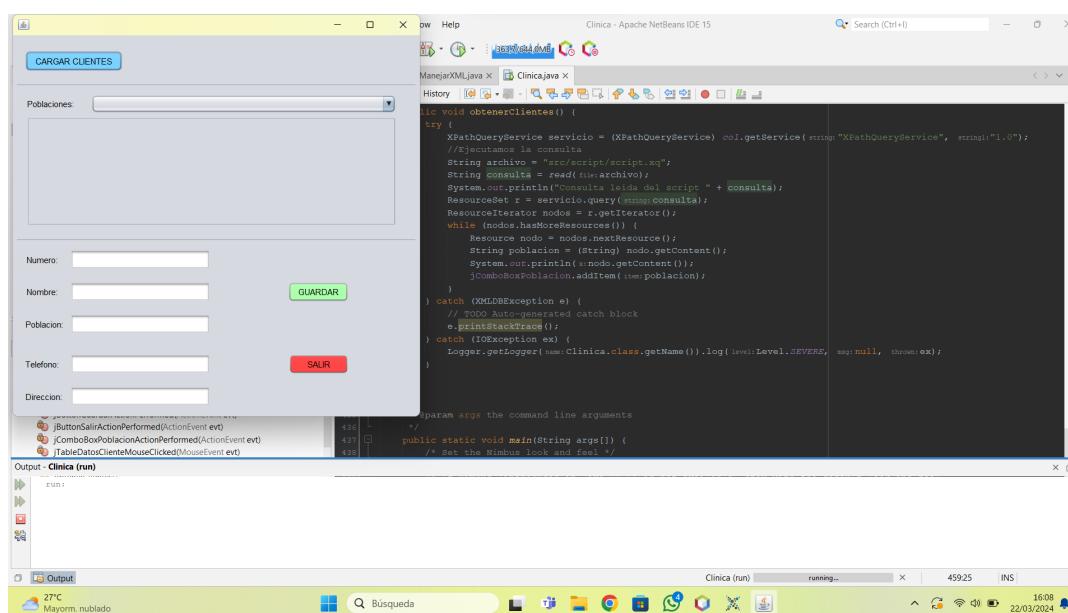


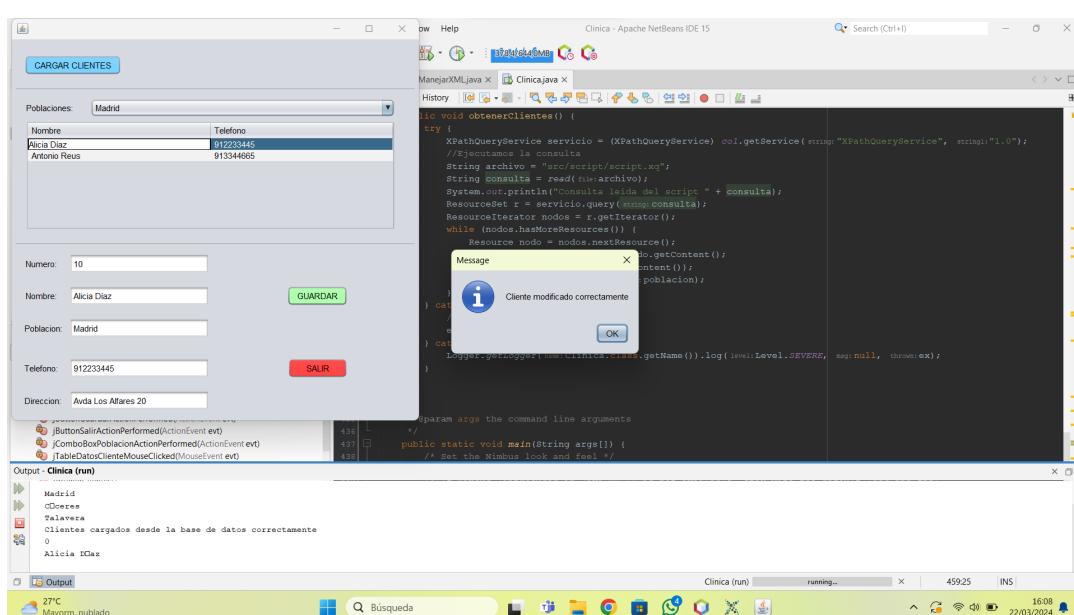
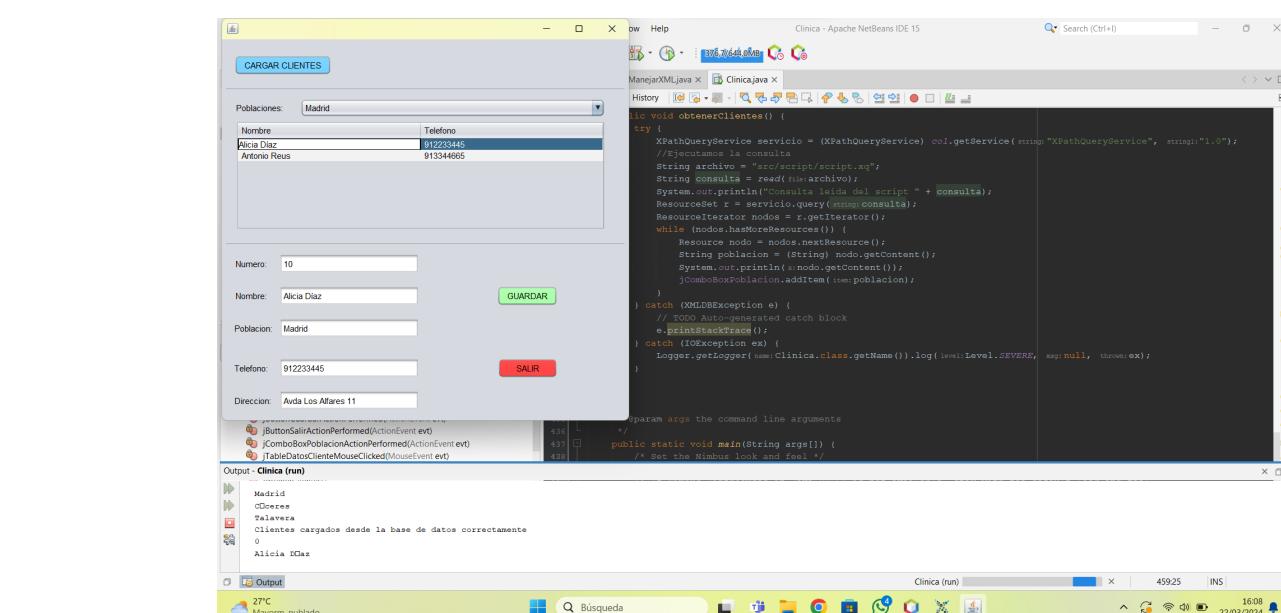
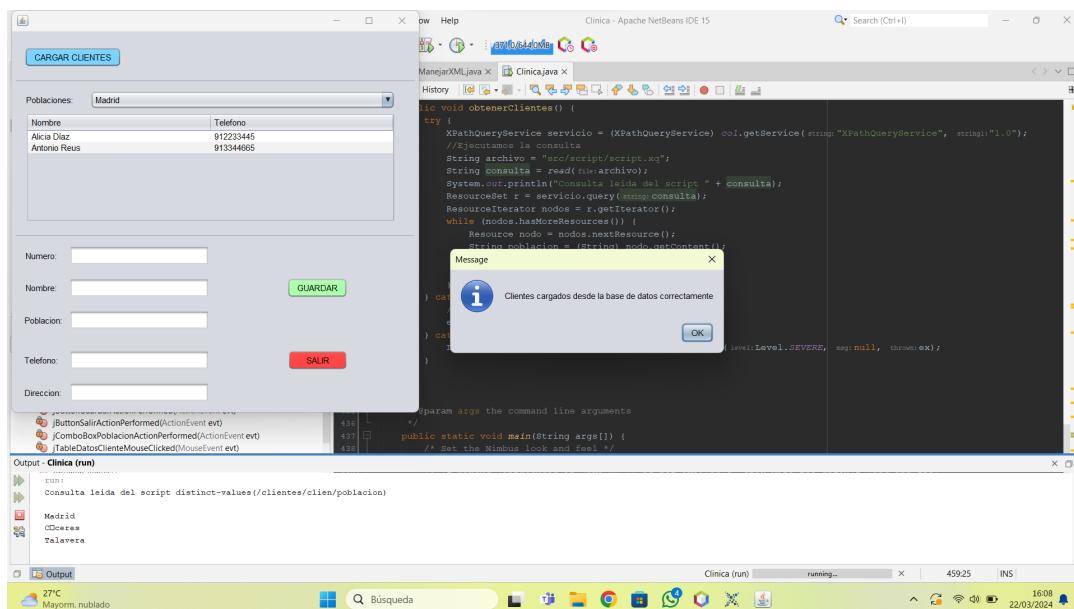
El método obtenerClientes contendrá un XPathQueryService, un string que contendrá la ruta hasta el archivo, un string que llamará al método read pasándole la variable que contiene la ruta del archivo, un ResourceSet que ejecutara la consulta pasándole la variable que la contiene, un ResourceIterator, un while, que recorrerá el ResourceIterator, creará un Resource que almacenará el contenido extraído, un string que guardará la población y posteriormente la añadirá al ComboBox como ítem.

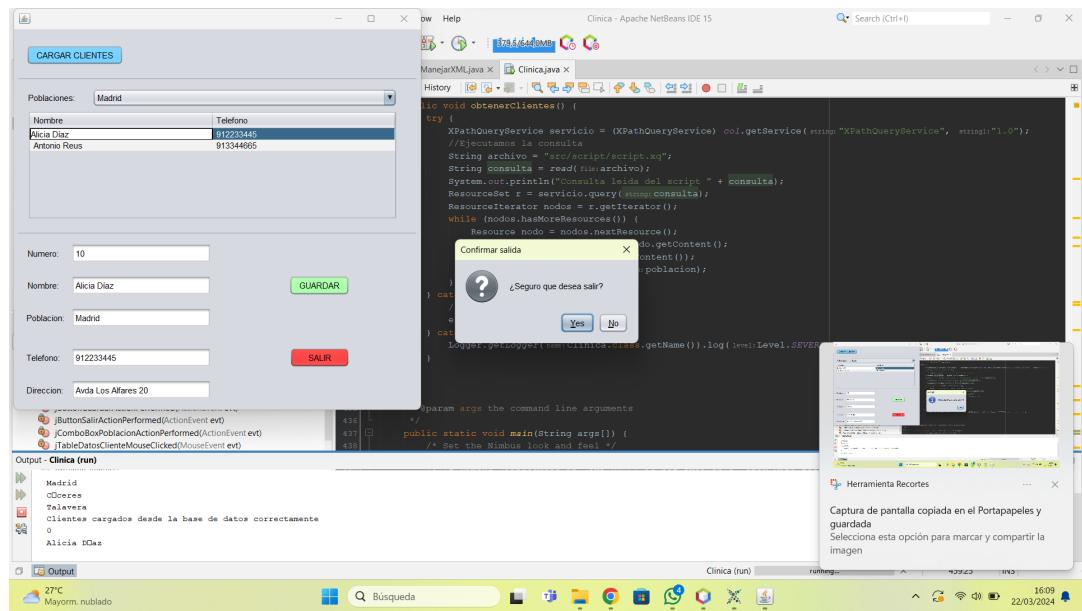
```

public void obtenerClientes() {
    try {
        XPathQueryService servicio = (XPathQueryService) col.getService("XPathQueryService", "1.0");
        //Ejecutamos la consulta
        String archivo = "src/script/script.xq";
        String consulta = read(script);
        System.out.println("Consulta leida del script " + consulta);
        ResourceSet resultado = servicio.query("xpath://" + archivo);
        ResourceIterator nodos = resultado.iterator();
        while (nodos.hasMoreResources()) {
            Resource nodo = nodos.nextResource();
            String poblacion = (String) nodo.getContent();
            System.out.println((String) nodo.getContent());
            jComboBoxPoblacion.addItem(item:poblacion);
        }
    } catch (XMLODEException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException ex) {
        Logger.getLogger(Clinica.class.getName()).log(Level.SEVERE, null, ex);
    }
}
    
```

A continuación podemos ver unas imágenes del programa en funcionamiento.







Y con esto damos por finalizada la tarea.