

# AD

# DOCUMENTACIÓN RESTAURANTE

ALUMNO: GAIZKA SECO FONSEKA

---

## INICIO DE LA APLICACIÓN

Antes de iniciar la aplicación, la primera vez que abres el proyecto hay que ejecutar los ficheros CrearFichEmpleado, CrearFichClientes, CrearFichPlatos, CrearFichLogin y CrearFichProductos. Después ya puedes iniciar la aplicación a través de la clase main.

## ESTRUCTURA DE LA APLICACIÓN

Esta es una aplicación de restaurante dirigida a los empleados que puedan trabajar allí ya que en ella se podrán gestionar empleados, clientes, platos... incluso generar un menú aleatorio.


Al iniciar la aplicación veremos que se inicia una ventana de login, al ser una aplicación para los empleados cada uno tendrá su propio usuario y contraseña que podrá usar para entrar, en este caso solo tendremos una que será la de admin(admin / 12345Abcde). Obviamente si usas una cuenta inexistente no se podrá acceder a la aplicación.



Después de iniciar sesión la aplicación mostrará un menú principal con el cual podremos gestionar datos del restaurante. Todas las ventanas siguientes tendrán las mismas funciones y estructura.

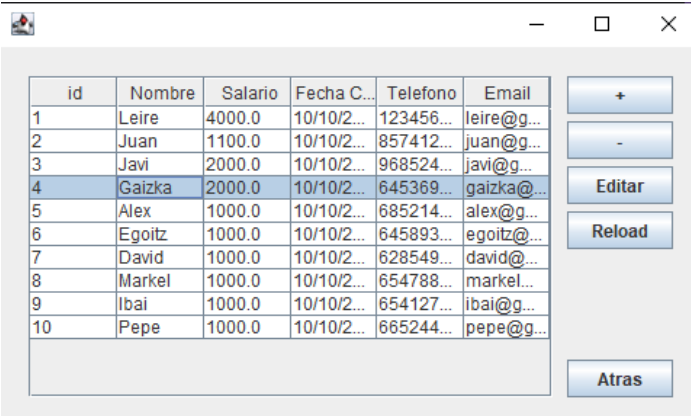


Mostraré de ejemplo la interfaz gráfica de los empleados aunque el uso es el mismo para todas las ventanas de gestión.



id	Nombre	Salario	Fecha C...	Telefono	Email
1	Leire	4000.0	10/10/2...	123456...	leire@g...
2	Juan	1100.0	10/10/2...	857412...	juan@g...
3	Javi	2000.0	10/10/2...	968524...	javi@g...
4	Gaizka	2000.0	10/10/2...	645369...	gaizka@...
5	Alex	1000.0	10/10/2...	685214...	alex@g...
6	Egoitz	1000.0	10/10/2...	645893...	egoitz@...
7	David	1000.0	10/10/2...	628549...	david@...
8	Markel	1000.0	10/10/2...	654788...	markel...
9	Ibai	1000.0	10/10/2...	654127...	ibai@g...
10	Pepe	1000.0	10/10/2...	665244...	pepe@g...

Lo más destacable es que se podrán ver todos los datos de la ventana seleccionada con una tabla, la tabla también nos servirá para eliminar cualquier dato que queramos, para hacerlo con seleccionar clicando en la tabla el dato y dándole al botón de - se borrará automáticamente.



id	Nombre	Salario	Fecha C...	Telefono	Email
1	Leire	4000.0	10/10/2...	123456...	leire@g...
2	Juan	1100.0	10/10/2...	857412...	juan@g...
3	Javi	2000.0	10/10/2...	968524...	javi@g...
4	Gaizka	2000.0	10/10/2...	645369...	gaizka@...
5	Alex	1000.0	10/10/2...	685214...	alex@g...
6	Egoitz	1000.0	10/10/2...	645893...	egoitz@...
7	David	1000.0	10/10/2...	628549...	david@...
8	Markel	1000.0	10/10/2...	654788...	markel...
9	Ibai	1000.0	10/10/2...	654127...	ibai@g...
10	Pepe	1000.0	10/10/2...	665244...	pepe@g...

De la misma manera si queremos editar un dato se hace de la misma manera que antes pero en vez de darle al botón de eliminar se le tendrá que dar al de editar y en este caso se abrirá una ventana con los datos cargados para poder editarlos.

---



**EDITAR EMPLEADO**

Nombre: Gaizka

Salario: 2000.0

Fecha Contrato: 10/10/2020

Telefono: 645369874

Email: gaizka@gmail.com

Editar

Atras

Dependiendo del dato que se quiera eliminar aparecerán diferentes campos rellenos, en el de arriba se puede ver un ejemplo de los empleados pero este sería el ejemplo de los clientes.



**EDITAR CLIENTE**

Nombre: Mario

Telefono: 654987321

Email: mario@gmail.com

Editar

Atras

Por último tendremos el botón de añadir que mostrará una ventana con los mismos campos de editar pero en este caso vacíos para poder añadir un dato nuevo. En ambos casos está controlado que no se puedan añadir o editar los datos si hay campos vacíos o si hay datos que deberían ser concretos como el número de teléfono.

---

También existe el botón de reload que su función es que si en algún momento la tabla se debería de haber actualizado y no lo ha hecho, ese botón se encargará de cargar los datos que hay almacenados en ese momento.

Volviendo a la ventana principal todavía quedan 2 botones por explicar:



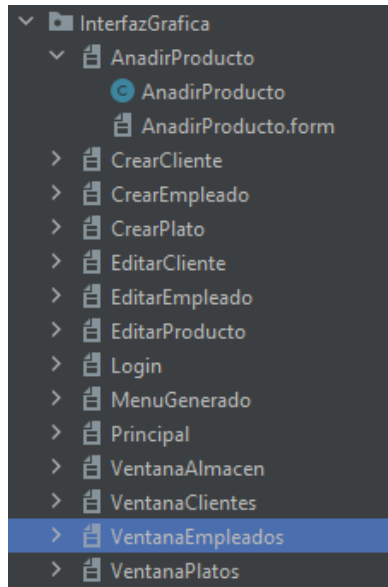
- Por una parte tenemos el botón de exportar todos los datos, el cual exporta todos los datos que tenga en ese momento los archivos .dat de empleados, clientes, platos y almacén y los crea en el archivo XML con su estructura adecuada.
- Por último tenemos el botón de generar menú, este botón abre una ventana en la que se ha generado un menú con un primero, segundo y postre, claro que los platos son aleatorios pero dentro de su categoría ya que al crearlos se les selecciona la categoría de primeros, segundos o postres.

## ESTRUCTURA DE CÓDIGO

El proyecto está dividido en dos partes mayores, una de ellas la parte gráfica y la otra las clases de utilidad.

En la parte gráfica, voy a explicar de manera general el funcionamiento:

- Tenemos un form y una clase por ventana que necesitamos, en ellas se crea la interfaz gráfica y lo que va a hacer cada botón.



- Todos los botones de añadir tienen la misma intención y funcionan de la misma manera. Como se puede ver su función es llamar a otra ventana que servirá para añadir el usuario.

```
anadirBoton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        cargarDatos();  
        JFrame frame = new CrearEmpleado(datos);  
        frame.setSize( width: 500, height: 300);  
        frame.setVisible(true);  
        dispose();  
    }  
});
```

- También está el botón de eliminar el cual recoge el id del dato que esté marcado en la tabla y se lo manda a una función que será la encargada de eliminarlo. También se encarga de que tengamos un dato seleccionado para que no se cometa ningún error.

```
eliminarBoton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        if (table1.getSelectedRow() == -1) {
            JOptionPane.showMessageDialog( parentComponent: null, message: "Para eliminar debes seleccionar en la tabla.");
        } else {
            int id = Integer.parseInt(table1.getValueAt(table1.getSelectedRow(), column: 0).toString());
            eliminarEmpleado(id);
        }
    }
});
```

- Tenemos el botón de reload el cual llama a la función de cargar la tabla que se encarga de volver a leer los datos que hay en los archivos y añadirlos a la tabla.

```
reloadBoton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { modificarTabla(); }
});
```

- También está el botón de atrás que nos manda a la ventana anterior en la que estuvimos.

```
atrasBoton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JFrame frame = new Principal();
        frame.setSize( width: 500, height: 300);
        frame.setVisible(true);
        dispose();
    }
});
```

- Luego tenemos las ventanas de añadir que estas cuentan con los datos a rellenar y un botón que se encarga de comprobar que los datos estén bien introducidos o que sean nulos y si todo está bien añade el dato al archivo. La manera de añadirlo está en una función que carga los datos a una lista, se le añade el nuevo objeto y sobrescribe el archivo.

```
anadirBoton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { anadirProducto(); }
});
```

```

public void anadirProducto() {
    try {
        String producto = productField.getText();
        int cantidad = Integer.parseInt(cantidadField.getText());

        //Se comprueba que no tengamos datos vacios
        if (producto.trim().equals("")) {
            JOptionPane.showMessageDialog( parentComponent: null, message: "Compruebe que los datos son correctos");
        } else {
            int id;
            //Se comprueba si hay datos en el array o no, ya que si el dat esta vacio podria dar problemas
            if (datos.size() == 0) {
                id = 1;
            } else {
                id = datos.get(datos.size() - 1).getId() + 1;
            }
            datos.add(new Producto(id, producto, cantidad));

            //Se crea flujos de entrada
            File file = new File( pathname: "Productos.dat");
            FileOutputStream fileo = new FileOutputStream(file);
            ObjectOutputStream fileobj = new ObjectOutputStream(fileo);

            //Escribimos los objetos producto
            for (Producto dato : datos) {
                fileobj.writeObject(dato);
            }

            //Mostramos mensaje de que se ha creado correctamente y volvemos a la venta de almacen
            JOptionPane.showMessageDialog( parentComponent: null, message: "El producto se ha añadido correctamente.");

            fileobj.close();
            JFrame frame = new VentanaAlmacen();
            frame.setSize( width: 500, height: 300);
            frame.setVisible(true);
            dispose();
        }
    }
}

```

- También tenemos la ventanas de editar, esta ventana tiene el mismo funcionamiento de la anterior al darle al botón comprueba los datos, si todo está correcto procede a editar el dato, se cargan los datos a la lista se encuentra el objeto que tenga esos datos se elimina y se añade el nuevo, nuevamente carga los datos nuevos en el archivo. En ambas ventanas si los datos no están bien introducidos saltará un mensaje de error y no se hará ningún cambio.

```

public EditarEmpleado(int id, List<Empleado> datos) {
    //Mostramos el panel de la interfaz
    setContentPane(PanelEditarEmpleados);
    //Cargamos los datos del empleado que se va a editar en los textFields
    for (Empleado dato : datos) {
        if (dato.getId() == id) {
            nombreField.setText(dato.getNombre());
            salarioField.setText(dato.getSalario().toString());
            fechaField.setText(dato.getFechaCon());
            telefonoField.setText(String.valueOf(dato.getTelefono()));
            emailField.setText(dato.getEmail());
        }
    }
}

```

Aquí recogeremos la lista que nos han enviado cargada y colocaremos los datos en los campos de la interfaz gráfica.



```

editarBoton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            String nombre = nombreField.getText();
            Double salario = Double.parseDouble(salarioField.getText());
            String fecha = fechaField.getText();
            int telefono = Integer.parseInt(telefonoField.getText());
            String email = emailField.getText();

            //Se comprueba que no hay datos vacios
            if (nombre.trim().equals("") || fecha.trim().equals("") || email.trim().equals("") || String.valueOf(telefono).length() != 9){
                JOptionPane.showMessageDialog( parentComponent: null, message: "Compruebe que los datos son correctos");
            } else{
                for (Empleado dato : datos) {
                    if (dato.getId() == id) {
                        dato.setNombre(nombre);
                        dato.setSalario(salario);
                        dato.setFechaCon(fecha);
                        dato.setTelefono(telefono);
                        dato.setEmail(email);
                    }
                }

                //Se crea el flujo de entrada
                File file = new File( pathname: "Empleados.dat");
                FileOutputStream fileo = new FileOutputStream(file);
                ObjectOutputStream fileobj = new ObjectOutputStream(fileo);

                //Se escribe cada objeto en el dat
                for (Empleado dato : datos) {
                    fileobj.writeObject(dato);
                }

                fileobj.close();
            }
        } catch (Exception ex) {
            JOptionPane.showMessageDialog( parentComponent: null, message: "Error al guardar los datos");
        }
    }
});

```

Y en este se puede ver cómo se comprueban los datos y si todo está correcto se añade a la lista y se escriben los datos al archivo.

- En este apartado también está el menú aleatorio. Para generar este menú todo sucede al darle al boton del menu principal y cargar la ventana, se cargan los platos 3 diferentes listas, cada plato irá a una lista dependiendo de si es primero, segundo o postre, después con un random se eligiera uno de cada lista y se añadirá a una lista de menu aleatorio y se cargan los datos en los campos para que el usuario los pueda ver.

---

```

public void generarMenu() {
    //Seleccionamos los platos por su categoria si es primero, segundo o postre
    Random r = new Random();
    cargarDatos();
    if (datos.size() != 0) {
        List<Plato> primeros = new ArrayList<>();
        List<Plato> segundos = new ArrayList<>();
        List<Plato> postres = new ArrayList<>();
        for (Plato dato : datos) {
            if (dato.getCategoria() == 1) {
                primeros.add(dato);
            } else if (dato.getCategoria() == 2) {
                segundos.add(dato);
            } else if (dato.getCategoria() == 3) {
                postres.add(dato);
            }
        }

        //Elegimos uno de cada categoria
        int p = primeros.size();
        int s = segundos.size();
        int po = postres.size();
        //control por si no hay primeros generados
        if (primeros.size() == 0) {
            menu.add(new Plato( nombre: ""));
        } else if (p == 1) {
            menu.add(primeros.get(0));
        } else {
            menu.add(primeros.get(r.nextInt( origin: 0, p)));
        }

        //control por si no hay segundos generados
    }
}

```

- Este apartado del menú también tiene un botón el cual recoge los datos de la lista y los exporta usando DOM a un XML.

```

guardarBoton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //Export los datos del menu generado a XML con DOM
        try {
            //creamos el documento con DOM
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            DOMImplementation implementation = builder.getDOMImplementation();
            Document document = implementation.createDocument( namespaceURI: null, qualifiedName: "MenuAleatorio", doctype: null);
            document.setXmlVersion("1.0");

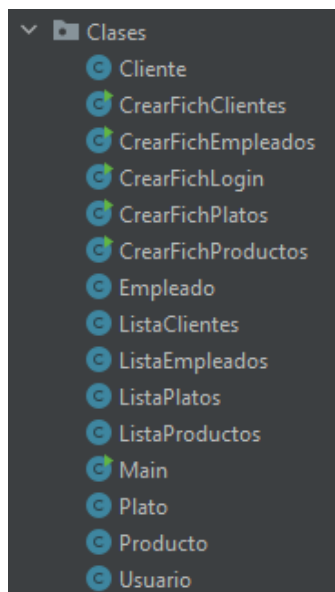
            //Creamos cada elemento con la funcion de crearElemento
            for (int i = 0; i < menu.size(); i++) {
                Element raiz = document.createElement( tagName: "plato");
                document.getDocumentElement().appendChild(raiz);
                crearElemento( datoPlato: "id", String.valueOf(menu.get(i).getId()), raiz, document);
                crearElemento( datoPlato: "nombre", menu.get(i).getNombre(), raiz, document);
                crearElemento( datoPlato: "descripcion", menu.get(i).getDescripcion(), raiz, document);
                crearElemento( datoPlato: "categoria", String.valueOf(menu.get(i).getCategoria()), raiz, document);
            }
        }
    }
});

```

Aquí se puede ver como en el botón se empiezan a crear los elementos que son necesarios para crear el XML con DOM.

- Por último el botón del menú usa XStream para exportar los archivos de empleados, platos, clientes y productos a XML.

En la parte de clases útiles:



- Tenemos clases por cada tipo de dato que vamos a guardar ya que he elegido que los datos van a ser guardados como objeto ya que me parece

---

más cómodo tratar con ellos. Estas clases son las siguientes: empleados, clientes, platos y productos (Ej: Empleados). Estas clases tienen variables dependiendo de qué datos queremos almacenar, el constructor y los getter/setter.

```
public class Empleado implements Serializable {
    3 usages
    int id;
    4 usages
    String nombre;
    3 usages
    double salario;
    3 usages
    String fechaCon;
    3 usages
    int telefono;
    3 usages
    String email;

    //Constructores

    public Empleado(int id, String nombre, Double salario, String fechaCon, int telefono, String email) {
        this.id = id;
        this.nombre = nombre;
        this.salario = salario;
        this.fechaCon = fechaCon;
        this.telefono = telefono;
        this.email = email;
    }
}
```

Una clase normal con sus variables, constructores...

- También tenemos unas clases que se llaman igual que las principales, es decir empleado, cliente, producto y platos pero estas nuevas clases tienen antes la palabra lista (Ej: ListaEmpleados), estas solo se usan el momento de exportar los datos por XStream ya que al hacerlo con una lista normal surgen errores. Estos solo tienen una lista creada, un constructor y un getter para poder acceder a ella.

```
public class ListaEmpleados {
    1 usage
    private List<Empleado> lista = new ArrayList<>();

    //Constructor
    1 usage
    public ListaEmpleados() {
    }

    public void add(Empleado emp) { lista.add(emp); }
}
```

- También están las clases que empiezan por CrearFich que sirven para crear un archivo dat inicial de los que se necesiten(Ej: CrearFichEmpleado crea el archivo empleados.dat con datos base ).

```
public class CrearFichEmpleados {
    public static void main(String[] args) {
        File file = new File( pathname: "Empleados.dat");
        try {
            //Creamos los flujos de entrada
            FileOutputStream fileo = new FileOutputStream(file);
            ObjectOutputStream fileobj = new ObjectOutputStream(fileo);

            //Creamos los array con los datos que queremos insertar
            String nombres[] = {"Leire", "Juan", "Javi", "Gaizka", "Alex", "Egoitz", "David", "Markel", "Ibai", "Pepe"};
            double salarios[] = {4000, 1100, 2000, 2000, 1000, 1000, 1000, 1000, 1000, 1000};
            String fechaCon[] = {"10/10/2020", "10/10/2020", "10/10/2020", "10/10/2020", "10/10/2020", "10/10/2020", "10/10/2020", "10/10/2020", "10/10/2020", "10/10/2020"};
            int telefono[] = {123456789, 857412563, 968524163, 645369874, 685214796, 645893214, 628549514, 654788932, 654127398, 665244615};
            String email[] = {"leire@gmail.com", "juan@gmail.com", "javi@gmail.com", "gaizka@gmail.com", "alex@gmail.com", "egoitz@gmail.com", "david@gmail.com", "markel@gmail.com", "ibai@gmail.com", "pepe@gmail.com"};

            //Insertamos los datos creando el objeto
            for (int i = 0; i < nombres.length; i++) {
                Empleado empleado = new Empleado( id: i + 1, nombres[i], salarios[i], fechaCon[i], telefono[i], email[i]);
                fileobj.writeObject(empleado);
            }

            fileobj.close();
            System.out.println("Se ha creado el DAT de los empleados.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

- Por último está la clase main que sirve para iniciar el programa principal que su función es llamar a la ventana de login.

```
public class Main {
    //Programa para iniciar la interfaz grafica
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                JFrame frame = new Login();
                frame.setSize( width: 500, height: 300);
                frame.setVisible(true);
            }
        });
    }
}
```