

# MERCADAW

---

## 1. ANÁLISIS

### REQUISITOS FUNCIONALES Y NO FUNCIONALES

#### FUNCIONALES

- Dar de alta un producto
- Dar de alta un empleado
- Visualizar los datos del producto
- Visualizar listado de empleados
- Dar de alta una compra
- Listar las compras
- Listar stock de los productos
- Obtener el precio de un producto
- Imprimir una etiqueta para clasificar el producto
- Calcular las nóminas de los trabajadores
- Exportar datos

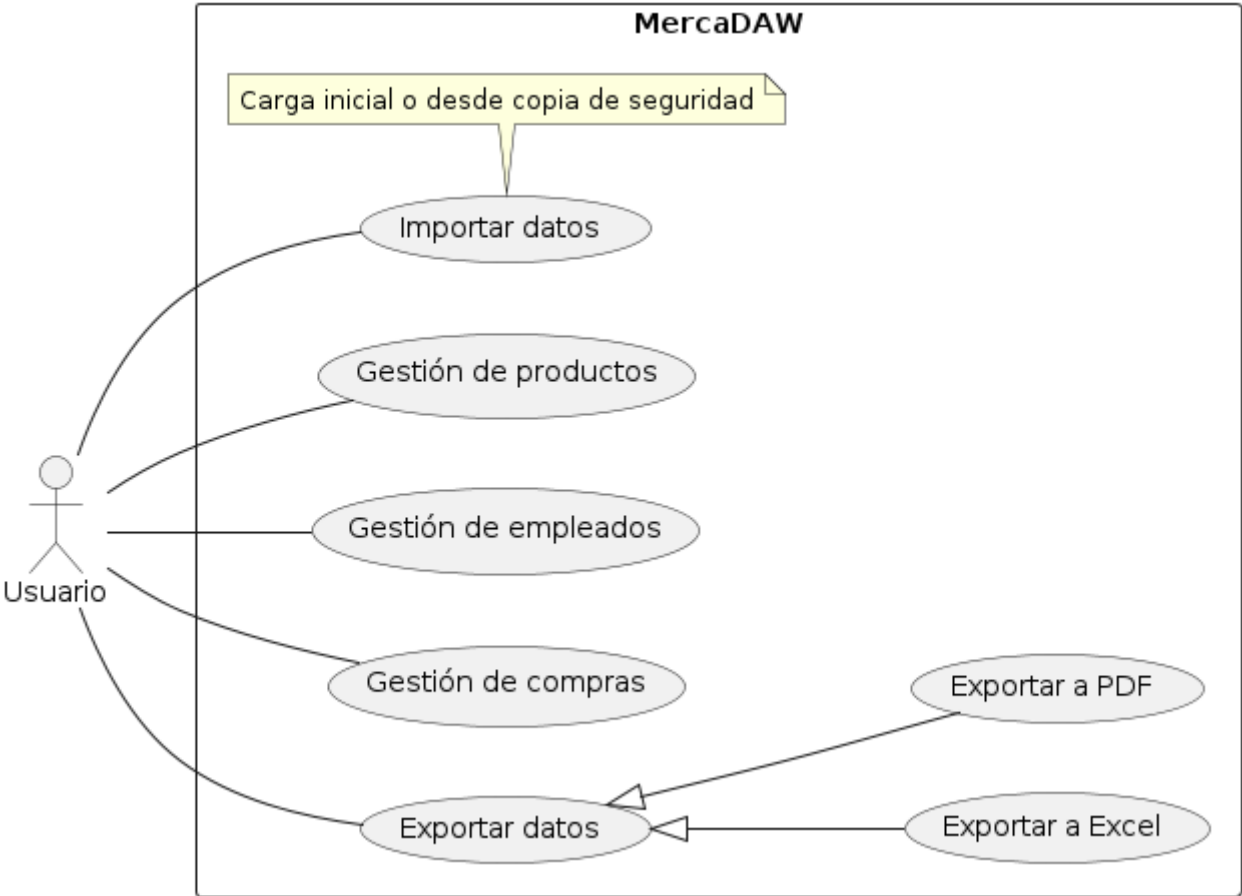
#### NO FUNCIONALES

- Se debe permitir una carga de los datos iniciales, a partir de un fichero, al iniciar la aplicación por primera vez o para recuperar una versión anterior a partir de una copia de seguridad.
- Los datos a cargar estarán en formato CVS, JSON o XML.
- El precio del producto se debe calcular en € y \$.
- El listado de compras debe ordenarse por código postal.
- Los datos se deben exportar a excel, pdf u otros formatos.
- Crear copias de seguridad.

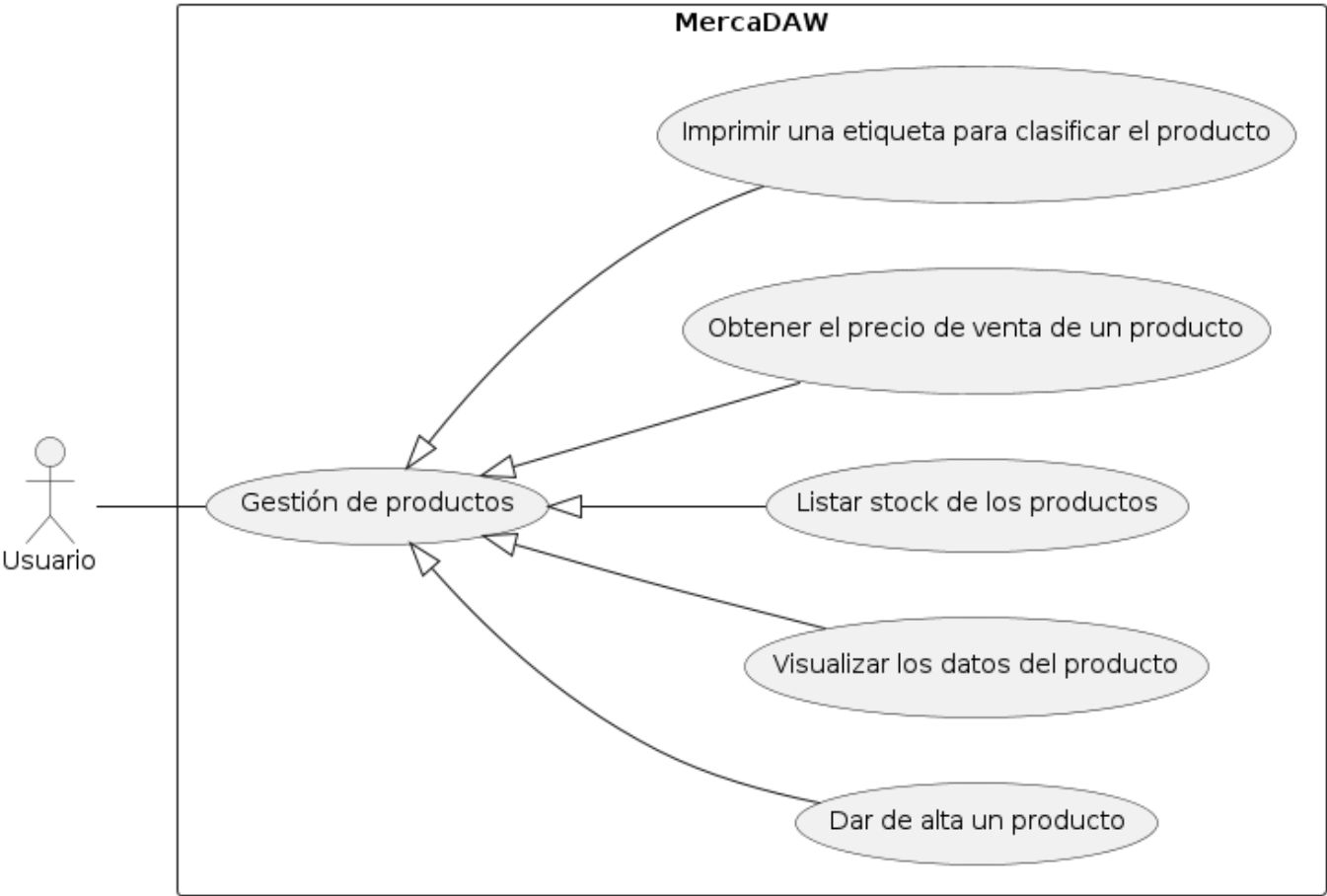
## 2. DISEÑO

### 2.1 DIAGRAMA DE CASOS DE USO

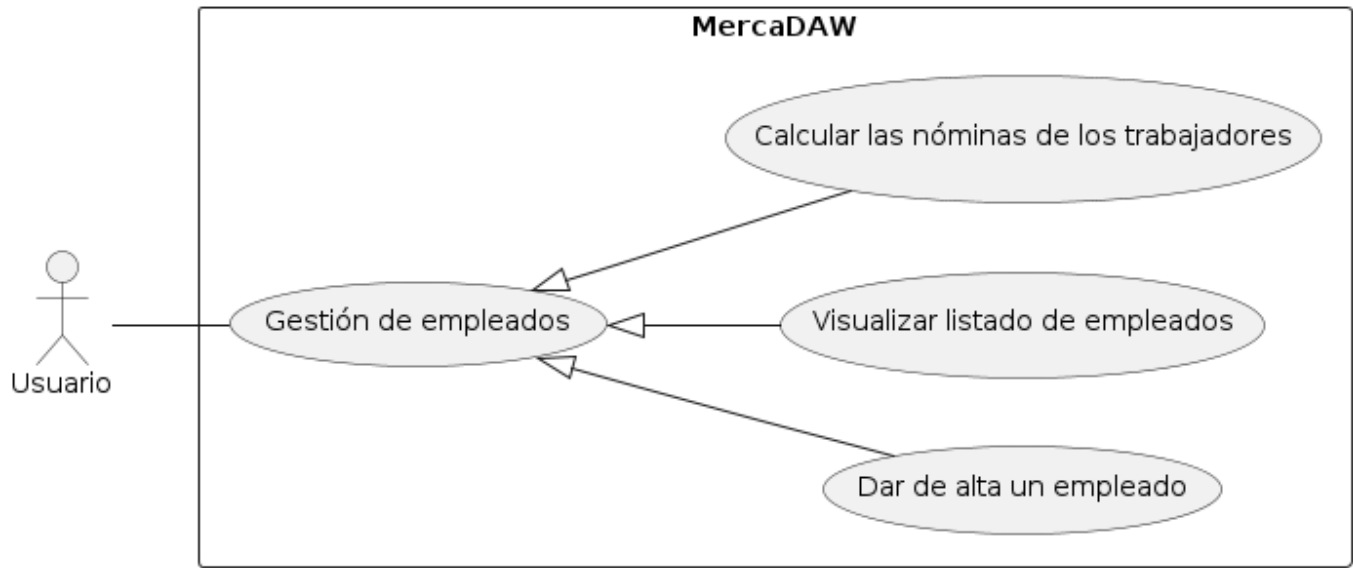
#### CASOS DE USO GENERAL



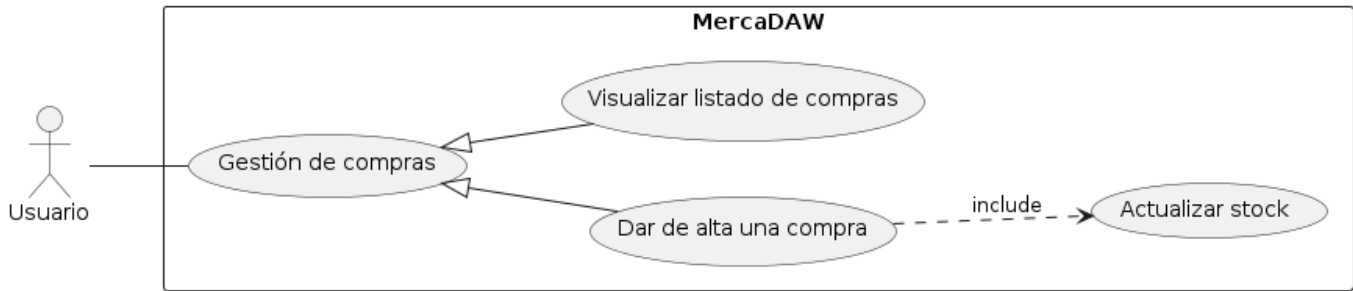
CASO DE USO PRODUCTO



CASO DE USO EMPLEADO

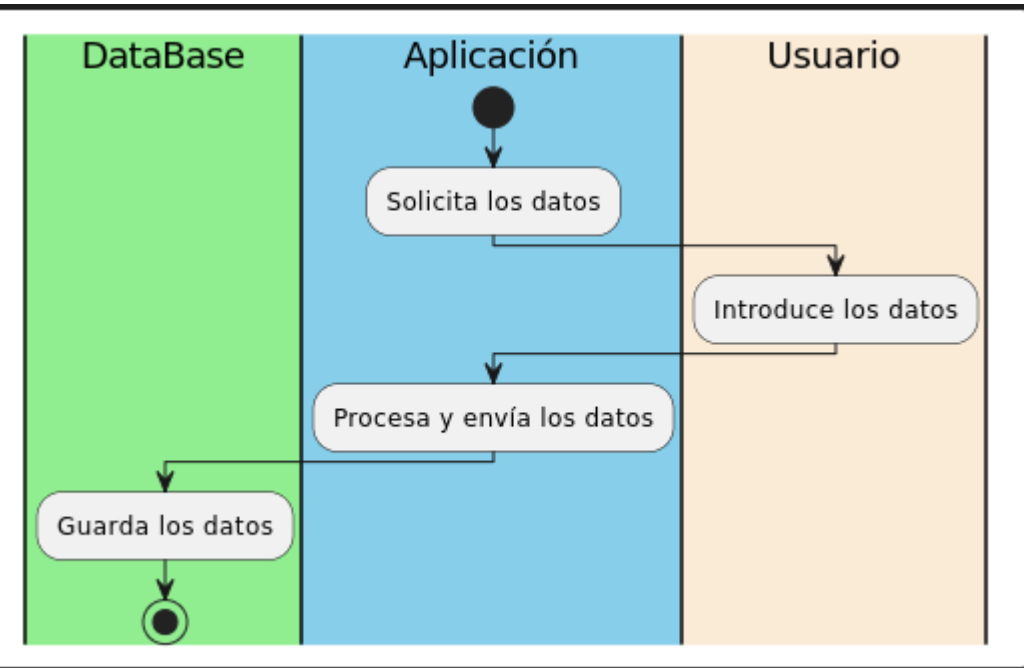


CASO DE USO COMPRA

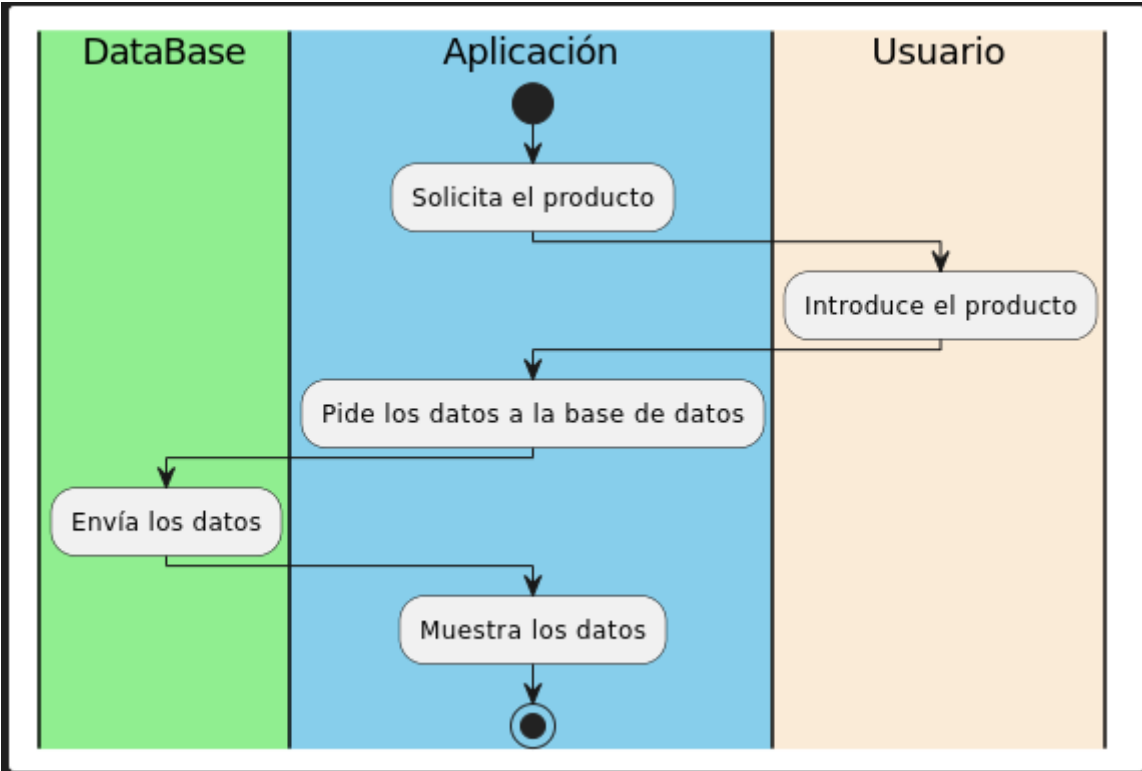


2.2 DIAGRAMAS DE ACTIVIDAD

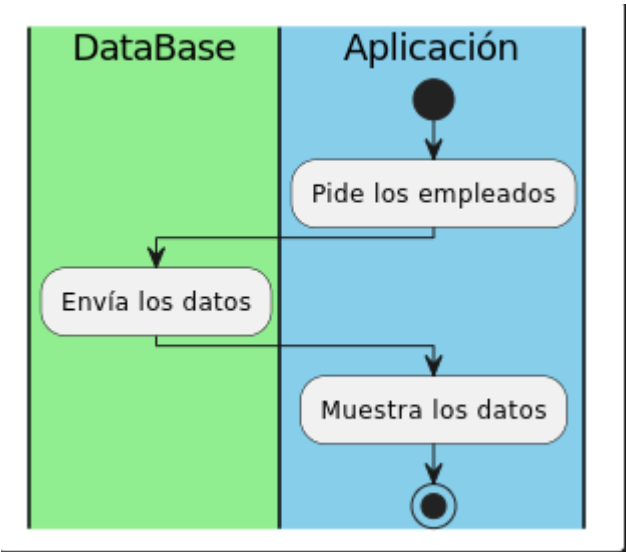
Dar de Alta



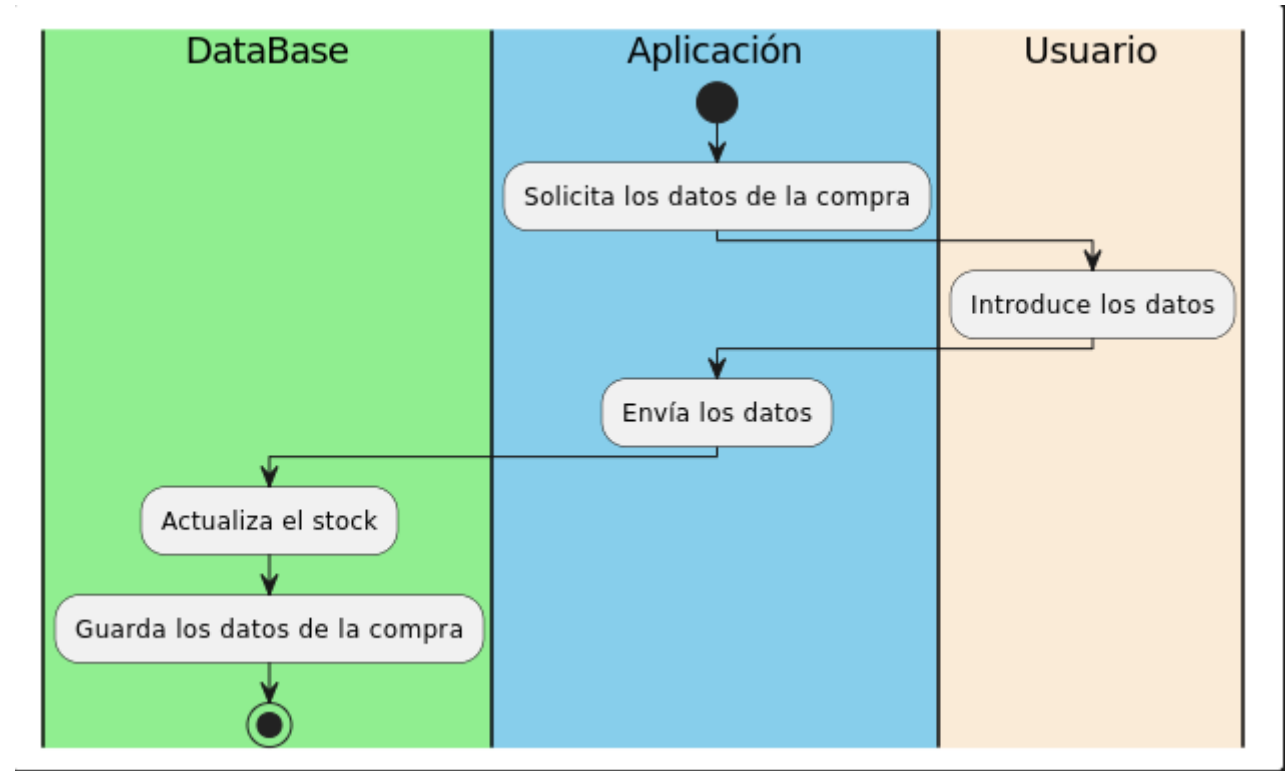
Visualizar Producto



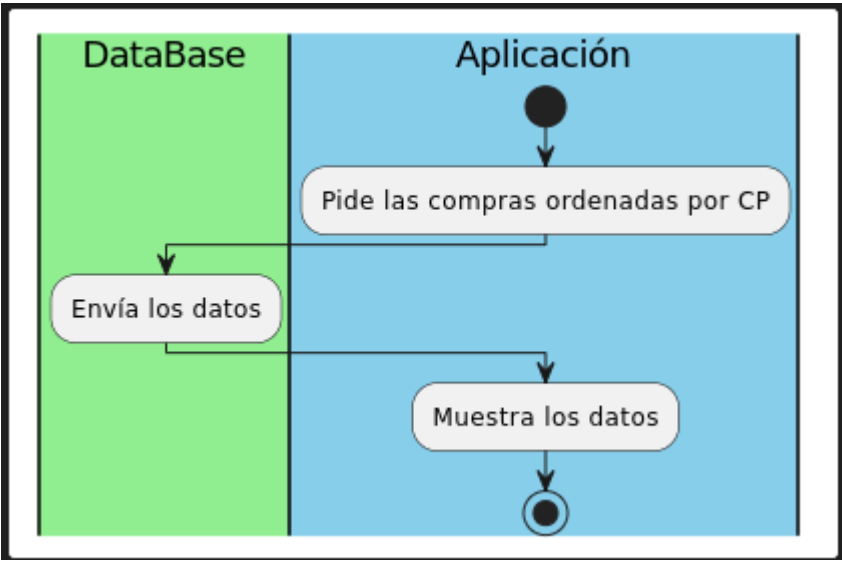
Visualizar Empleados



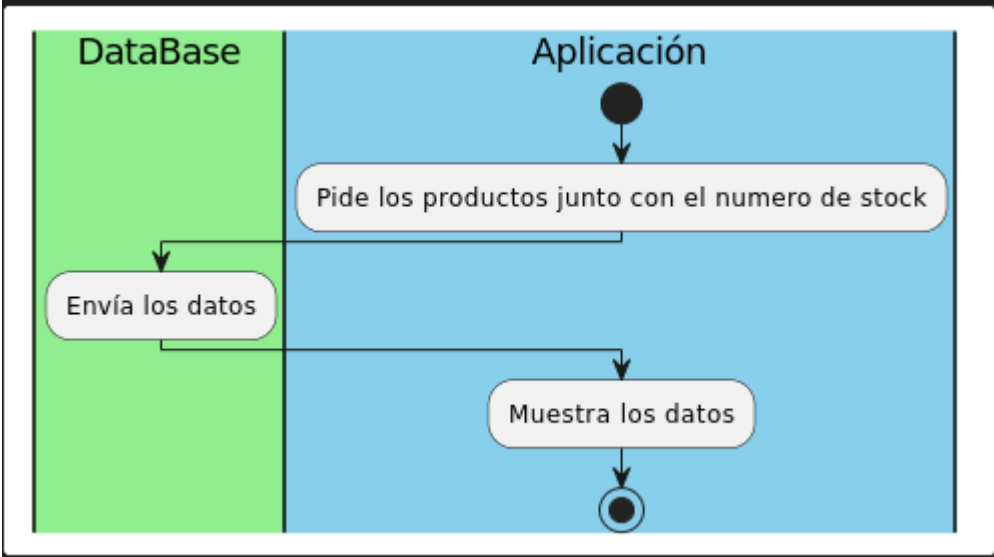
Alta de Compra



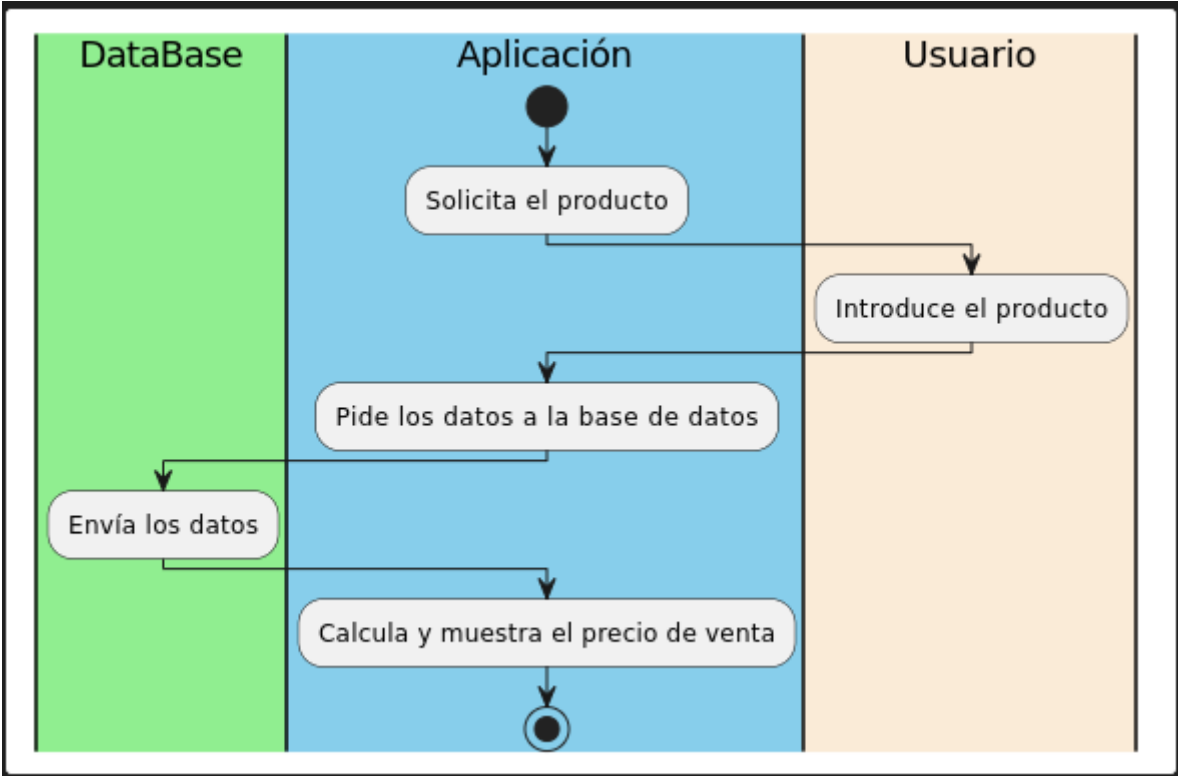
**Listar Compras por Código Postal**



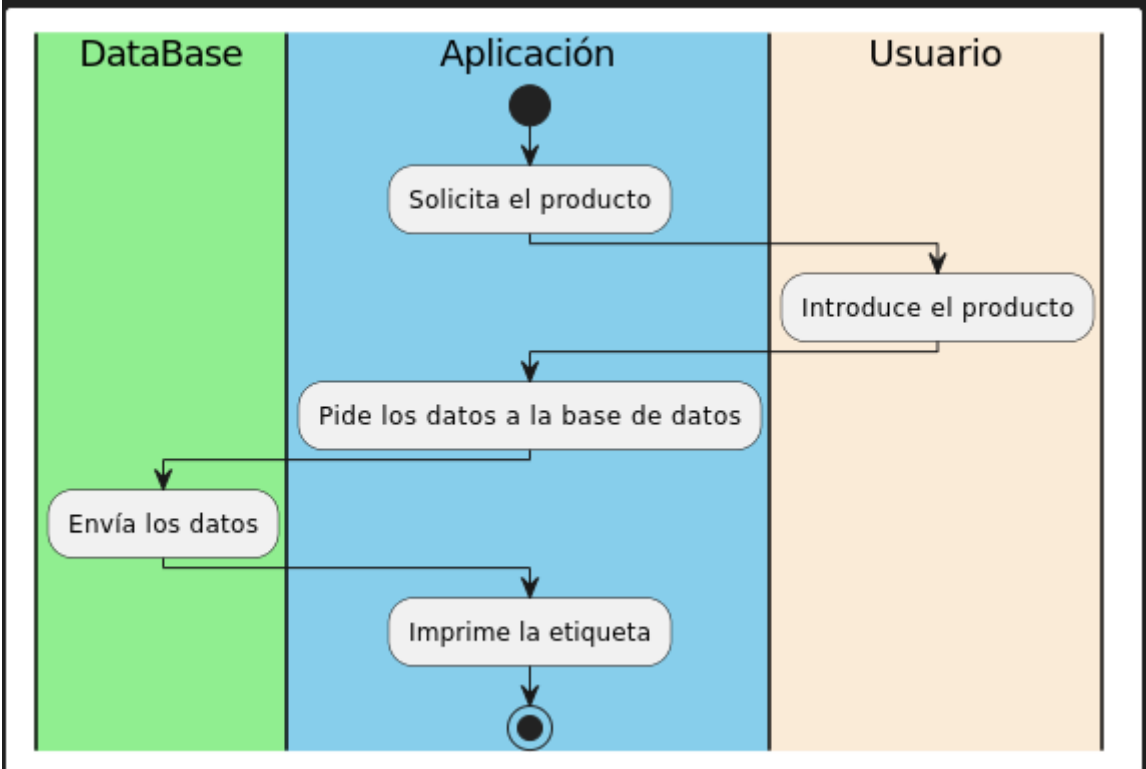
**Listar Stock**



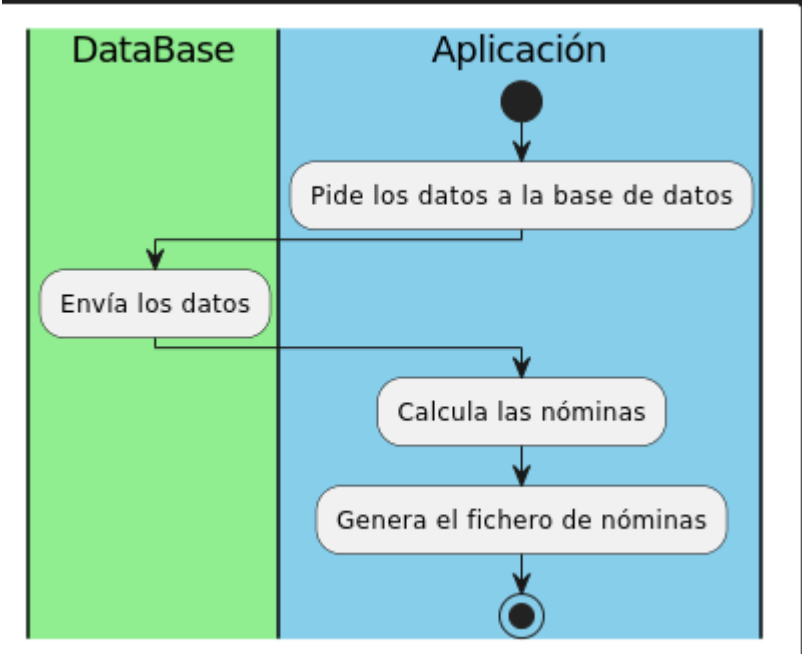
**Obtener Precio de Venta**



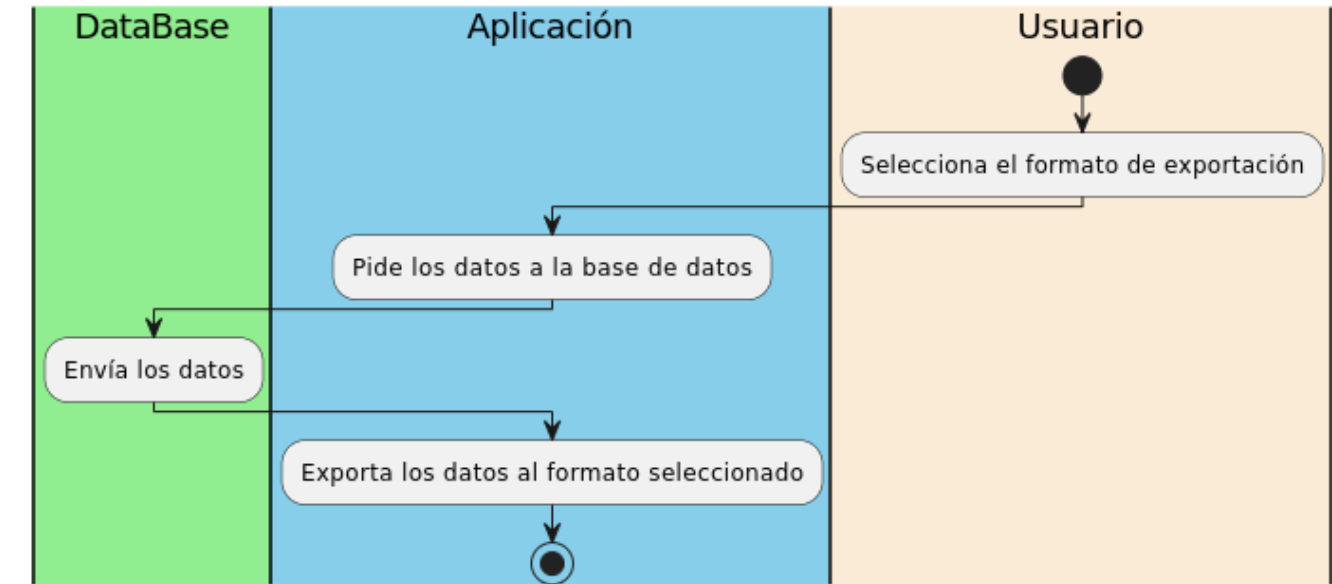
**Imprimir Etiqueta del Producto**



**Calcular las Nóminas de los Trabajadores**

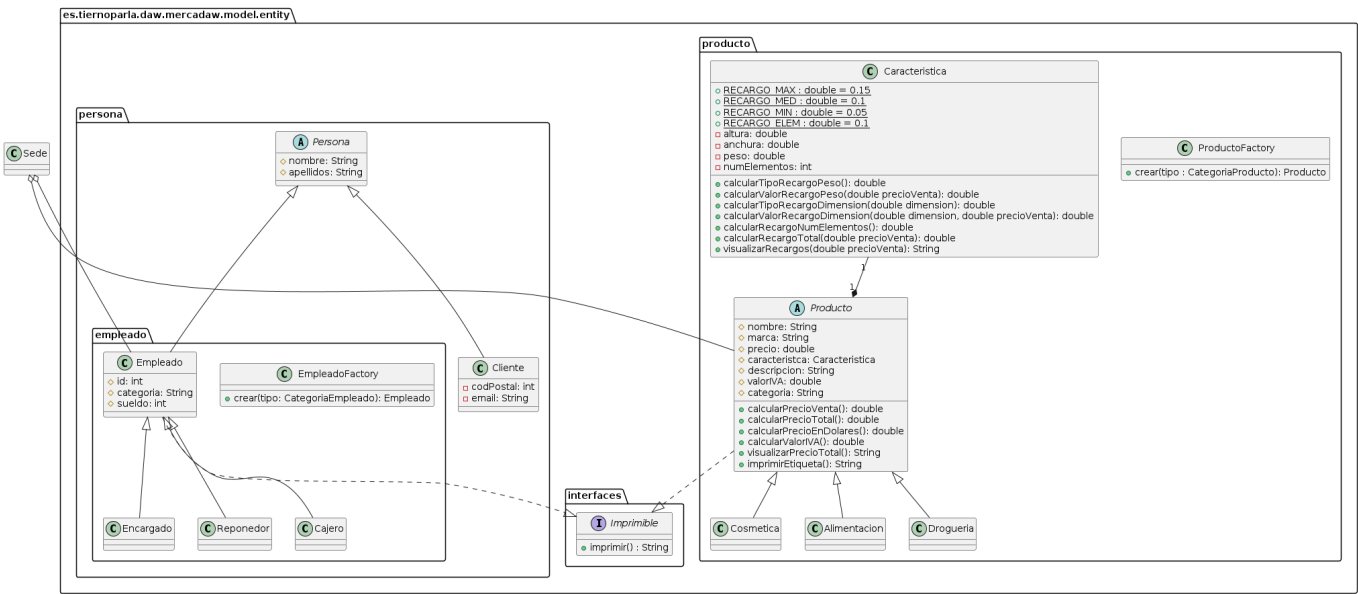


**Exportar Datos**



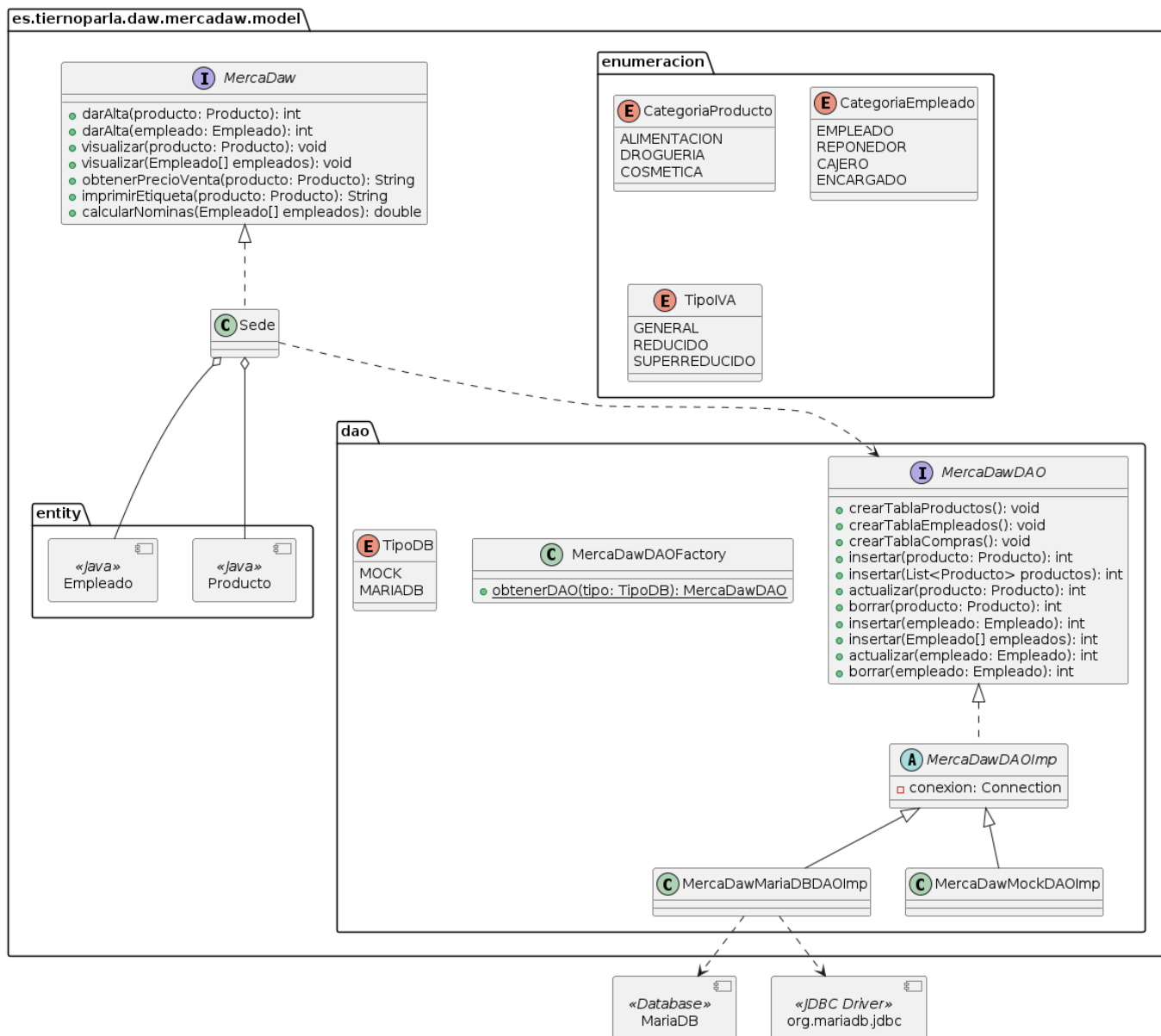
2.3 DIAGRAMAS DE CLASES

ENTITY

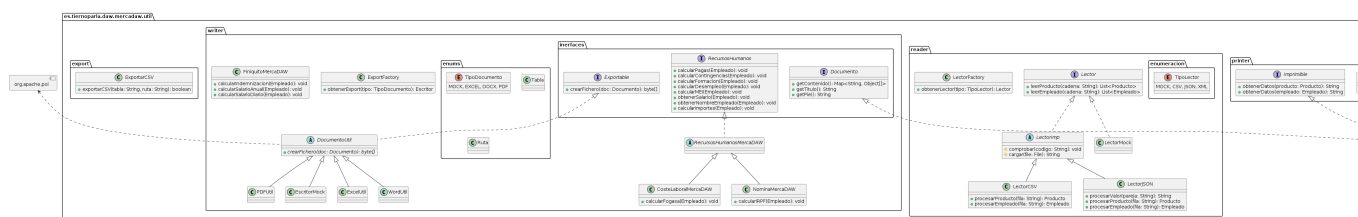


MODEL

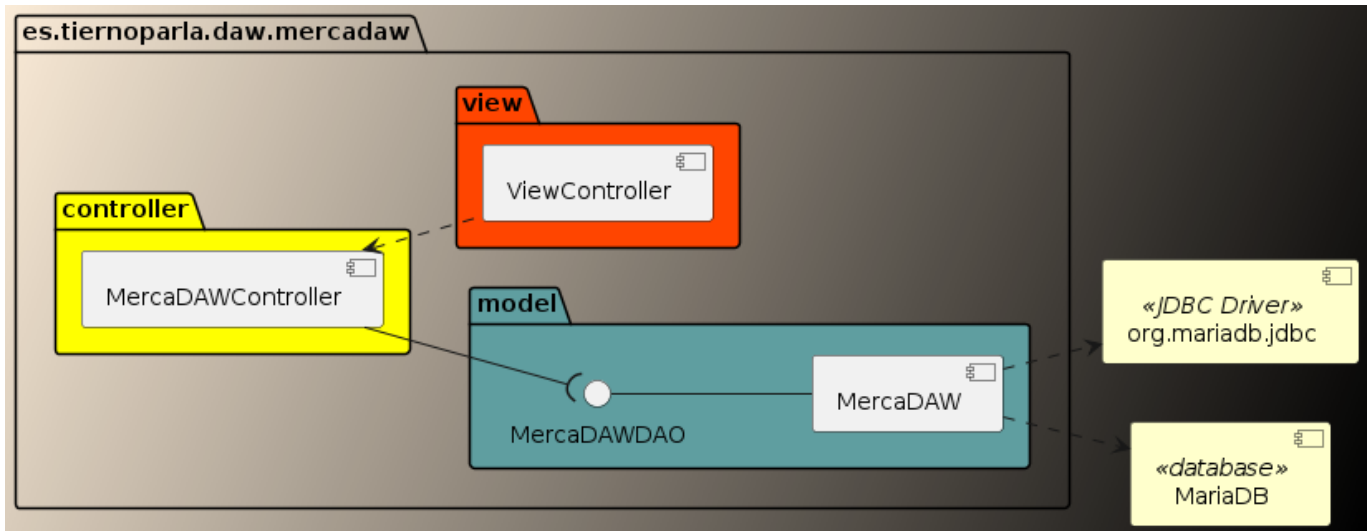




## UTIL



## 2.4 DIAGRAMA DE COMPONENTES



### 3. OBSERVACIONES DEL PROYECTO

#### Clase FileChooser

He utilizado la clase `FileChooser` para que salga el selector de archivos para elegir un fichero en la importacion de productos y empleados.

#### Libreria Apache Commons

He utilizado la libreria Apache Commons para exportar la base de datos a ficheros CSV.

#### Marker Interface

Hemos usado para la interfaz gestionable el patrón de diseño Marker Interface, que permite clasificar clases sin definir su funcionalidad.

### 4. INSTRUCCIONES

Para hacer funcionar la Aplicación correctamente

#### Scripts

Para que los scripts funcionen correctamente hay que darle permisos de ejecución, si no, estos no funcionarán y la aplicacion se cerrará al momento de ejecutar la acción que utilice estos mismos. Para darles permiso de ejecucion, ejecutar el siguiente comando en la carpeta raiz del proyecto:

```
sudo chmod +x *.sh
```

#### Carga de datos

En la raiz del proyecto encontrarás un archivo `.sql` donde están las creaciones de las tablas y las vistas, también están todos los inserts. Para ejecutarlo hay que abrir la extension en VSCode de

MariaDB y conectarse a la base de datos con el usuario root después de haber hecho `docker compose up`. Abrir una nueva **Query**, pegar el `.sql` y ejecutarlo.

## 5. PRUEBAS

Vamos a proceder a documentar las pruebas que vamos a realizar sobre nuestra aplicación.

En esta ocasión usaremos el método de **caja negra** ya que vamos a hacerlas antes de tener el código.

### Clases de Equivalencia

- Vamos a determinar las clases de Equivalencia.

```
- Peso Valido >0
- Altura/Anchura Valida >0
- Numero Elementos Validos >0
- Precio Valido >0
```

### Conjetura de errores

- En conjetura de errores al ser productos sabemos que todo no puede ser 0 , ni menor de 0.

```
- Conjetura de errores <=0
```

### Limites

- Dentro de los valores validos de los elementos de entrada ,tenemos ciertos limites de los cuales cambiará la actuación del precio respecto a ellos.

1. Peso

entradas	salida
<=1	5,00 %
>1<=5	10,00 %
>5	15,00 %

2. Altura y anchura

entradas	salida
<=0,5	0,00 %
>0,5	10,00 %

### 3. Numero de elementos

entrada	salida
$N \leq 2$	0
$N < 2$	$N * 0,10$

### Tabla de apoyo para los casos de prueba

- Al tener tantos valores he hecho una tabla de apoyo en la que me fijaré para hacer los casos de prueba

Clases de <u>equivalencia peso</u>	Limites	entradas	Salidas	precio suplementos
Cevp1	$\leq 1 \& \> 0$		1	0,05
Cevp2	$> 1 \leq 5$		2	0,1
Cevp3	$> 5$		6	0,15
Cenvp1	$\leq 0$		-1	error
				0,00 €
Clases de <u>equivalencia dimension</u>	Limites	entradas	Salidas	
Cevd1	$\leq 0,5 > 0$		0,5	0,05
Cevd2	$> 0,5$		1	0,1
Cenvd1	$\leq 0$		-1	error
Clases de <u>equivalencia elementos</u>	Limites	entradas	Salidas	
Ceve1	$\leq 2 \& \> 0$		2	0
Ceve2	$> 2$		3	0,3 cents
Cenve1	$\leq 0$		-1	error

### Casos de prueba basados en los valores que teníamos antes

entradas	salida		entrada	salida	
$\leq 0,5$	0,00 %		$N \leq 2$	0	
$> 0,5$	10,00 %		$N < 2$	$N * 0,10$	
		Casos	Precio base con beneficio	valores entrada	Precio final
		Caso 1	1,25 €	$\text{NumElem} * (\text{Cevp1} + 2 * \text{Cevd1}) + \text{Ceve1}$	1,63 €
		Caso 2	1,25 €	$\text{NumElem} * (\text{Cevp1} + 2 * \text{Cevd2}) + \text{Ceve1}$	1,88 €
		Caso 3	1,25 €	$\text{NumElem} * (\text{Cevp1} + 2 * \text{Cevd1}) + \text{Ceve2}$	2,11 €
Salidas	precio suplementos	Caso 4	1,25 €	$\text{NumElem} * (\text{Cevp1} + 2 * \text{Cevd2}) + \text{Ceve2}$	2,49 €
0,05	0,06 €	Caso 5	1,25 €	$\text{NumElem} * (\text{Cevp2} + 2 * \text{Cevd1}) + \text{Ceve1}$	1,75 €
0,1	0,13 €	Caso 6	1,25 €	$\text{NumElem} * (\text{Cevp2} + 2 * \text{Cevd2}) + \text{Ceve1}$	2,00 €
0,15	0,19 €	Caso 7	1,25 €	$\text{NumElem} * (\text{Cevp2} + 2 * \text{Cevd1}) + \text{Ceve2}$	2,30 €
error		Caso 8	1,25 €	$\text{NumElem} * (\text{Cevp2} + 2 * \text{Cevd2}) + \text{Ceve1}$	2,00 €
	0,00 €	Caso 9	1,25 €	$\text{NumElem} * (\text{Cevp3} + 2 * \text{Cevd1}) + \text{Ceve1}$	1,88 €
Salidas		Caso 10	1,25 €	$\text{NumElem} * (\text{Cevp3} + 2 * \text{Cevd2}) + \text{Ceve1}$	2,13 €
0,05	0,06 €	Caso 11	1,25 €	$\text{NumElem} * (\text{Cevp3} + 2 * \text{Cevd1}) + \text{Ceve2}$	2,49 €
0,1	0,13 €	Caso 12	1,25 €	$\text{NumElem} * (\text{Cevp3} + 2 * \text{Cevd2}) + \text{Ceve2}$	2,86 €
error		Caso 13	1,25 €	$\text{NumElem} * (\text{Cevp1} + \text{Cevd1} + \text{Cevd2}) + \text{Ceve1}$	1,75 €
Salidas		Caso 14	1,25 €	$\text{NumElem} * (\text{Cevp1} + \text{Cevd1} + \text{Cevd2}) + \text{Ceve2}$	2,30 €
0	0,00 €	Caso 15	1,25 €	$\text{NumElem} * (\text{Cevp2} + \text{Cevd1} + \text{Cevd2}) + \text{Ceve1}$	1,88 €
0,3 cents	0,30 €	Caso 16	1,25 €	$\text{NumElem} * (\text{Cevp2} + \text{Cevd1} + \text{Cevd2}) + \text{Ceve2}$	2,49 €
error		Caso 17	1,25 €	$\text{NumElem} * (\text{Cevp3} + \text{Cevd1} + \text{Cevd2}) + \text{Ceve1}$	2,00 €
		Caso 18	1,25 €	$\text{NumElem} * (\text{Cevp3} + \text{Cevd1} + \text{Cevd2}) + \text{Ceve2}$	2,68 €

### Nomenclatura de pruebas en JUnit 5

Para hacer las pruebas en JUnit 5 en concordancia con la ta tabla , hemos hecho un guía burros muy básico.

En este caso por prueba tenemos Caso! que en Junit 5 sería calcularPrecioCasoUnoTest.

Y así con cada caso sumándole uno.

```
@Test
public void calcularPrecioCasoUnoTest() {
    Producto p = new Alimento("a", "a", "a", 1, 0, 0, 0.5, 1);
```

```
    assertEquals(1.44, p.calcularPrecio(), 0.09);  
}
```

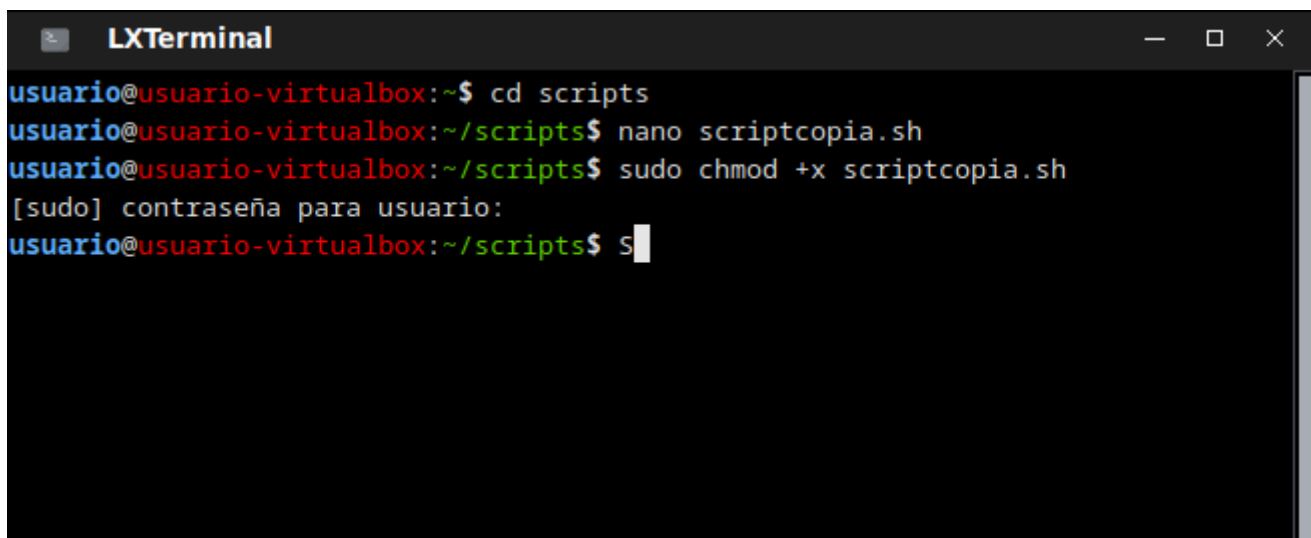
## 6. COPIA DE SEGURIDAD SCRIPT

Abrimos terminal y creamos una carpeta donde se vayan a depositar los backup de la base de datos (Destino)

Creamos otra carpeta donde vamos a tener nuestro script que generara copias de seguridad y las exportara en formato zip con el nombre de bacap y la fecha del mismo

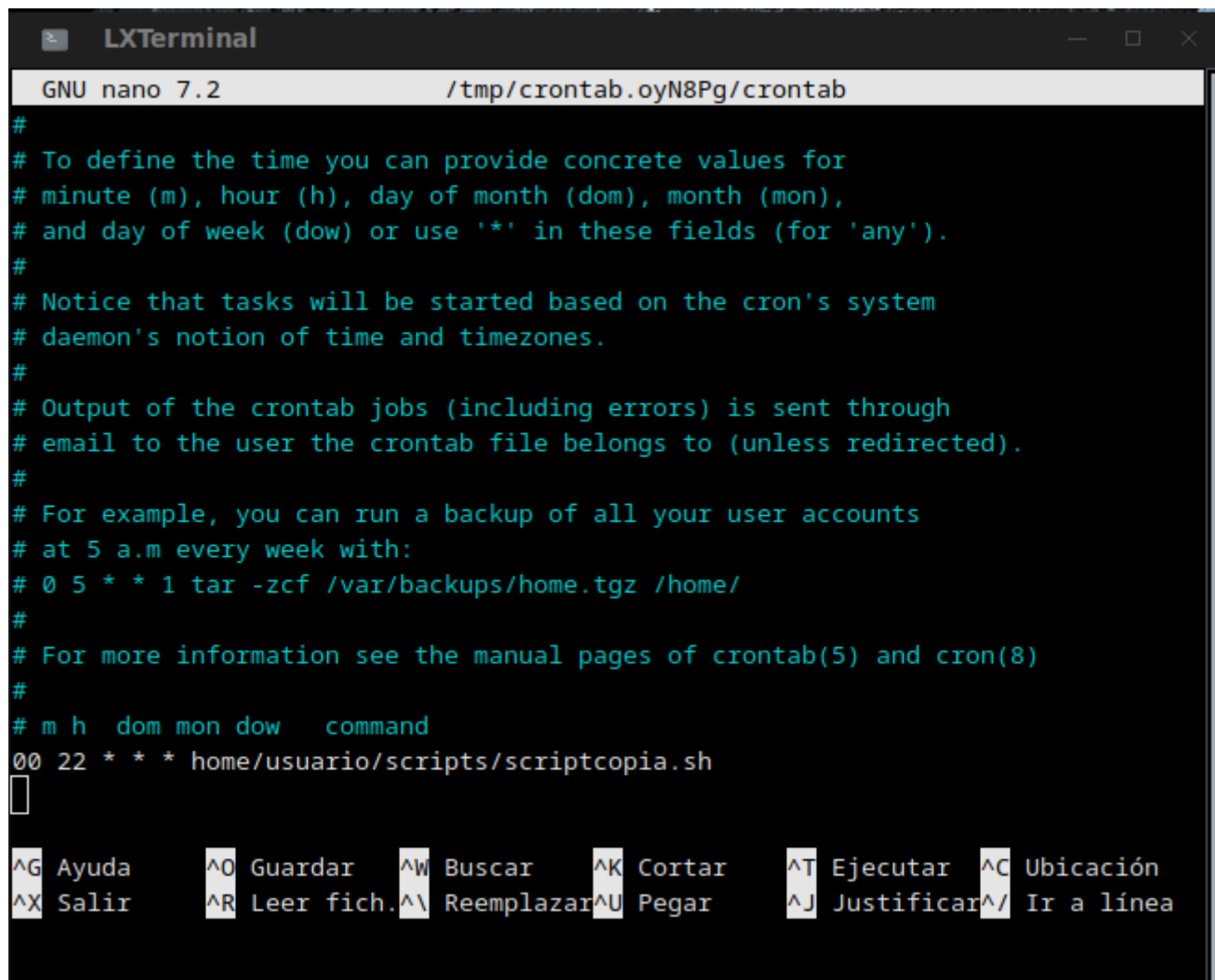
```
1 #!/bin/zsh  
2  
3 mvn exec:java -Dexec.mainClass="es.tiernoparla.daw.mercadaw.CopiaSeguridad"  
4  
5 ORIGEN="backup/"  
6 DESTINO="/home/rodri/CopiaSeguridadMercaDaw/"  
7  
8 DATE=$(date +%Y-%m-%d_%H-%M-%S)  
9 FICHERO="backup_$(DATE).zip"  
10  
11 zip -r "$DESTINO/$FICHERO" "$ORIGEN"  
12
```

Damos permiso de ejecución con el comando chmod +x



```
LXTerminal  
usuario@usuario-virtualbox:~$ cd scripts  
usuario@usuario-virtualbox:~/scripts$ nano scriptcopia.sh  
usuario@usuario-virtualbox:~/scripts$ sudo chmod +x scriptcopia.sh  
[sudo] contraseña para usuario:  
usuario@usuario-virtualbox:~/scripts$ s
```

Por ultimo ejecutamos el comando crontab -e y ponemos cada cuanto tiempo(minutos,hora,dia,mes y dia de la semana) queremos que se ejecute nuestro script



```
GNU nano 7.2 /tmp/crontab.oyN8Pg/crontab
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
00 22 * * * home/usuario/scripts/scriptcopia.sh

```

<b>^G</b> Ayuda	<b>^O</b> Guardar	<b>^W</b> Buscar	<b>^K</b> Cortar	<b>^T</b> Ejecutar	<b>^C</b> Ubicación
<b>^X</b> Salir	<b>^R</b> Leer fich.	<b>^\\</b> Reemplazar	<b>^U</b> Pegar	<b>^J</b> Justificar	<b>^/</b> Ir a línea