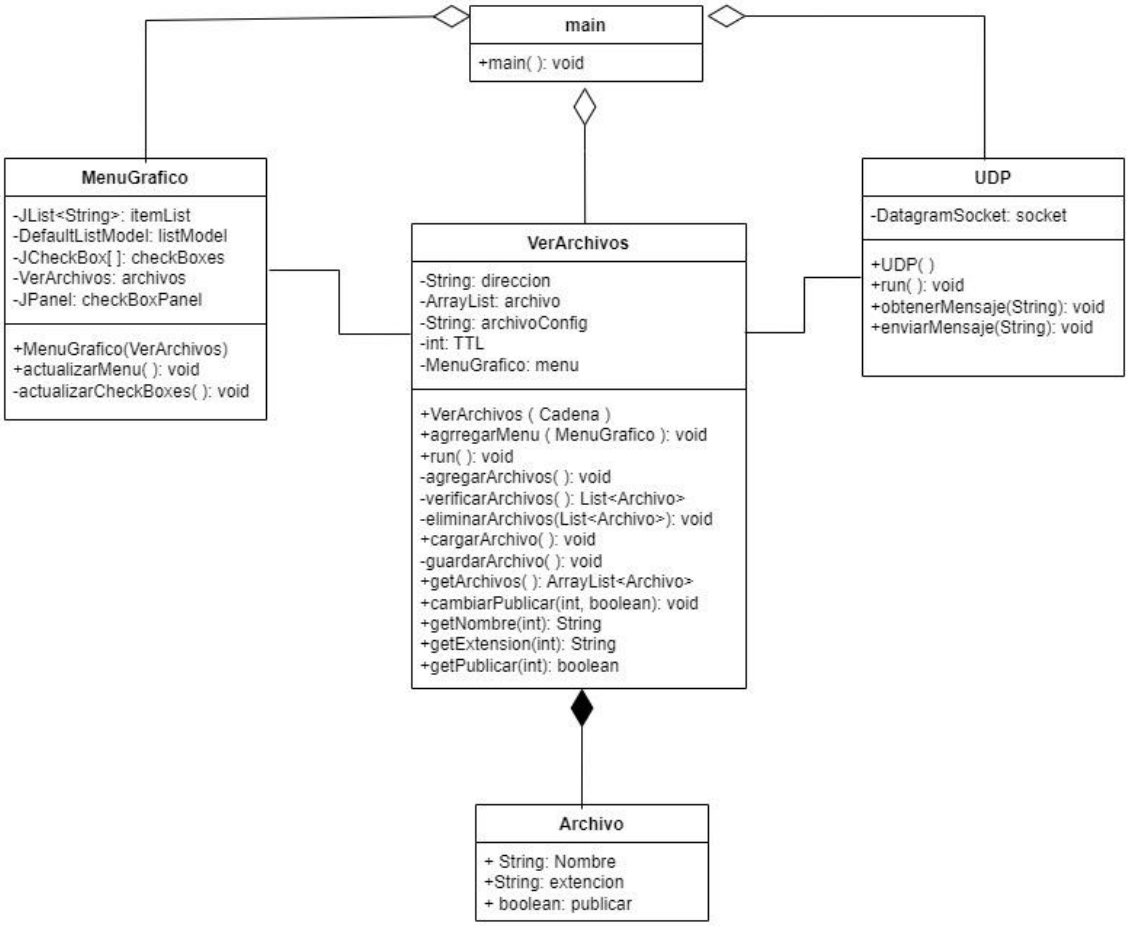
Servidor de nombres

Irvyn xicale cabrera - 201963582

Diagrama



Código fuente

Archivo.java

```
package DNS;

class Archivo {
    String nombre;
    String extension;
    boolean publicar;

    public Archivo(String nombre, String extension, boolean publicar) {
        this.nombre = nombre;
        this.extension = extension;
```

```
        this.publicar = publicar;
    }
}
```

VerArchivos.java

```
package DNS;

import java.io.*;
import java.util.*;
//import java.util.stream.Collectors;

import javax.swing.SwingUtilities;

class VerArchivos extends Thread {
    private String direccion = ""; // direccion donde revisaremos los
archivos
    private ArrayList<Archivo> archivo = new ArrayList<>(); // un array
donde guargamos cada archivo
    private String archivoConfig = System.getProperty("user.dir") +
"\\DNS\\config.inf"; // archivo config donde guardamos el TTL, ruda de la
carpeta y los archivos
    private int TTL = 5000; // tiempo para revisar la carpeta
    private MenuGrafico menu; // menu grafico

    public VerArchivos(String direccion) {
        this.direccion = direccion;
        this.archivo = new ArrayList<>();
        cargarArchivo(); // Cargar la lista de archivos
    }

    public void agregarMenu(MenuGrafico m){
        menu = m; // llamamos el menu
    }

    @Override
    public void run() {
        while (true) {
            try {
                Thread.sleep(TTL); // Espera el tiempo antes de la siguiente
actualización
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
```

```
            System.out.println("actualizando ...");
            agregarArchivos(); // metodo para agregar archivos nuevos
            List<Archivo> archivosAEliminar = verificarArchivos(); //
obtenemos una lista de los archivos a eliminar
            eliminarArchivos(archivosAEliminar); // eliminamos los archivos
de la lista
            guardarArchivo(); // actualizamos el config
        }
    }

    private void agregarArchivos() {
        File folder = new File(direccion); // cargamos la direccion de la
carpeta
        File[] files = folder.listFiles(); // cargamos los archivos

        for (File file : files) {
            if (file.isFile()) {
                String nombreCompleto = file.getName(); // obtenemos el
nombre completo del archivo
                String extension =
nombreCompleto.substring(nombreCompleto.lastIndexOf(".") + 1); // obtiene la
extencion
                String nombre = nombreCompleto.substring(0,
nombreCompleto.lastIndexOf(".")); // obtenemos el puro nombre sin extencion
                boolean publicar = false; // decidi que todos los archivos
recien ingresados no estara publicados
                boolean existeEnLista = archivo.stream().anyMatch(f ->
f.nombre.equals(nombre)); // con este metodo decidimos si el archivo ya
existe

                if (!existeEnLista) { // en caso de no existir lo agregamos
                    archivo.add(new Archivo(nombre, extension, publicar));
// agregamos al objeto Archivo
                    System.out.println("agregando " + nombre + " " +
extension + " " + publicar);
                }
            }
        }
    }

    private List<Archivo> verificarArchivos() {
        List<Archivo> archivosAEliminar = new ArrayList<>();// creamos el
arreglo

        for (Archivo archivoEnLista : archivo) {
```

```
            File archivoActual = new File(direccion + "/" +
archivoEnLista.nombre + "." + archivoEnLista.extension); //ponemos la
direccion completa del archivo
            if (!archivoActual.exists()) {// si no existe en la direccion
entonces se elimino y tenemos que eliminarlo
                System.out.println("El archivo " + archivoEnLista.nombre +
"." + archivoEnLista.extension + " no está presente y será eliminado.");
                archivosAEliminar.add(archivoEnLista);// agregamos el
archivo a la lista de eliminar
            }
        }

        return archivosAEliminar; // devolvemos los archivos a elminar
    }

    private void eliminarArchivos(List<Archivo> archivosAEliminar) {
        archivo.removeAll(archivosAEliminar); // eliminamos los archivos en
la lista
        SwingUtilities.invokeLater(() -> menu.actualizarMenu());//
actualizamos el menu
    }

    public void cargarArchivo() {
        try (BufferedReader reader = new BufferedReader(new
FileReader(archivoConfig))) {
            String line;
            int lineCount = 0; // Variable para llevar el conteo de líneas

            while ((line = reader.readLine()) != null) {
                lineCount++;// aumentamos el contador

                if (lineCount == 1) {// la primera linea es el TTL
                    TTL = Integer.parseInt(line); // guardamos el TTL
                    System.out.println("TTL: " + TTL);
                }if (lineCount == 2) {// la linea 2 es la direccion de la
carpeta compartida
                    direccion = line;
                    System.out.println("direccion: " + direccion);
                }if (lineCount == 3) {// la linea 3 es la direccion del
config
                    archivoConfig = line;
                    System.out.println("config: " + archivoConfig);
                }else{// despues de la 3 linea son los archivos previamente
guardados tiene una estructura como la siguiente
```

```
                    //nombre_archivo,extencion,(un boolean para saber si se
comparte o no)
                    String[] parts = line.split(",");// obtenemos el
nombre,extencion y el boolean
                    if (parts.length >= 3) {
                        archivo.add(new Archivo(parts[0], parts[1],
Boolean.parseBoolean(parts[2])));// agregamos a la lista de archivos
                        System.out.println("agregando " + parts[0] + " " +
parts[1] + " " + parts[2]);
                    }
                    for (Archivo archivo2 : archivo) {
                        System.out.println("lista ------> " +
archivo2.nombre + "." + archivo2.extension + " - " + archivo2.publicar);
                    }
                }

            }
        } catch (FileNotFoundException e) {
            System.err.println("El archivo de registro no existe. Se creará
uno nuevo.");// en caso de no existir el archivo mandamos el mensaje
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void guardarArchivo() {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(archivoConfig))) {// accedemos al config
            writer.write(Integer.toString(TTL));// guardamos el TTL
            writer.newLine();
            writer.write(direccion);// guardamos la direccion de la carpeta
            writer.newLine();
            writer.write(archivoConfig);// guardamos la direccion del
archivo config
            writer.newLine();
            for (Archivo archivo : archivo) {// guardamos todos loa archivos
en la lista de archivos
                writer.write(archivo.nombre + "," + archivo.extension + ","
+ archivo.publicar);
                writer.newLine();
            }
            SwingUtilities.invokeLater(() -> menu.actualizarMenu()); //
actualizamos el menu grafico
        } catch (IOException e) {
            e.printStackTrace();
```

```
        }
    }

    public ArrayList<Archivo> getArchivos() {// este metodo nos servira en
el menu grafico
        return archivo;
    }

    public void cambiarPublicar(int index, boolean cambio){// cambiamos el
boolean publicar(en el objeto de MenuGrafico)
        archivo.get(index).publicar = cambio;
    }

    public String getNombre(int index){// obtenemos el nombre
        return archivo.get(index).nombre;
    }

    public String getExtension(int index){// obtenemos la extensión
        return archivo.get(index).extension;
    }

    public boolean getPublicar(int index){// obtenemos el boleano de
publicar
        return archivo.get(index).publicar;
    }
}
```

MenuGrafico.java

```
package DNS;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ItemEvent;

public class MenuGrafico extends JFrame {
    private JList<String> itemList;
    private DefaultListModel<String> listModel;
    private JCheckBox[] checkBoxes;
    private VerArchivos archivos;
    private JPanel checkBoxPanel;

    public MenuGrafico(VerArchivos archivos) {
```

```
        this.archivos = archivos;// agregamos el archivo donde tenemos la
lista de archivos
        //agregamos atributos principales de nuestro menu
        setTitle("Menú Gráfico");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(500, 700);
        setLocationRelativeTo(null);

        listModel = new DefaultListModel<>();// agregamos el nombre de los
archivos que tenemos
        for (int i = 0; i < archivos.getArchivos().size(); i++) {
            listModel.addElement(archivos.getNombre(i) + "." +
archivos.getExtension(i));
        }

        itemList = new JList<>(listModel); // los agregamos

        itemList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        itemList.addListSelectionListener(e -> actualizarCheckBoxes());

        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout());
        panel.add(new JScrollPane(itemList), BorderLayout.CENTER);// lo
agregamos a nuestro panel

        checkBoxPanel = new JPanel(); // Inicializamos checkBoxPanel
        checkBoxPanel.setLayout(new BoxLayout(checkBoxPanel,
BoxLayout.PAGE_AXIS));

        checkBoxes = new JCheckBox[listModel.getSize()];
        for (int i = 0; i < listModel.getSize(); i++) {
            // agregamos los checkbox con el atributu de publicar
            checkBoxes[i] = new JCheckBox();
            checkBoxes[i].setSelected(archivos.getPublicar(i));
            final int index = i;

            checkBoxes[i].addItemListener(e -> {// agregamos un evento al
checbox
                if (e.getStateChange() == ItemEvent.SELECTED) {// si el
checbox esta seleccionado
                    archivos.cambiarPublicar(index, true);// cambiamos el
estado
                } else if (e.getStateChange() == ItemEvent.DESELECTED) {//
en caso contrario
```

```
                    archivos.cambiarPublicar(index, false);// cambiamos el
estado
                }
                System.out.println("Cambio realizado para " +
archivos.getNombre(index) + "." + archivos.getExtension(index));
            });

            checkBoxPanel.add(checkBoxes[i]);// agregamos el checkbox al
panel
        }

        panel.add(checkBoxPanel, BorderLayout.EAST);// agregamos los
checkbox al panel

        add(panel); // agregamos el panel
    }

    public void actualizarMenu() {
        listModel.clear();// limpiamos el la lista
        for (int i = 0; i < archivos.getArchivos().size(); i++) {
            listModel.addElement(archivos.getNombre(i) + "." +
archivos.getExtension(i));// volvemos a agregar los archivos
        }
        actualizarCheckBoxes(); // Actualiza los checkboxes cuando se
actualiza la lista de elementos
    }

    private void actualizarCheckBoxes() {
        for (JCheckBox checkBox : checkBoxes) {
            checkBoxPanel.remove(checkBox); // Elimina los checkboxes
existentes
        }

        checkBoxes = new JCheckBox[listModel.getSize()];
        for (int i = 0; i < listModel.getSize(); i++) { // agregamos los
checkboxes
            checkBoxes[i] = new JCheckBox();
            checkBoxes[i].setSelected(archivos.getPublicar(i));
            final int index = i;

            checkBoxes[i].addItemListener(e -> {
                if (e.getStateChange() == ItemEvent.SELECTED) {
                    archivos.cambiarPublicar(index, true);
                } else if (e.getStateChange() == ItemEvent.DESELECTED) {
                    archivos.cambiarPublicar(index, false);
```

```
                }
                System.out.println("Cambio realizado para " +
archivos.getNombre(index) + "." + archivos.getExtension(index));
            });

            checkBoxPanel.add(checkBoxes[i]); // Agrega los nuevos
checkboxes
        }

        checkBoxPanel.revalidate(); // Actualiza la disposición del panel
        checkBoxPanel.repaint(); // Repinta el panel
    }
}
```

UDP.java

```
package DNS;
import java.io.IOException;
import java.net.*;

public class UDP extends Thread{
    private DatagramSocket socket;

    public UDP() {
        try {
            socket = new DatagramSocket(50000); // agregamos el puerto
        } catch (SocketException e) {
            e.printStackTrace();
        }
    }

    public void run() {
        try {
            while (true) {
                // recibimos el mensaje en el servidor
                byte[] receiveData = new byte[1024];
                DatagramPacket receivePacket = new
DatagramPacket(receiveData, receiveData.length);
                socket.receive(receivePacket);

                String message = new String(receivePacket.getData(), 0,
receivePacket.getLength());
                obtenerMesaje(message);// enviamos el mensaje para procesar
```

```
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void obtenerMesaje(String message) {// en este metodo
procesaremos el mensaje
        System.out.println("Mensaje recibido en el servidor: " + message);
    }

    public void enviarMensaje(String message, InetAddress
destinationAddress, int destinationPort) {// metodo para enviar mensaje
        try {
            byte[] sendData = message.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, destinationAddress, destinationPort);
            socket.send(sendPacket);// enviamos el mensaje
            System.out.println("Mensaje enviado desde el cliente: " +
message);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Main.java

```
package DNS;
// librerias para ser cliente
/*
import java.net.InetAddress;
import java.net.UnknownHostException;
*/
import javax.swing.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class main {
    public static void main(String[] args){
        UDP servidor = new UDP(); // creamos el objeto UDP
```

```
        String direccion = "";// iniciamos la direcion de la carpeta
        boolean primerInicio = true;// controlamos si es el primer inicio


        try (BufferedReader reader = new BufferedReader(new
FileReader(System.getProperty("user.dir") + "\\DNS\\config.inf"))) {
            // si entramos aqui entonces el archivo config existe y no es el
primer inicio del programa
            primerInicio = false;// no el el primer inicio
            String line;
            int lineCount = 0; // Variable para llevar el conteo de líneas

            while ((line = reader.readLine()) != null || lineCount < 3) {
                lineCount++;
                if (lineCount == 2) {
                    direccion = line;// leemos la dirección de la carpeta
                    System.out.println("direccion previa: " + direccion);
                }
            }
        } catch (FileNotFoundException e) {
            System.err.println("primer inicio del programa");
            primerInicio = true;// es el primer inicio del programa
        } catch (IOException e) {
            e.printStackTrace();
        }

        if (primerInicio == true) {
            SwingUtilities.invokeLater(() -> {// creamos un menu para
obtener la direccion de la carpeta compartida
                JFileChooser fileChooser = new JFileChooser();
                fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ON
LY);

                int result = fileChooser.showOpenDialog(null);

                if (result == JFileChooser.APPROVE_OPTION) { // cuando
tenemos la direccion
                    File selectedDirectory = fileChooser.getSelectedFile();
                    VerArchivos verArchivos = new
VerArchivos(selectedDirectory.getAbsolutePath()); // creamos el objeto de
verArchivos con la direccion que el usuario nos dio
                    verArchivos.start();
                    SwingUtilities.invokeLater(() -> {// creamos el menu
grafico
```

```
                        MenuGrafico menu = new MenuGrafico(verArchivos);//
creamo y pasamos el objeto de verArchivos al menu
                        verArchivos.agregarMenu(menu);// vinculamos el menu
al objeto de verArchivos
                        menu.setVisible(true);// mostramos el menu
                    });
                }
            });
        }else{// no es el primer inicio del programa
            VerArchivos verArchivos = new VerArchivos(direccion);// creamos
el objeto verArchivos
            verArchivos.start();
            SwingUtilities.invokeLater(() -> {
                MenuGrafico menu = new MenuGrafico(verArchivos);// creamo y
pasamos el objeto de verArchivos al menu
                verArchivos.agregarMenu(menu);// vinculamos el menu
                menu.setVisible(true);// mostrmos el menu
            });
        }


        servidor.start();

    }
}
```