

# HMN LOGISTICS

Desarrollado por Hugo Díaz

# ÍNDICE

<b>1. ANÁLISIS.....</b>	3
<b>1.1. Introducción.....</b>	3
<b>1.2 Requerimientos de la aplicación .....</b>	3
<b>1.2.1 Requisitos funcionales.....</b>	3
<b>1.2.2 Requisitos no funcionales .....</b>	4
<b>1.3 Requerimientos de Hardware y Software.....</b>	5
<b>1.3.1 Hardware .....</b>	5
<b>1.3.2 Software.....</b>	5
<b>1.4 Casos de uso .....</b>	7
<b>2. DISEÑO .....</b>	28
<b>2.1 Diagrama de clases .....</b>	28
<b>2.2 Estructura de Almacenamiento.....</b>	36
<b>3. CODIFICACIÓN.....</b>	37
<b>4. PRUEBAS .....</b>	39

## 1. ANÁLISIS

### 1.1. Introducción

Necesitamos desarrollar una aplicación que nos permita gestionar todo lo relacionado con la empresa HMNLogistics, mediante la cual podremos borrar, añadir, visualizar y modificar los registros que contiene nuestro negocio.

### 1.2 Requerimientos de la aplicación

#### 1.2.1 Requisitos funcionales

Con esta aplicación, el usuario podrá seleccionar diferentes opciones del programa, las cuales son:

- a. **ARTÍCULOS** → El usuario podrá añadir contactos con los siguientes campos a llenar: un código del cliente el cual posee el artículo (este cliente debe de estar dado de alta en el sistema previamente), el nombre del artículo, una breve descripción, y su categoría (Deportes, Ocio, ...)., Además podremos borrar un artículo mediante su código de artículo. Por último, podremos visualizar todos nuestros artículos, con un buscador (con opción de reconocimiento de voz) el cual filtrará nuestra búsqueda y seleccionando un artículo concreto podremos modificar algunos de sus campos.
- b. **CLIENTES** → El usuario podrá añadir clientes con los siguientes campos a llenar: un nombre y un código de sucursal (la cual debe de estar previamente dada de alta en el sistema. Además, podremos borrar un cliente mediante su código de cliente. Por último, podremos visualizar todos nuestros clientes, con un buscador (con opción de reconocimiento de voz) el cual filtrará nuestra búsqueda y seleccionando un cliente concreto podremos modificar algunos de sus campos.
- c. **SUCURSALES** → El usuario podrá añadir sucursales con los siguientes campos a llenar: una provincia española (incluyendo sus dos ciudades autónomas) y su dirección. Además, podremos borrar una sucursal mediante su código de sucursal. Por último, podremos visualizar todas nuestras sucursales, con un buscador (con opción de reconocimiento de voz) el cual filtrará nuestra búsqueda y seleccionando una sucursal concreta podremos modificar algunos de sus campos.
- d. **PEDIDOS** → El usuario podrá añadir pedidos con los siguientes campos a llenar: un código de distribuidor (el cual debe de estar registrado en el sistema), un código de sucursal (que también debe de estar dada de alta), la fecha del pedido, el número de artículos que contiene el pedido, el peso total (en kg) y el precio (en €). Además podremos borrar un pedido mediante su código de pedido. Por último, podremos visualizar todas nuestras sucursales, con un buscador (con opción de reconocimiento de voz) el cual filtrará nuestra búsqueda y seleccionando un pedido concreto podremos modificar algunos de sus campos.

- e. **DISTRIBUIDORES** → El usuario podrá añadir distribuidores mediante un nombre. Además podremos borrar un distribuidor mediante su código de distribuidor. Por último, podremos visualizar todas nuestras sucursales, con un buscador (con opción de reconocimiento de voz) el cuál filtrará nuestra búsqueda.
- f. **DEPARTAMENTOS** → El usuario podrá añadir departamentos con los siguientes campos a llenar: un código de sucursal (la cuál debe de estar dada de alta en el sistema), y el nombre del departamento. Además podremos borrar un departamento mediante su código de departamento. Por último, podremos visualizar todas nuestras sucursales, con un buscador (con opción de reconocimiento de voz) el cuál filtrará nuestra búsqueda y seleccionando un departamento concreto podremos modificar algunos de sus campos.
- g. **EMPLEADOS** → El usuario podrá añadir empleados con los siguientes campos a llenar: un código de departamento (el cuál debe de estar dado de alta en el sistema), el DNI del empleado, su nombre, sus apellidos, su fecha de nacimiento, el salario que va a recibir, el domicilio, un teléfono de contacto y el oficio que va a desempeñar en la empresa. Además podremos borrar al usuario mediante su DNI. En este caso, los empleados podrán ser visualizados en dos listas distintas, una que contendrá sus datos personales y otra los datos más importantes para la empresa, se podrán hacer búsquedas para filtrar los datos (con opción de reconocimiento de voz) y al seleccionar un empleado podremos modificar sus atributos
- h. **GUARDAR** → El usuario tendrá un opción en el menú principal que le permitirá guardar todos los cambios de manera inmediata.
- i. **GUARDAR Y SALIR** → Al igual que la anterior opción nos guardará todos los cambios de manera instantánea y cerrará la aplicación.

### 1.2.2 Requisitos no funcionales

La aplicación será diseñada para cumplir los siguientes requisitos:

- a. **USABILIDAD:** Nuestra aplicación debe de ser bastante simple y sencilla para que cualquier usuario la pueda usar sin ningún problema (debe de estar preparada para cualquier entrada por parte del usuario), y debe de tener una interfaz lo más visual y atractiva posible.
- b. **RENDIMIENTO:** El programa debe de ser lo más efectivo y rápido posible, debe de tener lo justo y necesario para no forzar errores, y que nuestra aplicación ejecute el recorrido más corto posible para una opción de la aplicación (como por ejemplo, listar), eso sí, debiendo de pasar por todos los caminos implementados (todos los casos posibles).
- c. **ESCALABIDAD:** La aplicación debe de soportar una gran cantidad de datos sin que ello afecte a la rapidez de la aplicación.

- d. **COMPATIBILIDAD:** El programa debe de ser compatible con distintos S.O. como por ejemplo Linux, Windows, MacOS...
- e. **SEGURIDAD:** Debemos de proporcionar seguridad al usuario, y que su información enviada vaya protegida y segura, para garantizar la privacidad.
- f. **DOCUMENTACIÓN:** La aplicación debe de estar documentada de una manera clara y legible, para que cualquier desarrollador mediante esta documentación pueda seguir manteniendo de la forma menos compleja posible dicha app.
- g. **DISPONIBILIDAD:** El programa debe de poder usarse en todo momento, que no genere caídas y que no tenga posea de inactividad.

### 1.3 Requerimientos de Hardware y Software

Para poder soportar esta agenda telefónica necesitamos los siguientes requerimientos:

#### 1.3.1 Hardware

- a. **DISPOSITIVO** → debemos tener un dispositivo compatible, que puede hacer uso de la aplicación, en este caso un ordenador o un portátil.
- b. **PROCESADOR** → Un procesador que tenga la suficiente potencia y velocidad como para ejecutar la aplicación y funcione de una forma dinámica.
- c. **RAM** → Se recomienda un Memoria RAM que pueda mantener de una forma óptima el funcionamiento de la aplicación.
- d. **ESPACIO DE ALMACENAMIENTO** → Un espacio de almacenamiento para que se puedan guardar todo los datos de la aplicación de HMNLogistics.

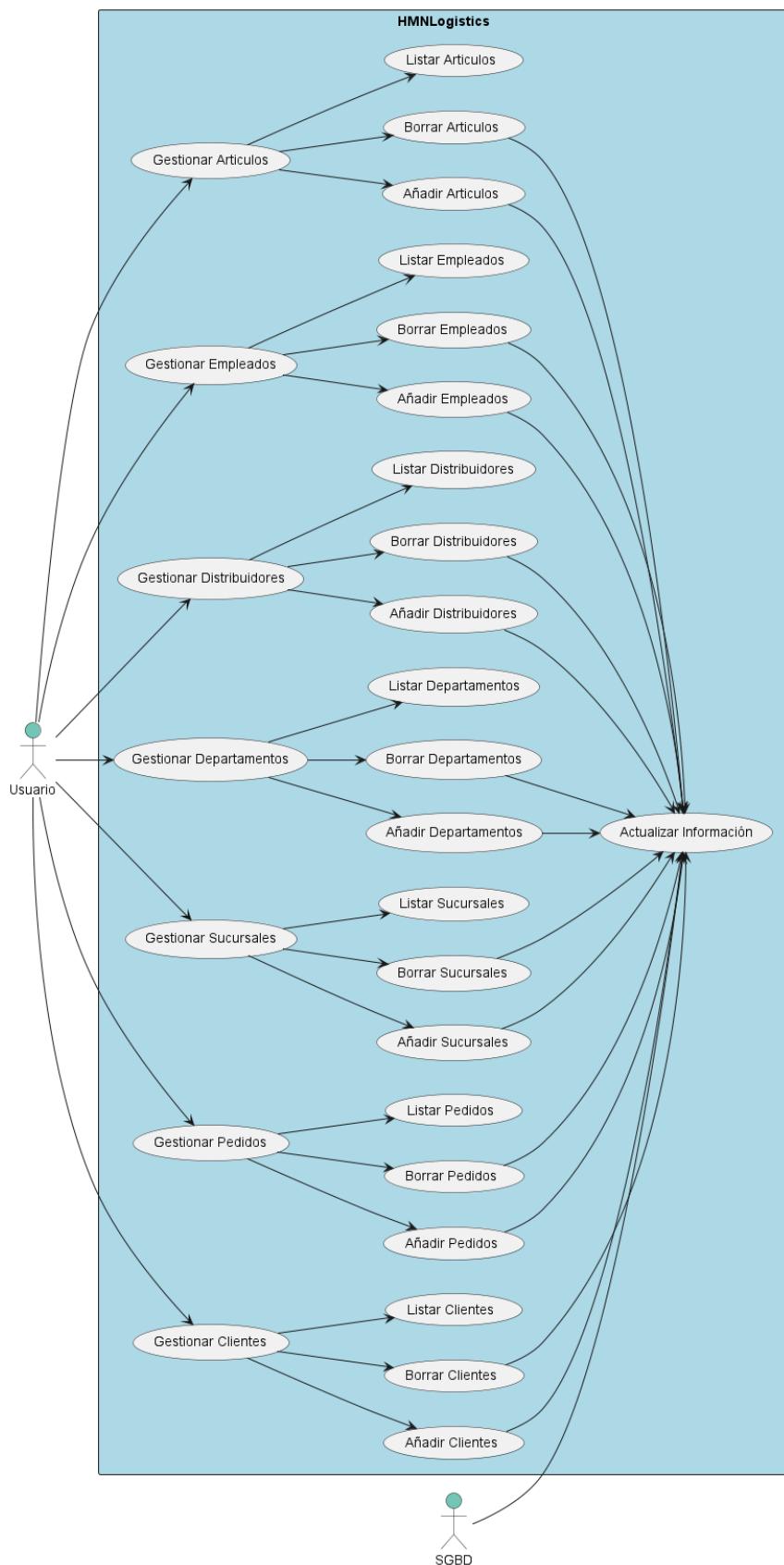
#### 1.3.2 Software

- a. **SISTEMA OPERATIVO** → La aplicación es compatible con Windows, Linux y MacOS.
- b. **VERSION DEL S.O.** → La aplicación requiere una versión mínima de S.O. para poder poner en funcionamiento la agenda telefónica.
- c. **BASES DE DATOS** → La agenda utiliza un SGBD para almacenar toda la información, en concreto MySQL.



- d. **ENTORNO DE DESARROLLO →** Como mínimo usaremos la consola de Python para poner en funcionamiento la aplicación, pero es recomendable un entorno de desarrollo como por ejemplo PyCharm o Visual Studio Code.

## 1.4 Casos de uso





Tenemos 2 actores en esta aplicación:

**USUARIO:**



La aplicación tendrá un acceso directo en el escritorio con el nombre de hmnllogistics y su correspondiente ícono

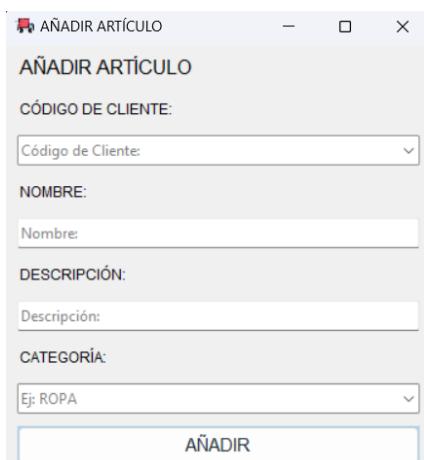
Al abrir la aplicación se encontrará el siguiente menú:



Al clicar la opción de artículos nos aparecerá el siguiente submenú con tres opciones



Con la opción de añadir artículos se nos abrirá la siguiente ventana



AÑADIR ARTÍCULO

CÓDIGO DE CLIENTE:

NOMBRE:

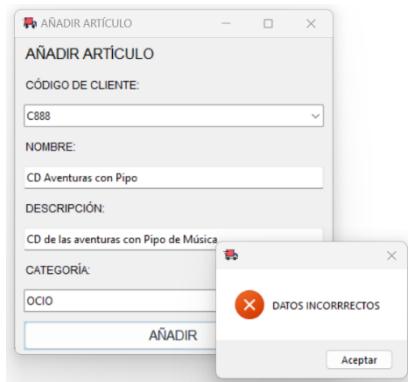
DESCRIPCIÓN:

CATEGORÍA:

Ej: ROPA

AÑADIR

Si intentamos añadir un artículo con un código de cliente no registrado en el sistema nos saldrá la siguiente advertencia



AÑADIR ARTÍCULO

CÓDIGO DE CLIENTE:

C888

NOMBRE:

CD Aventuras con Pipo

DESCRIPCIÓN:

CD de las aventuras con Pipo de Música

CATEGORÍA:

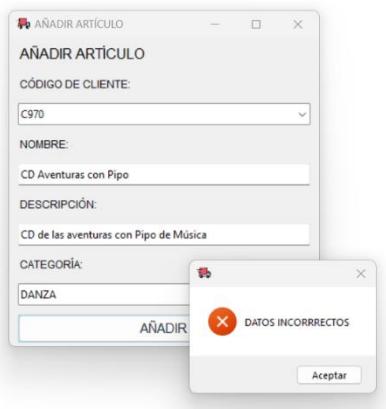
OCIO

AÑADIR

DATOS INCORRECTOS

Aceptar

También si añadimos una categoría distinta a las del desplegable



AÑADIR ARTÍCULO

CÓDIGO DE CLIENTE:

C970

NOMBRE:

CD Aventuras con Pipo

DESCRIPCIÓN:

CD de las aventuras con Pipo de Música

CATEGORÍA:

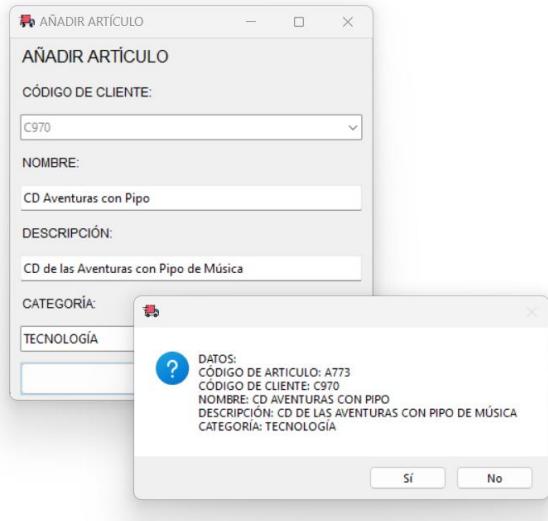
DANZA

AÑADIR

DATOS INCORRECTOS

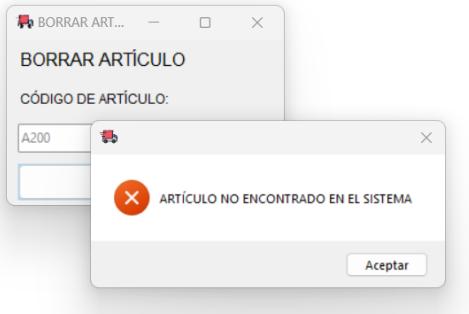
Aceptar

Si añadimos valores correctos no saldrá la siguiente ventana, dónde se mostrarán nuestros datos introducidos

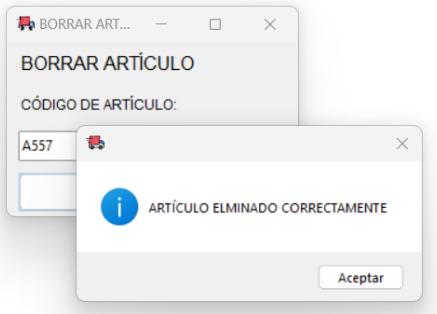


Si clicamos en si se añadirá al sistema, en caso de seleccionar no, se cerrará la ventana y no se producirá ningún cambio.

Si abrimos la ventana de borrado de artículos e introducimos un artículo no registrado en el sistema nos sucederá lo siguiente



Si seleccionamos un artículo registrado



En la opción de listar nos encontraremos la siguiente interfaz



MIS ARTÍCULOS

REALIZAR BUSQUEDA:

NUM_SUCURSAL	COD_ARTICULO	COD_CLIENTE	NOMBRE	DESCRIPCION	CATEGORIA
1	A696	C181	ZAPATILLAS NEW GENERA	NUEVA COLECCIÓN GUCCI	ROPA
2	A184	C815	BALÓN DE FÚBOL LA LIGA	BALÓN DE JOMA	DEPORTES
3	A714	C815	ZAPATILLAS VITALIS 21	ZAPATILLAS DE LA MARCA JO	DEPORTES
4	A439	C40	GALLETAS CAMPURRIANA	GALLETAS DE LA ABUELA DE T	ALIMENTACI
5	A404	C380	TOSTA RICA OCEANIX	GALLETAS TOSTA RICA DE CH	ALIMENTACI
6	A141	C663	LIBRO DE GERÓNIMO STIL	LIBROS DE GERÓNIMO STIL	OCIO
7	A926	C181	ZAPATILLAS CROC AZULE	ZAPATILLAS DE LA MODALIDA	ROPA
8	A69	C987	AURICULARES GEAR LOGIT	AURICULARES DE LA NUEVA C	TECNOLOGÍA

MODIFICAR ARTÍCULO

Podremos hacer una búsqueda de cualquier campo mediante texto o voz, para que nos sea más fácil encontrar un artículo



MIS ARTÍCULOS

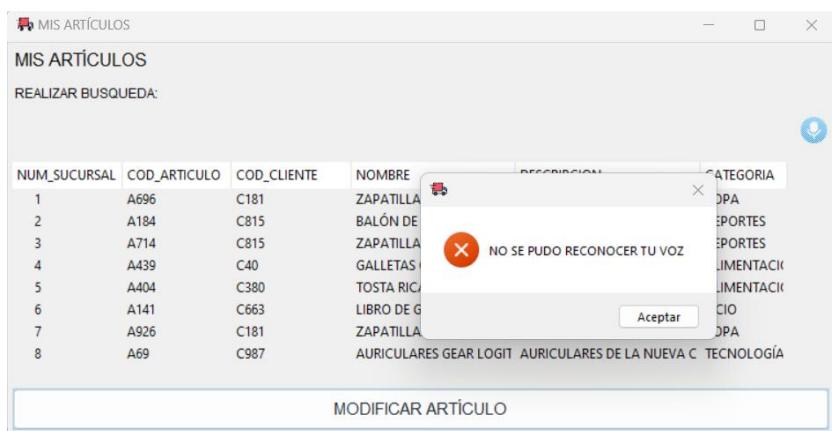
REALIZAR BUSQUEDA:

DEPORTES

NUM_SUCURSAL	COD_ARTICULO	COD_CLIENTE	NOMBRE	DESCRIPCION	CATEGORIA
1	A184	C815	BALÓN DE FÚBOL LA LIGA	BALÓN DE JOMA	DEPORTES
2	A714	C815	ZAPATILLAS VITALIS 21	ZAPATILLAS DE LA MARCA JO	DEPORTES

MODIFICAR ARTÍCULO

Si el reconocedor no detecta la voz sucederá lo siguiente:



MIS ARTÍCULOS

REALIZAR BUSQUEDA:

NUM_SUCURSAL	COD_ARTICULO	COD_CLIENTE	NOMBRE	DESCRIPCION	CATEGORIA
1	A696	C181	ZAPATILLA	ZAPATILLA	ROPA
2	A184	C815	BALÓN DE	BALÓN DE	DEPORTES
3	A714	C815	ZAPATILLA	ZAPATILLA	DEPORTES
4	A439	C40	GALLETAS	GALLETAS	ALIMENTACI
5	A404	C380	TOSTA RIC	TOSTA RIC	ALIMENTACI
6	A141	C663	LIBRO DE	LIBRO DE	OCIO
7	A926	C181	ZAPATILLA	ZAPATILLA	ROPA
8	A69	C987	AURICULARES GEAR LOGIT	AURICULARES DE LA NUEVA C	TECNOLOGÍA

MODIFICAR ARTÍCULO

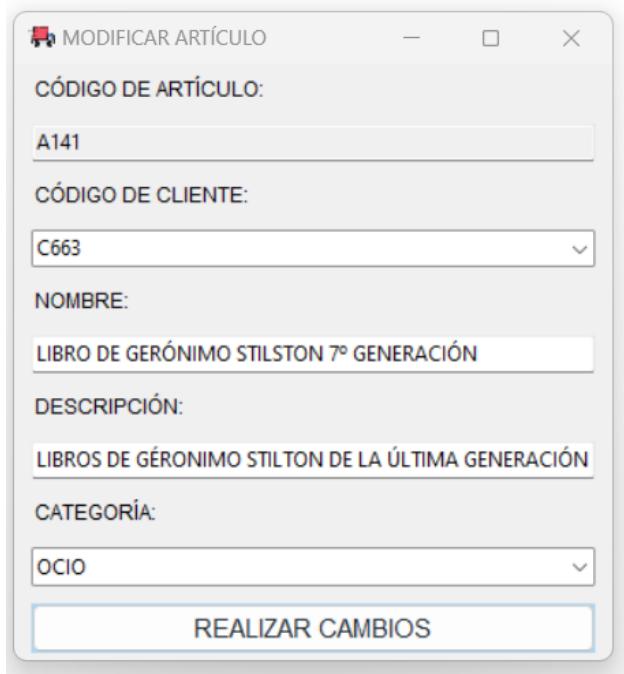
NO SE PUDO RECONOCER TU VOZ

Aceptar

Si queremos modificar un artículo sin ninguno seleccionado nos saltará la siguiente advertencia:



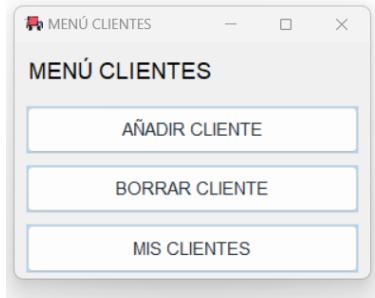
Si seleccionamos uno y le damos modificar nos saltará una ventana con sus campos a modificar, como es lógico el campo de código de artículo es solo lectura y no se puede modificar



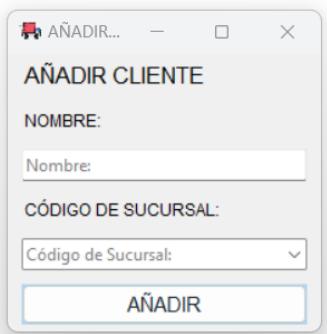
The screenshot shows a window titled 'MODIFICAR ARTÍCULO' with fields for modifying an article. The 'CÓDIGO DE ARTÍCULO' field contains 'A141'. The 'CÓDIGO DE CLIENTE' dropdown contains 'C663'. The 'NOMBRE' field contains 'LIBRO DE GERÓNIMO STILTON 7º GENERACIÓN'. The 'DESCRIPCIÓN' field contains 'LIBROS DE GÉRONIMO STILTON DE LA ÚLTIMA GENERACIÓN'. The 'CATEGORÍA' dropdown contains 'OCIO'. At the bottom is a 'REALIZAR CAMBIOS' button.

Tendrá las mismas restricciones que la ventana de añadir artículos.

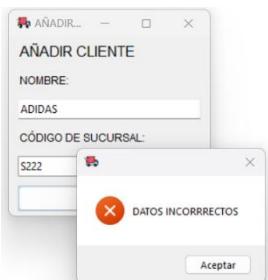
Al clicar la opción de clientes nos aparecerá el siguiente submenú con tres opciones



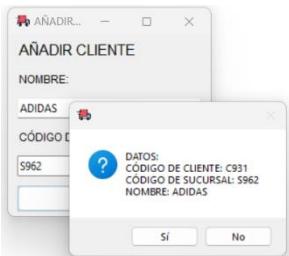
Al añadir un cliente nos encontramos la siguiente ventana:



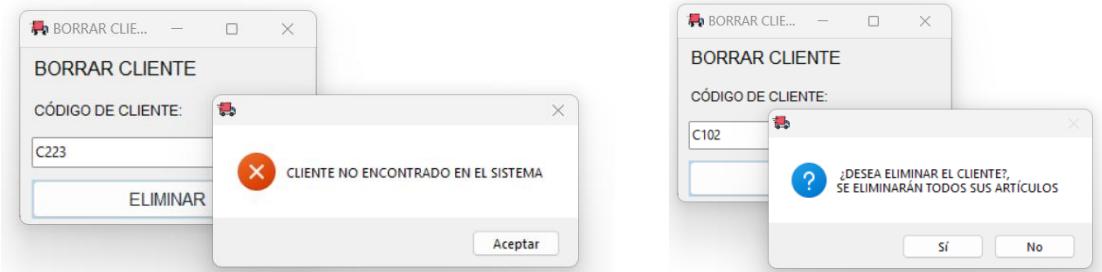
Al añadir un cliente de una sucursal no dada de alta sucede lo siguiente:



Si está dada de alta nos muestra los datos, que clicando la opción "si" se añadirán al sistema



En la opción borrar clientes nos pedirá un borrado con el código del cliente, si introducimos un cliente inexistente nos saltará el siguiente error, si introducimos uno existente borraremos ese cliente y todos sus artículos correspondientes.



Si clicamos en el submenú mis clientes obtendremos la siguiente pantalla:

MIS CLIENTES			
REALIZAR BUSQUEDA:			
NUM_CLIENTE	COD_CLIENTE	COD_SUCURSAL	NOMBRE
1	C39	S21	ANAYA
2	C102	S21	SMINT
3	C987	S301	LOGITECH
4	C970	S603	ZARA
5	C773	S650	OXFORD
6	C815	S21	JOMA
7	C380	S962	TOSTA RICA
8	C797	S154	BERSHKA
9	C44	S37	L'OREAL

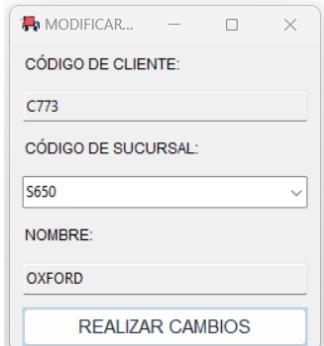
**MODIFICAR CLIENTE**

Al igual que en artículos podremos hacer búsqueda tanto por texto como por voz.

MIS CLIENTES			
REALIZAR BUSQUEDA:			
NUM_CLIENTE	COD_CLIENTE	COD_SUCURSAL	NOMBRE
1	C39	S21	ANAYA
2	C102	S21	SMINT
3	C815	S21	JOMA
4	C631	S21	BIC
5	C11	S21	JBL

**MODIFICAR CLIENTE**

Y si modificamos un cliente con un cliente seleccionado nos aparecerá lo siguiente:



CÓDIGO DE CLIENTE:  
C773

CÓDIGO DE SUCURSAL:  
S650

NOMBRE:  
OXFORD

**REALIZAR CAMBIOS**

Con el campo código de cliente como solo lectura. Tendrá las mismas restricciones que la ventana de adición de clientes.

Si abrimos el menú de sucursales nos aparecerán 3 posibles opciones



MENÚ SUCURSALES

AÑADIR SUCURSAL

BORRAR SUCURSAL

MIS SUCURSALES

El clicar añadir sucursal nos aparecerá la siguiente ventana



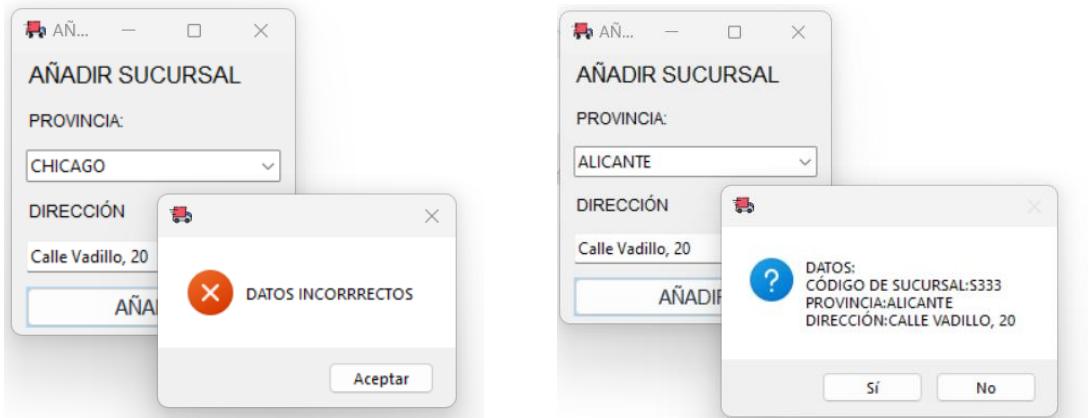
AÑADIR SUCURSAL

PROVINCIA:  
Ej: Álava

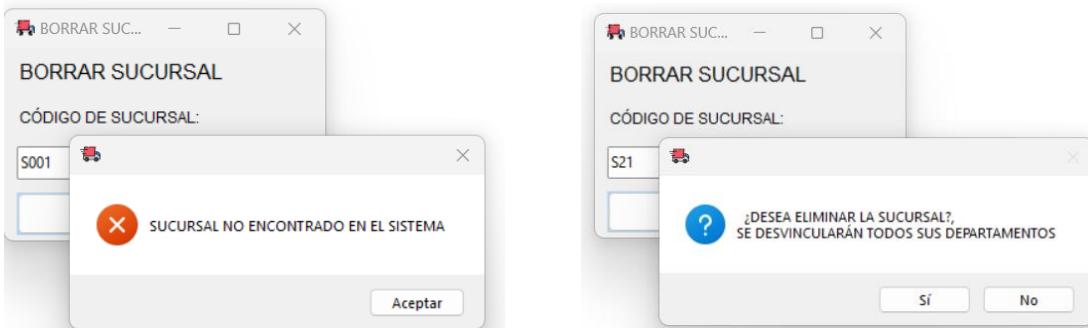
DIRECCIÓN  
Dirección:

**AÑADIR**

Podremos añadir una sucursal mediante una provincia española (además de las ciudades autónomas Ceuta y Melilla) y la dirección de la sucursal, en caso de que la provincia española no exista nos saltará un error de datos inválidos como en las anteriores ventanas, y si son correctos una ventana de confirmación de adición con los datos introducidos.



Borraremos una sucursal por su código de sucursal estas son las posibles opciones, la primera es un código no dado de alta y la segunda si está registrado en el sistema, tenemos que tener en cuenta que si borramos una sucursal, desvinculamos todos sus departamentos.



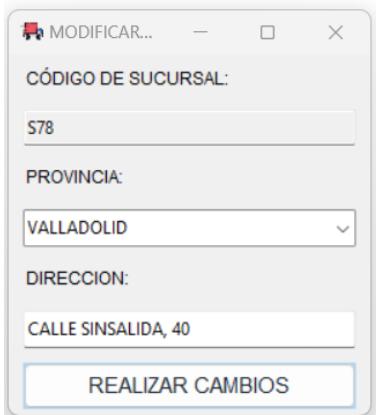
Podemos visualizar las sucursales en la ventana de mis sucursales.



NUM_SUCURSAL	COD_SUCURSAL	PROVINCIA	DIRECCION
1	S000	-----	-----
2	S962	MÁLAGA	CALLE ARBUSTOS, 92
3	S345	CIUDAD REAL	CALLE FERNANDO ALONSO, 71
4	S21	BARCELONA	CALLE VELÁZQUEZ, 61
5	S78	VALLADOLID	CALLE SINSALIDA, 40
6	S650	PONTEVEDRA	AVENIDA LA HISPANIDAD, 44
7	S603	LÉRIDA	CALLE TORRECILLA, 90
8	S154	CASTELLÓN	PLAZA DE LOS MERCADOS 94
9	S296	JAÉN	CALLE BLANCA, 10
10	S301	GUADALAJARA	AVENIDA DEL EJÉRCITO, 08

MODIFICAR SUCURSAL

Al igual que las ventanas anteriores del listado podemos hacer búsquedas tanto por texto como por voz si seleccionamos una sucursal nos aparecerá la siguiente ventana

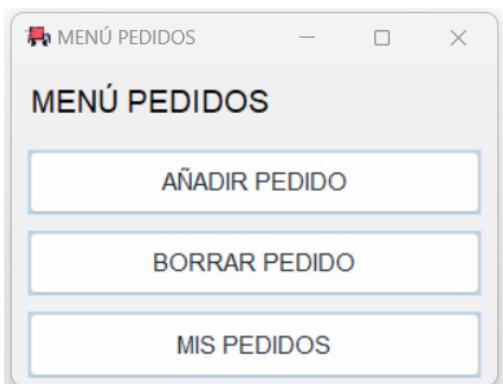


The window title is "MODIFICAR...". It contains the following fields:

- CÓDIGO DE SUCURSAL: S78
- PROVINCIA: VALLADOLID
- DIRECCION: CALLE SINSALIDA, 40
- A blue button labeled "REALIZAR CAMBIOS" (Perform Changes) is at the bottom.

El campo de código de sucursal será de solo lectura y esta ventana tendrá las mismas restricciones que la de añadir sucursales

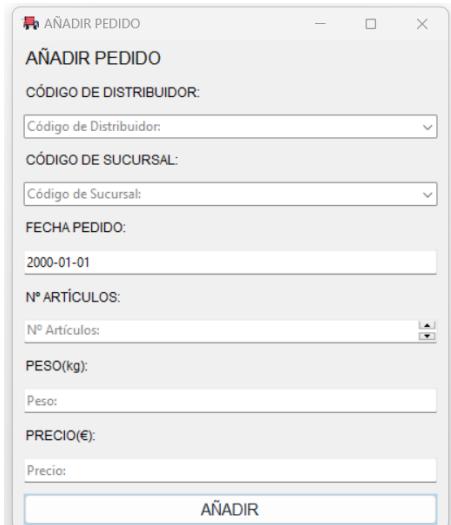
Si clicamos en la opción de pedidos nos aparecerá el siguiente submenú:



The window title is "MENÚ PEDIDOS". It contains the following menu items:

- AÑADIR PEDIDO
- BORRAR PEDIDO
- MIS PEDIDOS

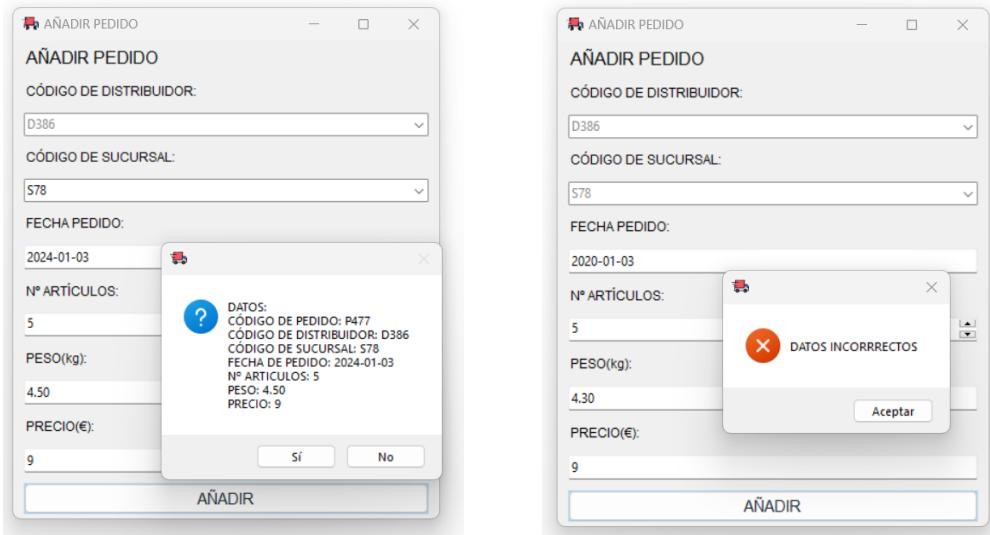
Si seleccionamos la opción añadir pedido nos aparecerán los siguientes campos a rellenar



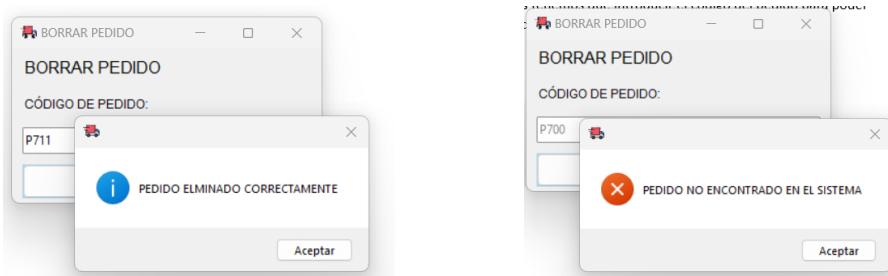
The window title is "AÑADIR PEDIDO". It contains the following fields:

- AÑADIR PEDIDO
- CÓDIGO DE DISTRIBUIDOR: (dropdown menu)
- CÓDIGO DE SUCURSAL: (dropdown menu)
- FECHA PEDIDO: 2000-01-01
- Nº ARTÍCULOS: (dropdown menu)
- PESO(kg): (dropdown menu)
- Peso: (text input)
- PRECIO(€): (text input)
- A blue button labeled "AÑADIR" (Add) is at the bottom.

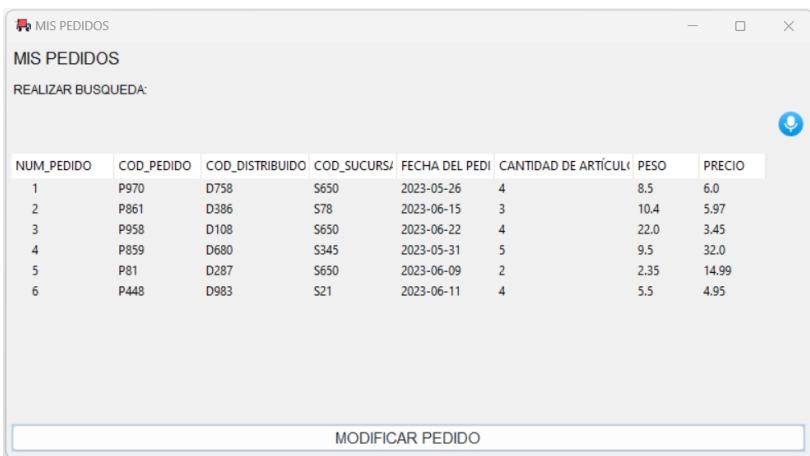
Para poder añadir un pedido deberemos de seleccionar un código de distribuidor y un código de sucursal vigente en el sistema, además de una fecha con un formato 'yyyy-mm-dd' y posterior y a la fecha del sistema y tanto nº de artículos como precio y como peso deberán ser números. En la foto de la derecha para que nos haya saltado el error hemos añadido una fecha anterior a la del sistema.



En la ventana de borrar pedidos tenemos que introducir el código del pedido para poder eliminarlo, nos puede suceder dos situaciones:

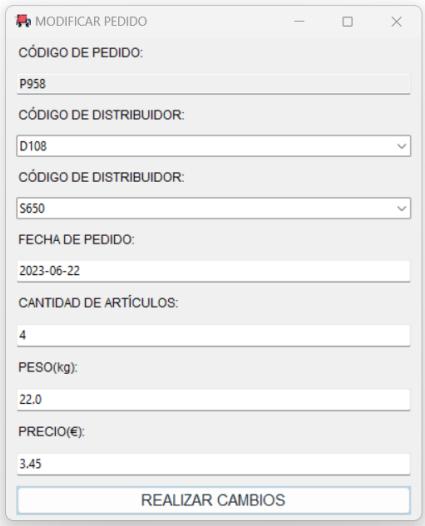


Si queremos visualizar todos nuestros pedidos seleccionaremos la ventana de mis pedidos con el siguiente aspecto:



NUM_PEDIDO	COD_PEDIDO	COD_DISTRIBUIDO	COD_SUCURSAL	FECHA DEL PEDI	CANTIDAD DE ARTÍCULOS	PESO	PRECIO
1	P970	D758	S650	2023-05-26	4	8.5	6.0
2	P861	D386	S78	2023-06-15	3	10.4	5.97
3	P958	D108	S650	2023-06-22	4	22.0	3.45
4	P859	D680	S345	2023-05-31	5	9.5	32.0
5	P81	D287	S650	2023-06-09	2	2.35	14.99
6	P448	D983	S21	2023-06-11	4	5.5	4.95

Al igual que con las ventanas de listado anterior podremos hacer por búsquedas tanto por texto como por voz y si seleccionamos un pedido se nos abrirá la siguiente ventana.



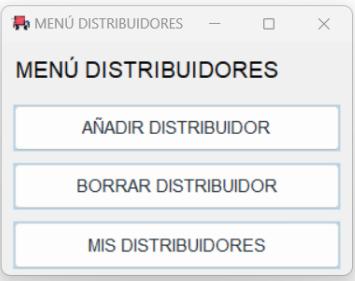
The window title is "MODIFICAR PEDIDO". It contains the following fields:

- CÓDIGO DE PEDIDO: P958
- CÓDIGO DE DISTRIBUIDOR: D108
- CÓDIGO DE DISTRIBUIDOR: S650
- FECHA DE PEDIDO: 2023-06-22
- CANTIDAD DE ARTÍCULOS: 4
- PESO(kg): 22.0
- PRECIO(€): 3.45

At the bottom is a blue button labeled "REALIZAR CAMBIOS".

Tendrá el campo del código de pedido de solo lectura y las mismas restricciones que la ventana anterior

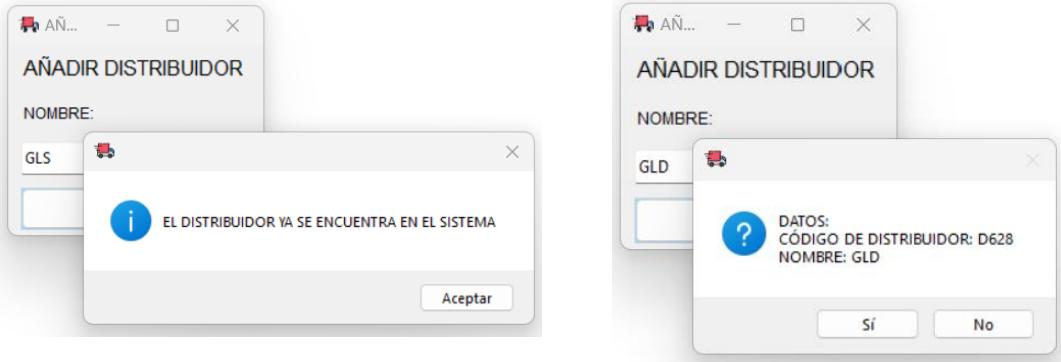
Si seleccionamos el botón de distribuidores se nos abrirá la siguiente ventana:



The window title is "MENÚ DISTRIBUIDORES". It contains three buttons:

- AÑADIR DISTRIBUIDOR
- BORRAR DISTRIBUIDOR
- MIS DISTRIBUIDORES

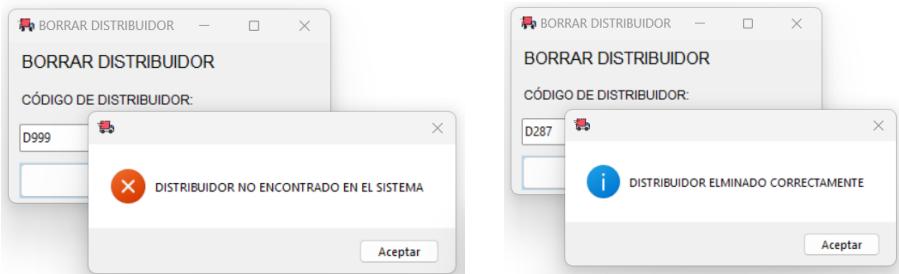
Si clicamos añadir distribuidor nos aparecerá una ventana con un solo campo (nombre), ya que el código de distribuidor lo genera el sistema. El sistema comprobará si el nombre del distribuidor está o no en el sistema.



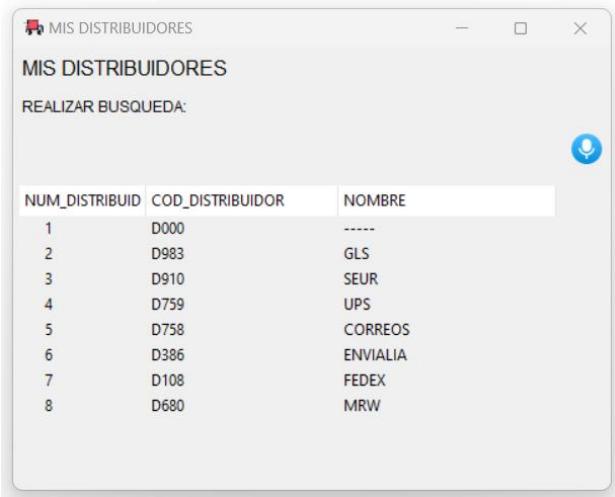
The first window is titled "AÑADIR DISTRIBUIDOR" and has a "NOMBRE:" field containing "GLS". A modal dialog box shows an info icon and the message "EL DISTRIBUIDOR YA SE ENCUENTRA EN EL SISTEMA" with an "Aceptar" button.

The second window is also titled "AÑADIR DISTRIBUIDOR" and has a "NOMBRE:" field containing "GLD". A modal dialog box shows a question mark icon and the message "DATOS: CÓDIGO DE DISTRIBUIDOR: D628 NOMBRE: GLD" with "Sí" and "No" buttons.

Podremos borrar un distribuidor por su código de distribuidor, en caso de borrar el distribuidor, todos sus pedidos serán desvinculados de este distribuidor

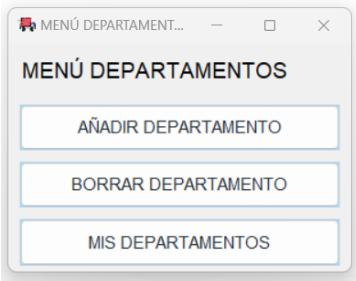


Podemos obtener un listado de nuestros distribuidores y hacer búsqueda mediante texto o voz, en este caso al tener un solo campo (nombre) y no ser modificable, no tendremos la opción de modificar al distribuidor.

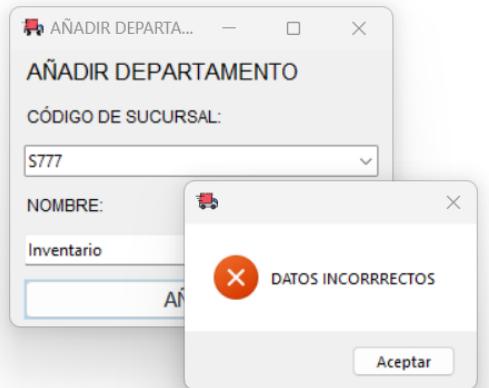


NUM_DISTRIBUID	COD_DISTRIBUIDOR	NOMBRE
1	D000	-----
2	D983	GLS
3	D910	SEUR
4	D759	UPS
5	D758	CORREOS
6	D386	ENVALIA
7	D108	FEDEX
8	D680	MRW

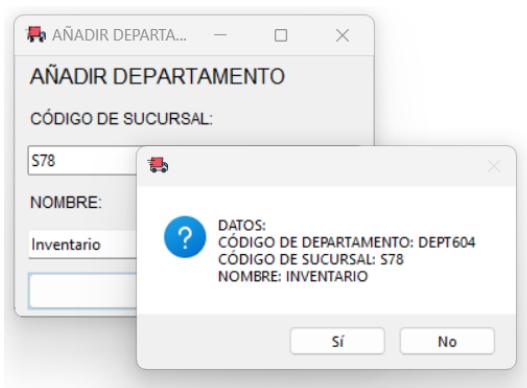
Al clicar la opción de los departamentos al igual que con todos los campos anteriores no aparecerá una ventana con 3 botones



Al añadir un departamento nos saldrán los siguientes campos a rellenar, el código de sucursal (el cual deberá estar dado de alta) y el nombre de ese departamento.

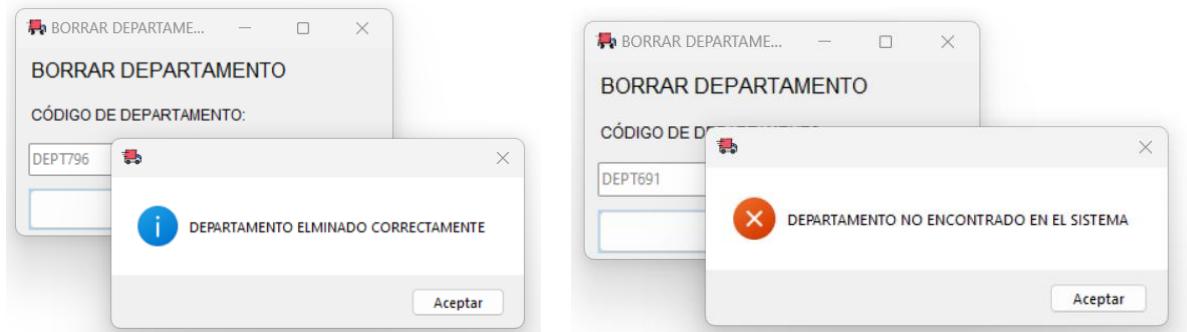


Código de sucursal inválido

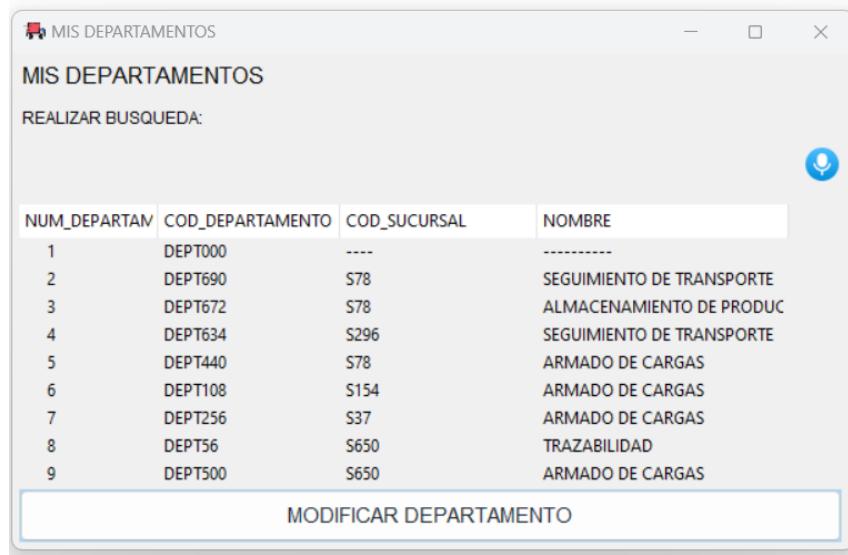


Código de sucursal válido

Borraremos un departamento mediante su código de sucursal



Además podremos poder ver todos nuestros departamentos y hacer búsquedas tanto por texto como por reconocimiento de voz



The screenshot shows a table with columns: NUM\_DEPARTAM, COD\_DEPARTAMENTO, COD\_SUCURSAL, and NOMBRE. The data is as follows:

NUM_DEPARTAM	COD_DEPARTAMENTO	COD_SUCURSAL	NOMBRE
1	DEPT000	----	-----
2	DEPT690	S78	SEGUIMIENTO DE TRANSPORTE
3	DEPT672	S78	ALMACENAMIENTO DE PRODUC
4	DEPT634	S296	SEGUIMIENTO DE TRANSPORTE
5	DEPT440	S78	ARMADO DE CARGAS
6	DEPT108	S154	ARMADO DE CARGAS
7	DEPT256	S37	ARMADO DE CARGAS
8	DEPT56	S650	TRAZABILIDAD
9	DEPT500	S650	ARMADO DE CARGAS

Below the table is a button labeled 'MODIFICAR DEPARTAMENTO'.

Si queremos modificar un departamento deberemos seleccionarlo, clicar 'modificar departamento' y nos aparecerá la siguiente ventana

**MODIFICAR...**

CÓDIGO DE DEPARTAMENTO:

CÓDIGO DE SUCURSAL:

NOMBRE:

**REALIZAR CAMBIOS**

El código de departamento será de solo lectura y las restricciones de cada campo serán las mismas que la ventana de añadir contactos.

El submenú empleados tendrá 4 opciones a diferencia de los demás submenús ya que consideramos que los datos de los empleados los podemos dividir en datos más personales, y datos más dirigidos a la información que debe tener la empresa.

**MENÚ EMPLEADOS**

**MENÚ EMPLEADOS**

AÑADIR EMPLEADO

BORRAR EMPLEADO

GESTIÓN DE LOS EMPLEADOS

DATOS DE LOS EMPLEADOS

La opción de añadir empleado se compondrá de bastantes campos

 AÑADIR EMPLEADO

### AÑADIR EMPLEADO

CÓDIGO DE DEPARTAMENTO:

DNI:

NOMBRE:

APELLIDOS:

FECHA DE NACIMIENTO:

SALARIO:

DOMICILIO:

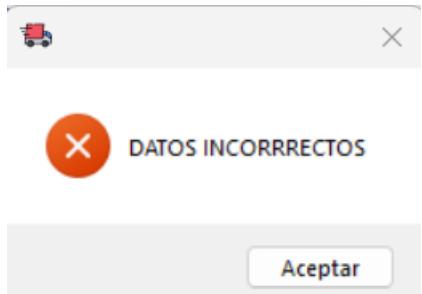
TELÉFONO:

OFICIO:

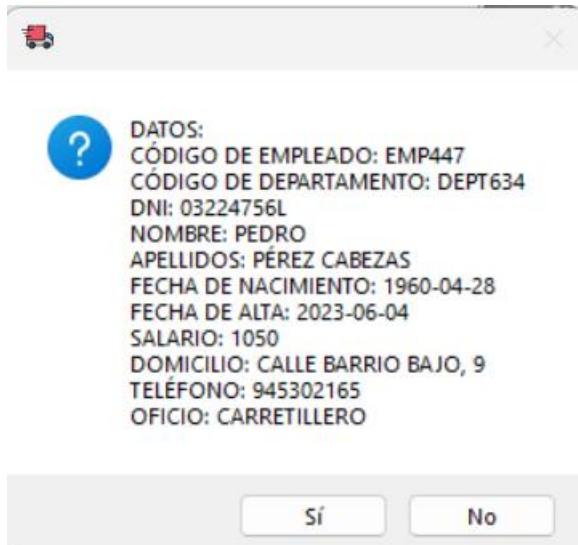
**AÑADIR**

Validaremos sobre todo el código de departamento, si está o no en el sistema, que el DNI tenga 8 números y 1 letra, el formato de la fecha de nacimiento que sea 'yyyy-mm-dd' y que el oficio sea una de los del desplegable:

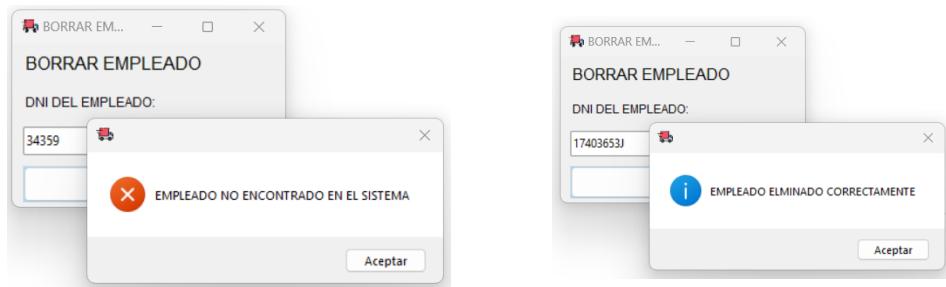
Al igual que con las ventanas anteriores si alguna de las condiciones anteriores no se cumple forzará una ventana de error



Si son correctos me mostrará los datos introducidos y si clico 'sí' se añadirá el empleado al sistema.



Si seleccionamos la ventana de borrado podremos optar a dos opciones la de la izquierda si introducimos un DNI que no está en el sistema, la derecha si eliminamos el empleado correctamente



Tenemos dos vistas de nuestros empleados

MIS EMPLEADOS

REALIZAR BUSQUEDA:

MODIFICAR EMPLEADO

NUM_EMPLEADC	COD_EMPLEADC	COD_DEPARTAN	DNI	FECHA DE ALTA	TELÉFONO	SALARIO	OFICIO
1	EMP1	DEPT690	44373161C	2023-01-26	654580578	1080	GESTOR
2	EMP2	DEPT185	91549689Y	2017-02-02	933837342	1110	ANALISTA
3	EMP3	DEPT000	24741099E	2014-09-11	675715860	920	RR.HH
4	EMP5	DEPT690	39510091R	2013-03-02	638082755	1620	MOZO DE ALMA
5	EMP6	DEPT000	27427023Y	2022-10-30	682036274	1640	RR.HH
6	EMP7	DEPT56	13773995P	2014-07-20	671971348	1440	CARRETTERO
7	EMP8	DEPT56	58977763J	2019-05-15	664961055	1040	TÉCNICO
8	EMP9	DEPT690	61161629Y	2023-03-30	917769007	960	GESTOR
9	EMP10	DEPT56	44466122S	2013-07-12	664928645	1380	CARRETTERO
10	EMP11	DEPT256	25502655W	2020-05-04	603520543	1530	GESTOR

MIS EMPLEADOS

REALIZAR BUSQUEDA:

MODIFICAR EMPLEADO

NUM_EMPLEADC	DNI	NOMBRE	APELLIDOS	FECHA_NAC	DOMICILIO
1	44373161C	JUAN	QUINTANA LUNA	1985-10-26	CALLE SAN JORGE .
2	91549689Y	CARLOS	ALARCÓN CRESPO	1997-08-10	CALLE LILAS 35
3	24741099E	JULIA	FLORES PACEROS	1989-04-27	CALLE CELIO 53
4	39510091R	CARLA	ALEGRIA PONCE	1994-10-17	CALLE RÍO TAJO 68
5	27427023Y	JESÚS	AROCA PAMPAS	1990-09-03	CALLE CAPITOLIO .
6	13773995P	PABLO	CORACHA DÍAZ	1975-07-27	CALLE SANTA MAF
7	58977763J	RUTH	PASTA FLORES	1993-12-03	CALLE NÁPOLES 75
8	61161629Y	MÓNICA	DÍAZ SANTONA	2004-04-16	CALLE LIRIOS 62
9	44466122S	NOÉ	PACHIGUA PACHE	1981-06-19	CALLE FAIAL 51
10	25502655W	SERGIO	JARA OJAL	1983-12-12	CALLE JAZMÍN 46

En ambas ventanas de mis empleados podremos modificar nuestros empleados, con algunos campos de solo lectura, y tendremos las mismas restricciones que la ventana de añadir empleados.

**MODIFICAR EMPLEADO**

CÓDIGO DE EMPLEADO:

CÓDIGO DE DEPARTAMENTO:

**MODIFICAR EMPLEADO**

DNI:

NOMBRE:

TELÉFONO:

APELLIDOS:

SALARIO:

FECHA DE NACIMIENTO:

OFICIO:

DOMICILIO:

Por último, tendremos en el menú principal una que nos guarda los datos

**HMN LOGISTICS**

**¿QUÉ DESEA GESTIONAR?**



ARTÍCULOS	
SUCURSALES	
DISTRIBUIDORES	
EMPLEADOS	
<b>GUARDAR Y SALIR</b>	

**?** DESEAN GUARDAR LOS CAMBIOS

La última opción nos permitirá guardar los datos y salir del sistema



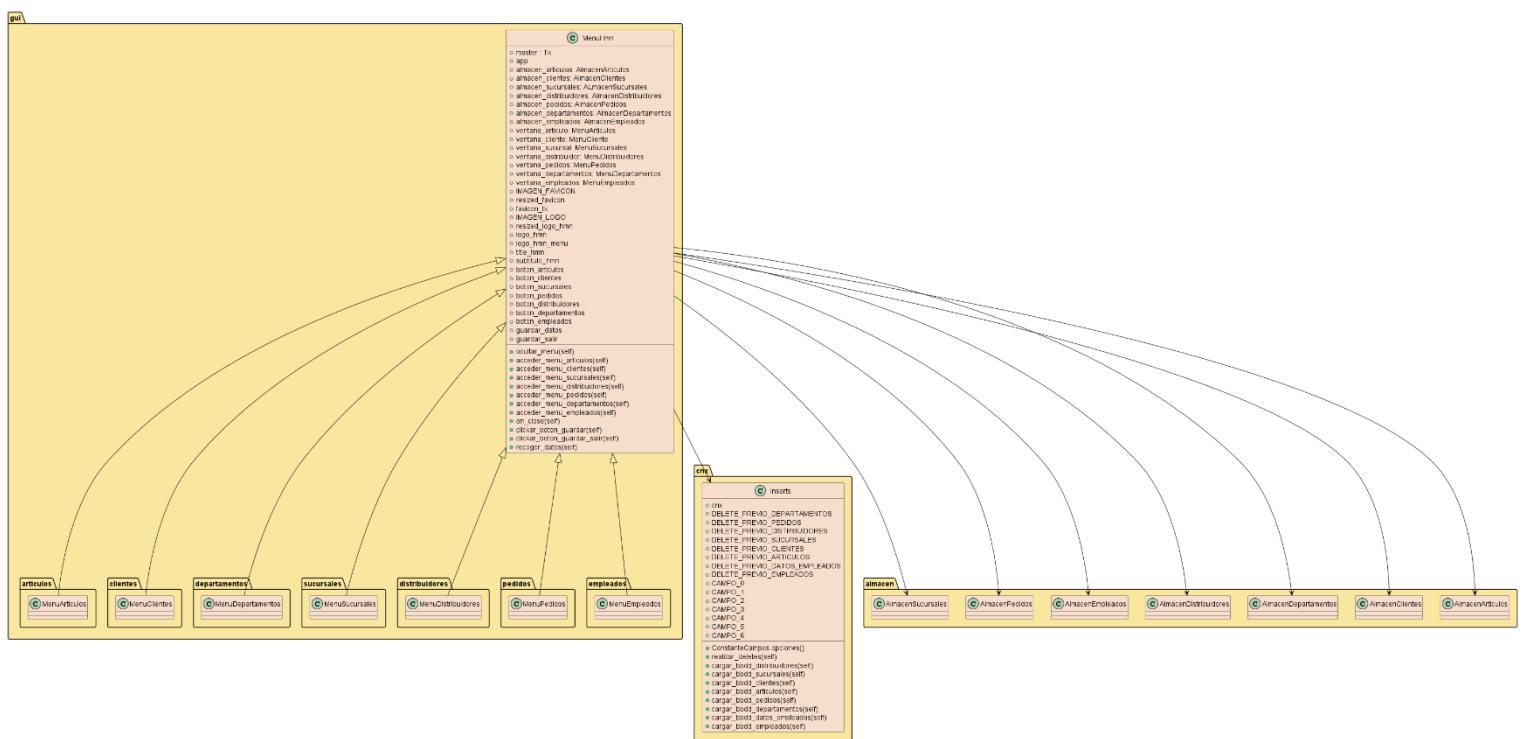
## 2. DISEÑO

Para hacer esta aplicación hemos decidido que la mejor estructura de diseño sería la POO, ya que cada clase sería una ventana, cada clase sería un almacén donde guardar los datos de nuestra empresa y cada clase crearía nuevos objetos de cada botón, nuevos artículos, nuevas sucursales...

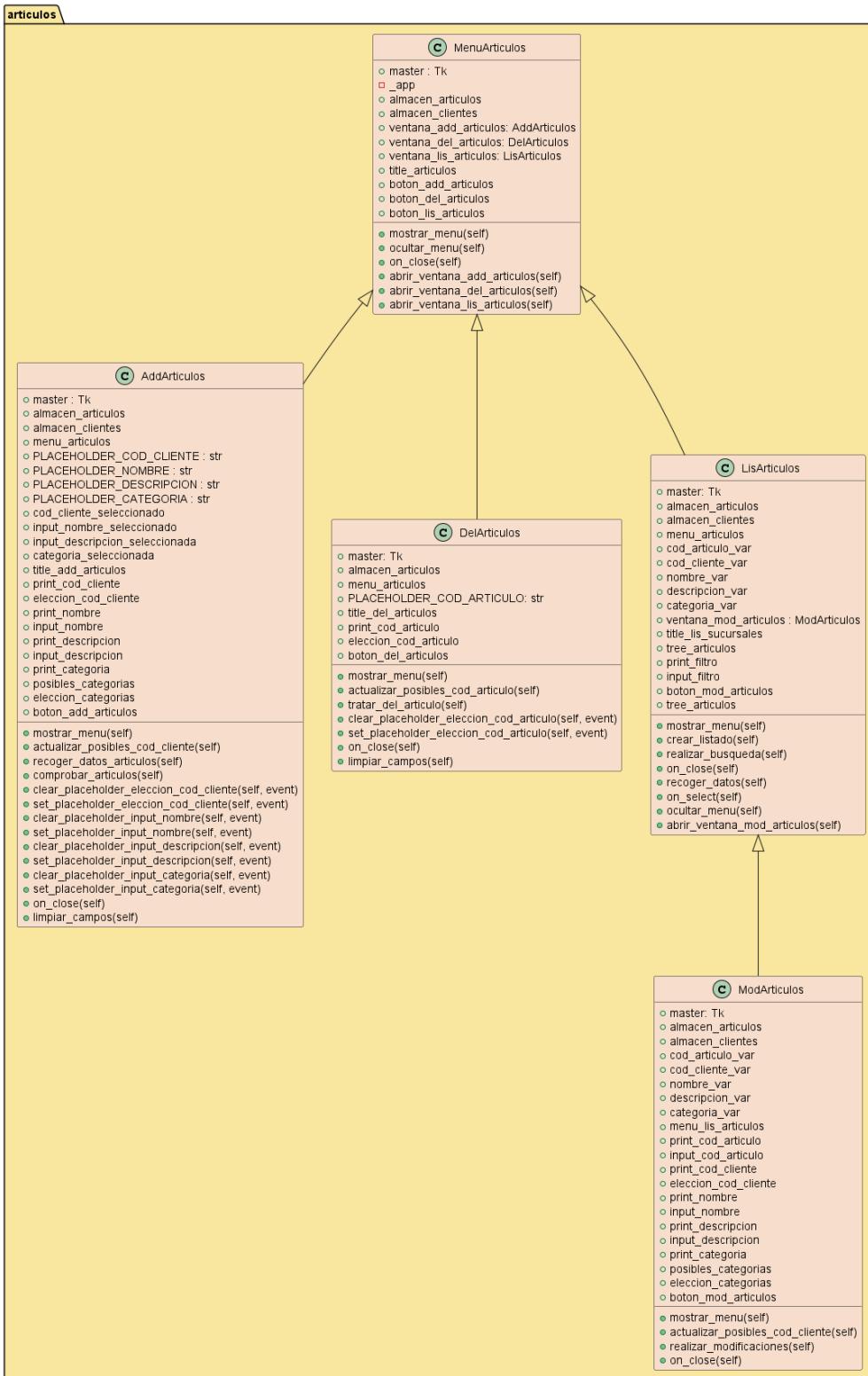
## 2.1 Diagrama de clases

Al tener muchas clases hemos decidido crear varios diagramas de clases e ir correlacionándolos respectivamente.

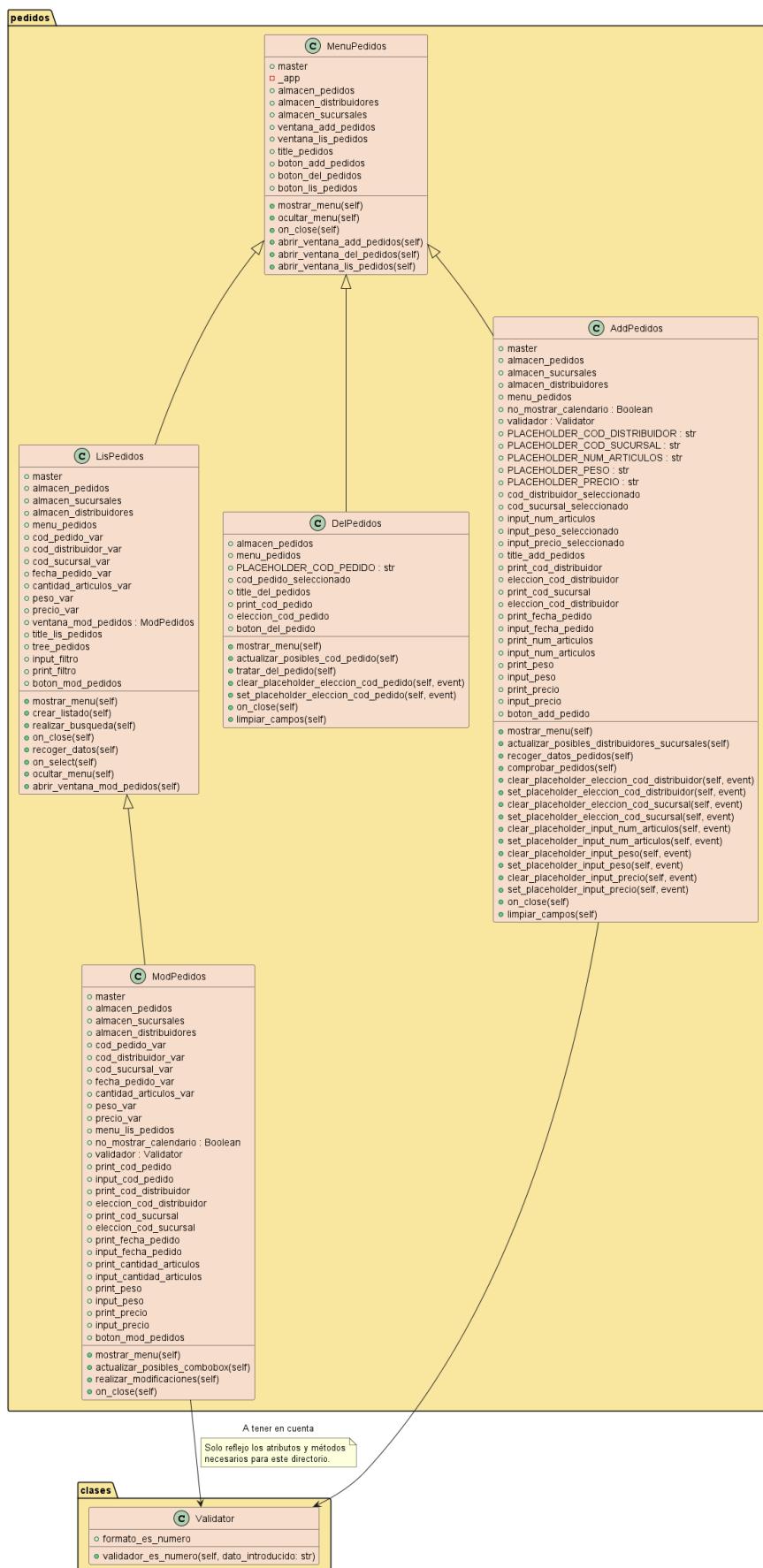
#### ■ DIAGRAMA DEL MENÚ PRINCIPAL



## ■ DIAGRAMA DE ARTÍCULOS

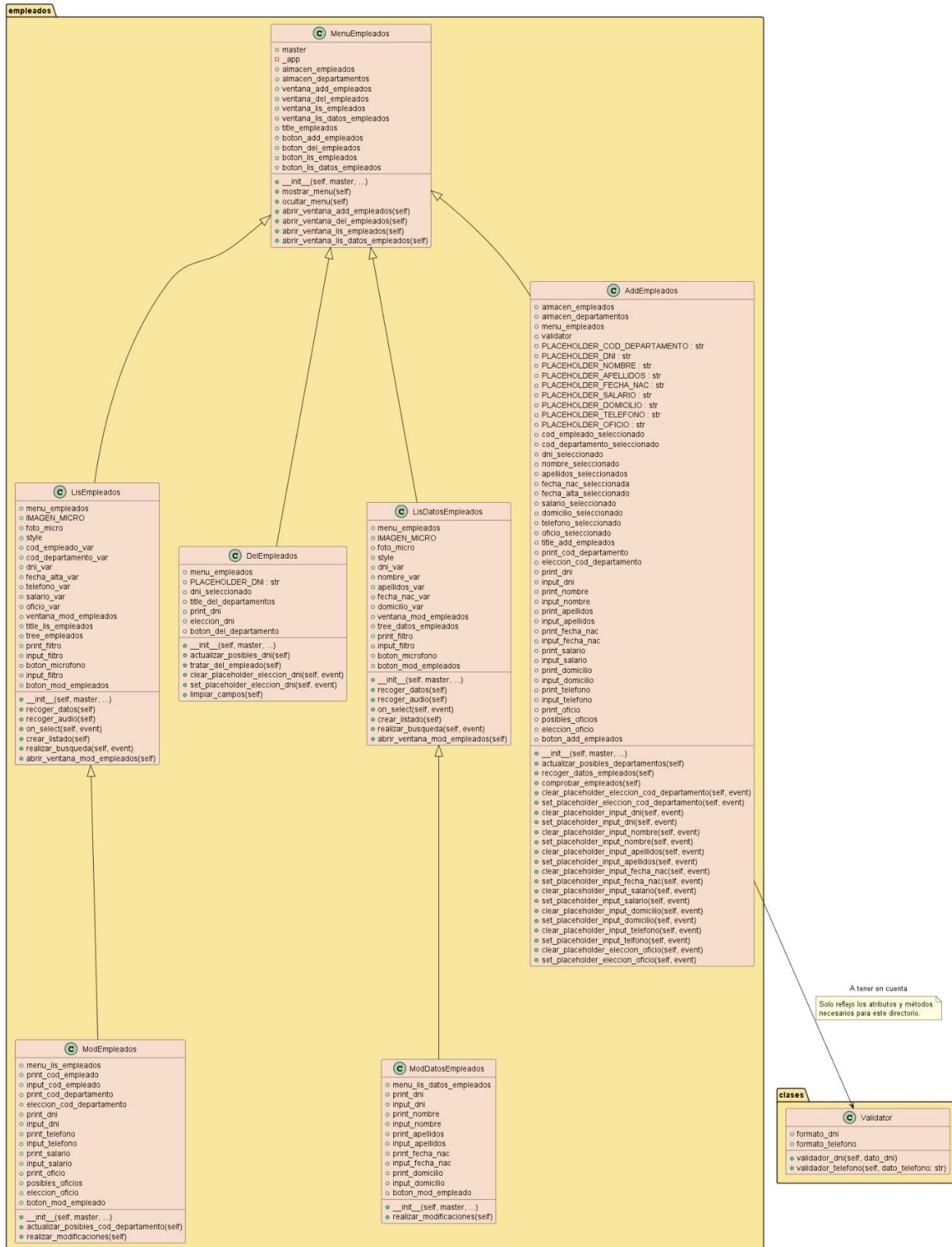


## ■ DIAGRAMA DE PEDIDOS

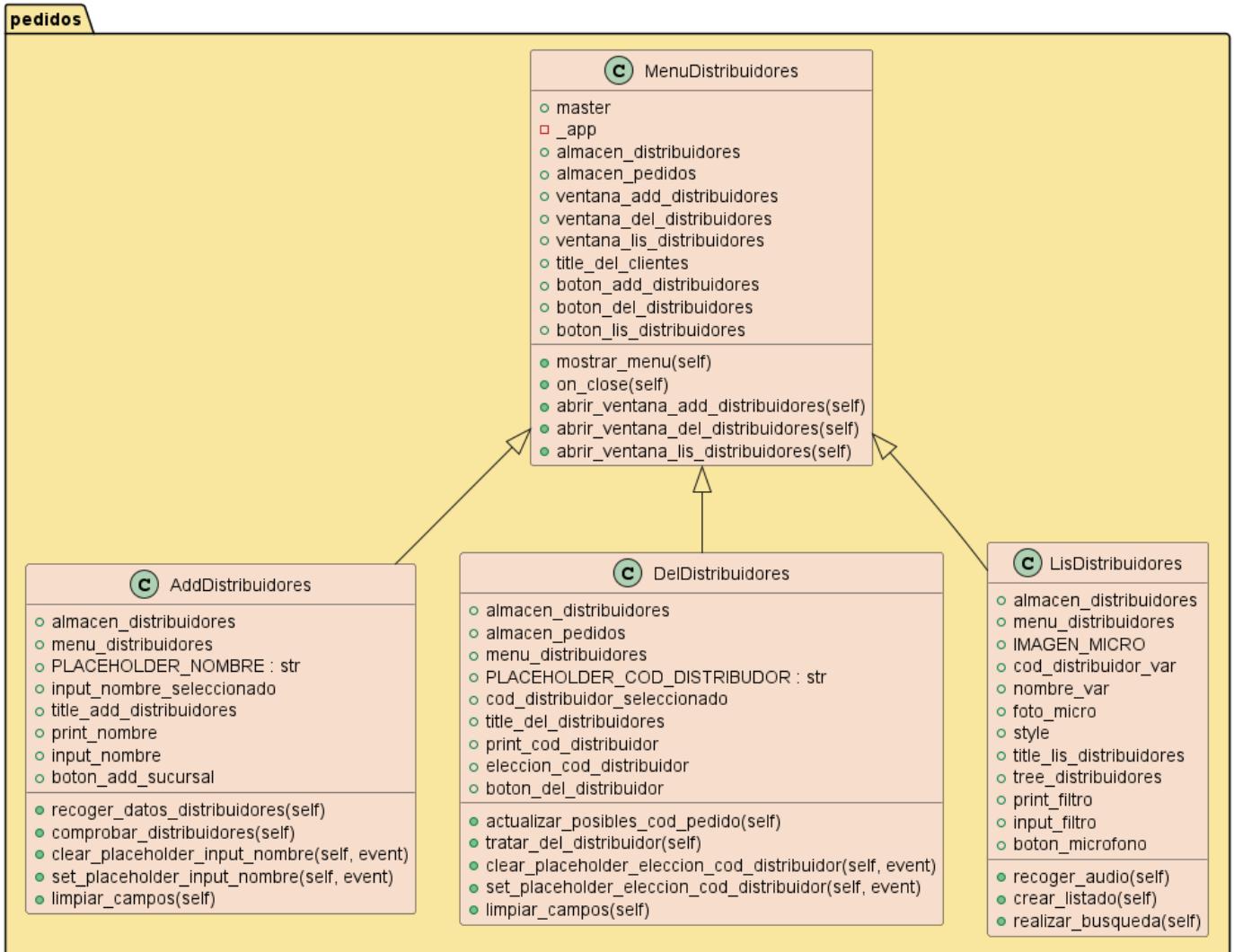




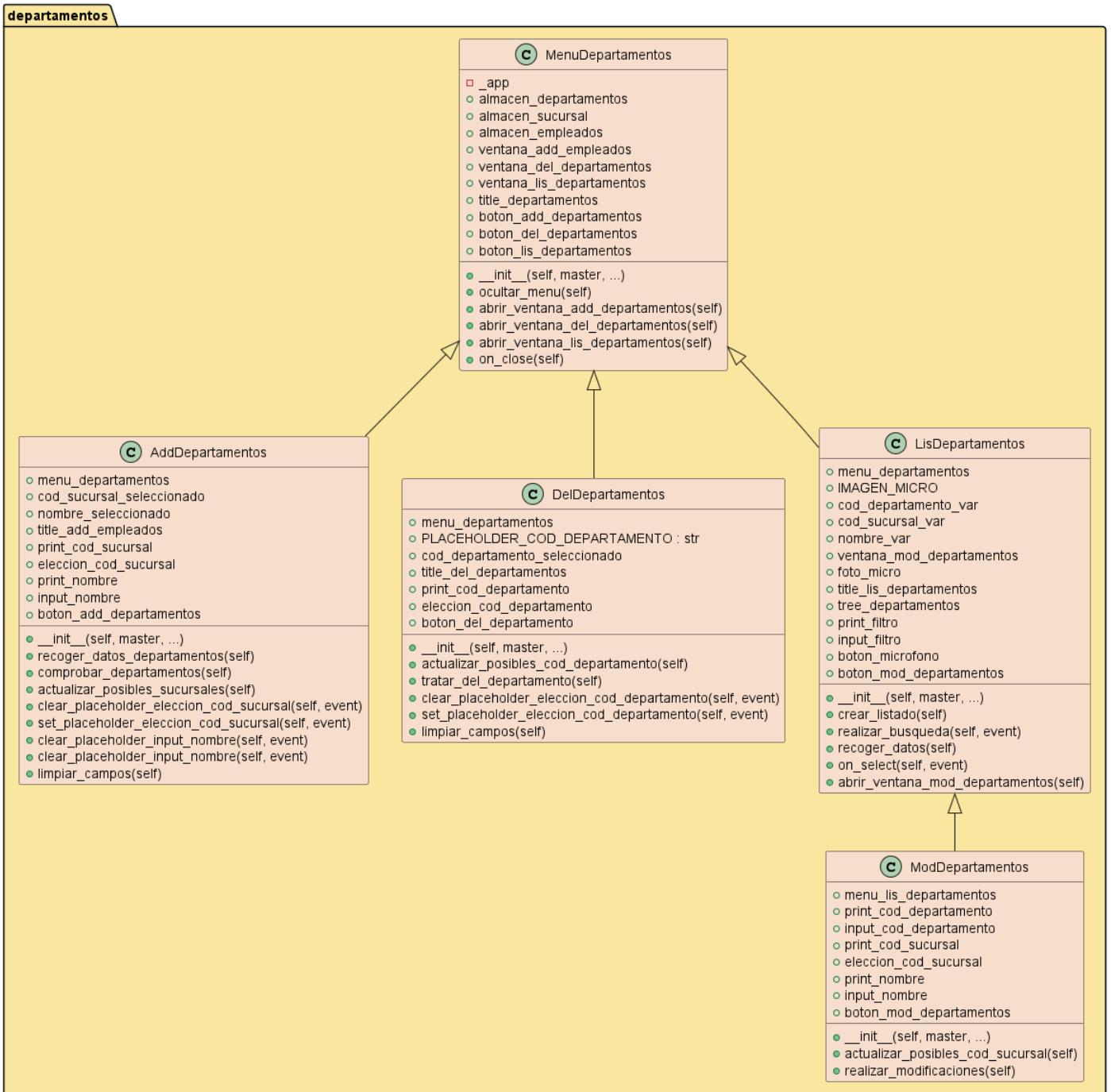
## ■ DIAGRAMA DE EMPLEADOS



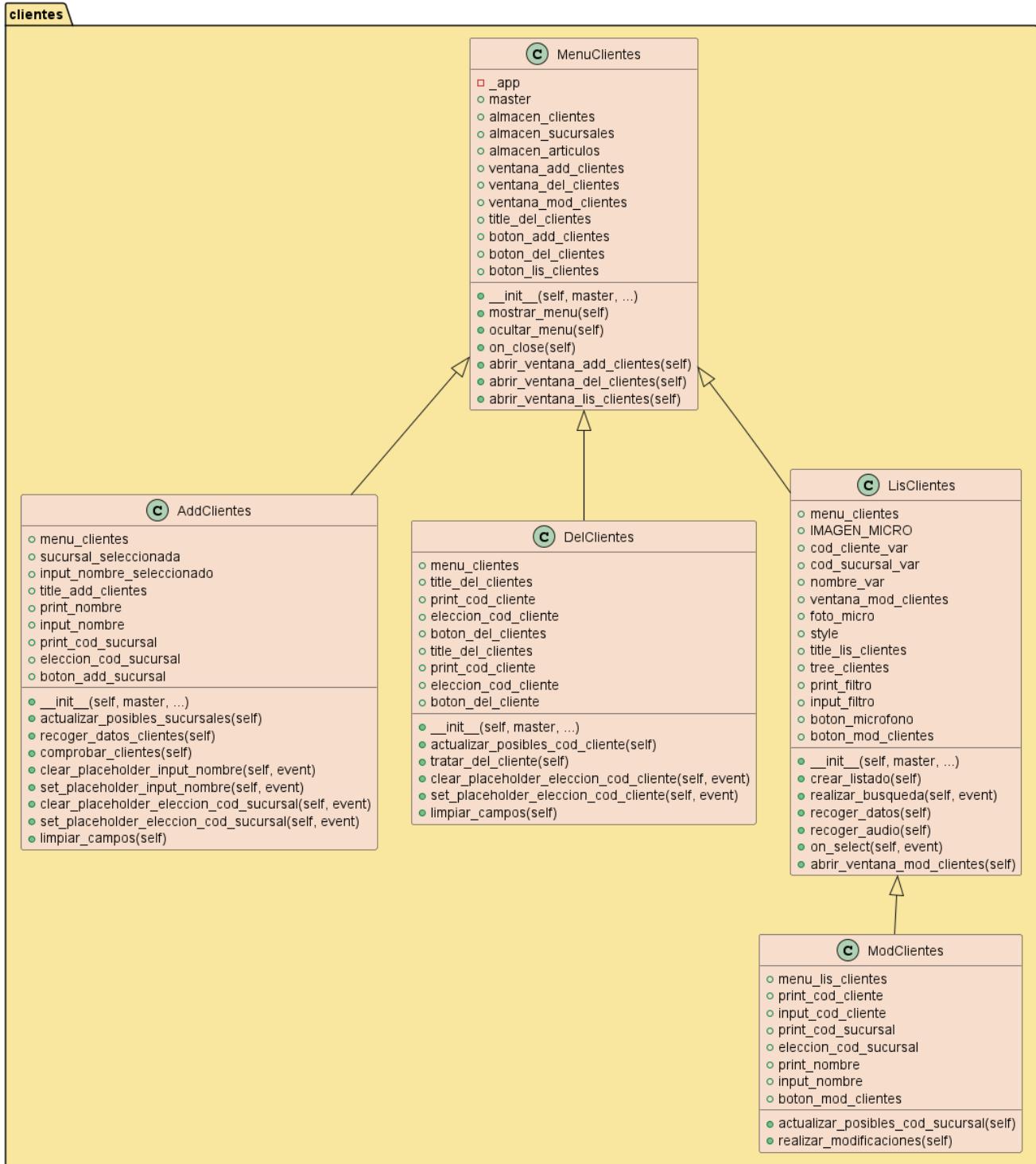
## ■ DIAGRAMA DE DISTRIBUIDORES



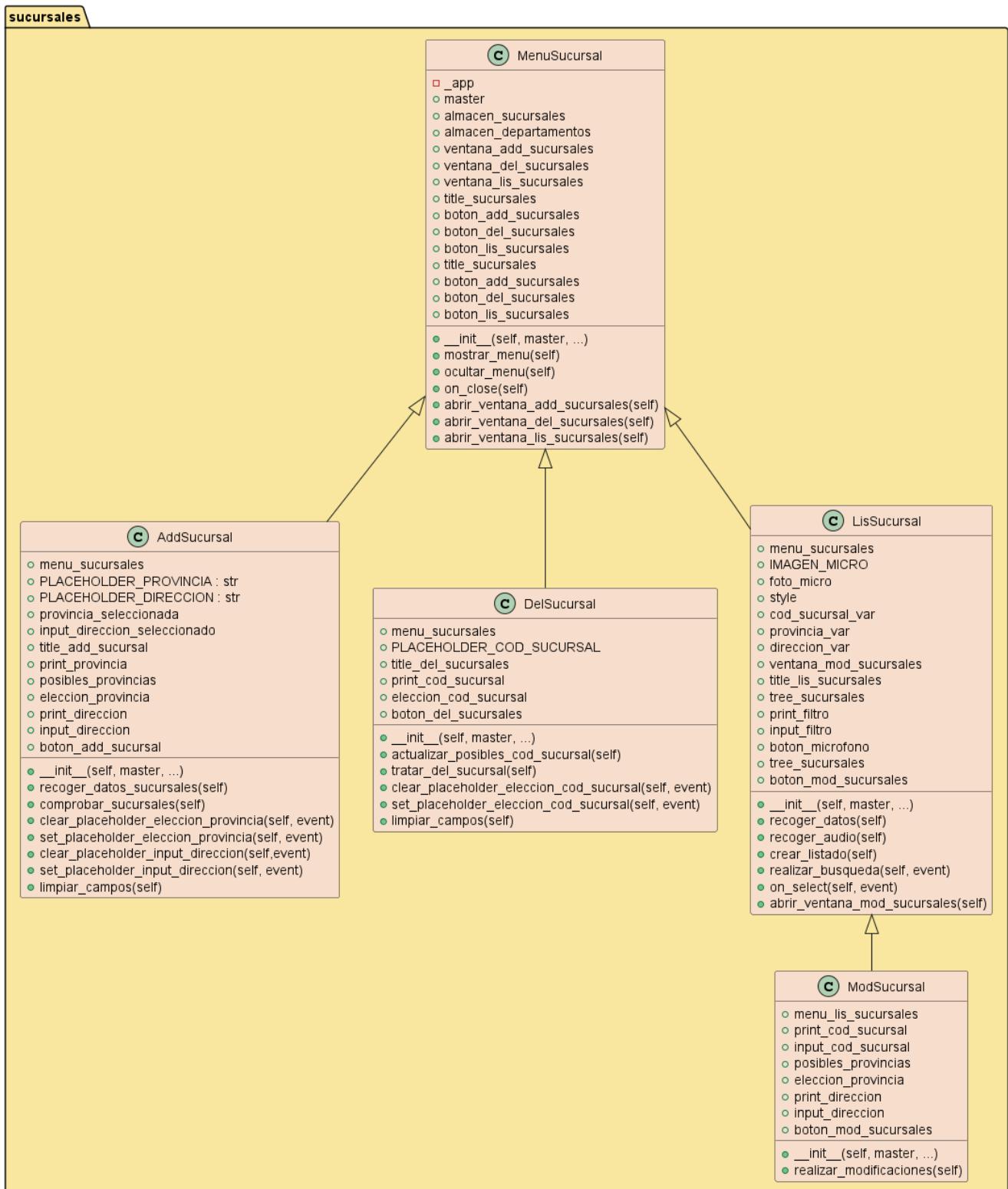
## ■ DIAGRAMA DE DEPARTAMENTOS



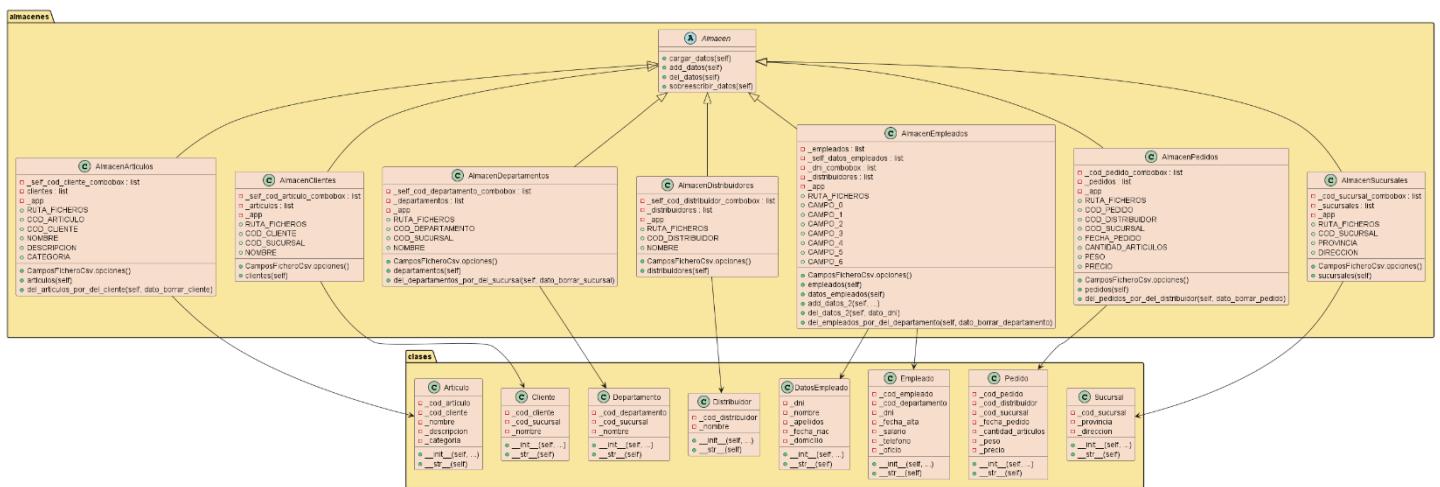
## ■ DIAGRAMA DE CLIENTES



## ■ DIAGRAMA DE SUCURSALES

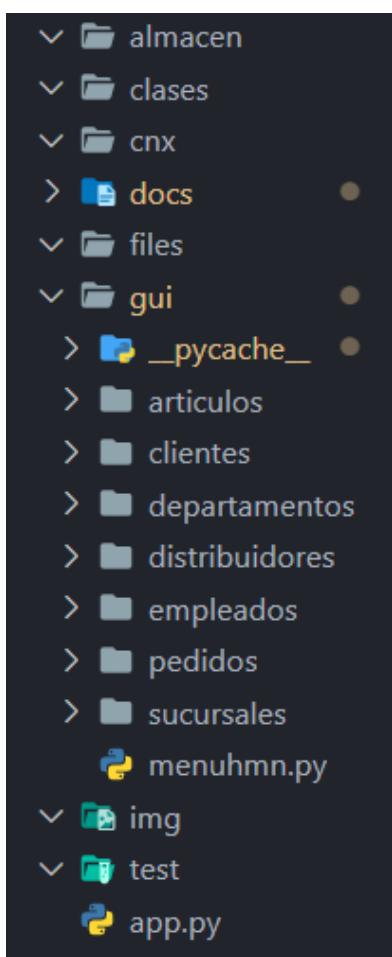


## ■ DIAGRAMA DE ALMACENES



## 2.2 Estructura de Almacenamiento

Organizaremos nuestra aplicación de la siguiente forma:



En la carpeta '**almacen**' guardaremos todo lo relacionado con la modificación de datos, el sistema internamente borrará y añadirá las líneas de ficheros correspondientes, en '**clases**' guardaremos todos las clases para crear nuestros objetos, empleado, sucursal, artículo..., en **cnx** generaremos todo los ficheros que nos permitan conectarnos a nuestro SGBD y que cada vez que guardemos nuestros datos en la aplicación se vean los cambios reflejados en nuestra base de datos. En la carpeta **docs** generaremos todo lo relacionado con la documentación de nuestra aplicación, nuestro pdf que documente toda la aplicación. Las imágenes (en **img**) de cada diagrama de clases y los casos de uso. Y por último en **puml** todos los diagramas generados con PlantUML.

En **files** guardaremos todos los ficheros que contengan los datos de todos nuestros objetos almacenados y cada vez que el programa arranque y cada vez que lo cerremos que se generen cambios en ellos. En el directorio **gui** tendremos todos los ficheros relacionados con la parte gráfica de nuestra aplicación. Tendremos el menuhmn que hará de pantalla de bienvenida a la aplicación y todos los movimientos que pueda hacer el usuario de cada objeto estarán almacenados en sus directorios correspondientes, tenemos otra carpeta **img**

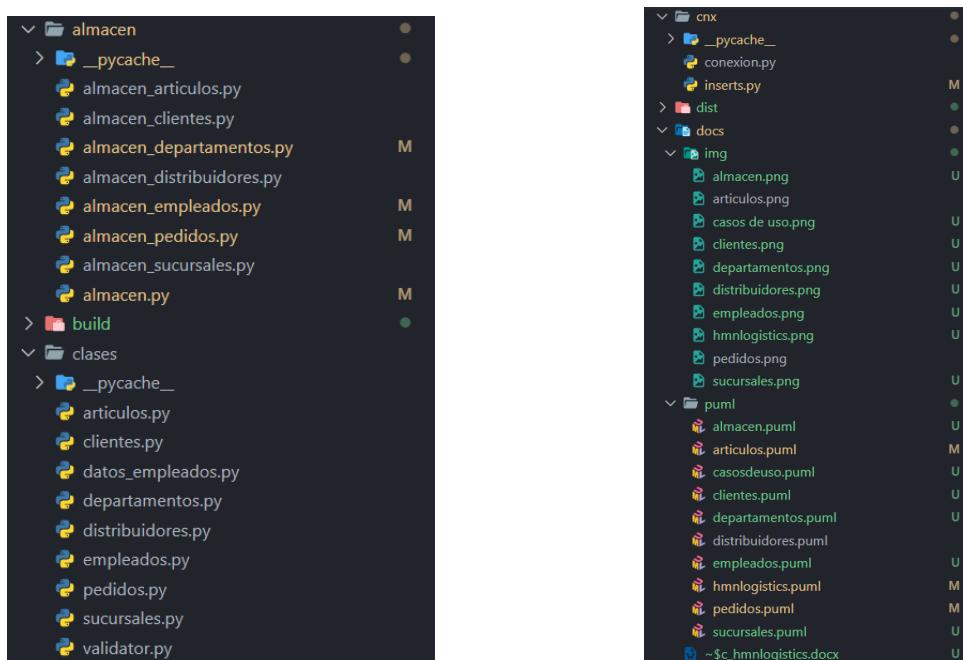
Donde guardaremos imágenes implementadas en nuestra interfaz, como por ejemplo un favicon que aparecerá en todas las ventanas y una imagen que aparecerá en nuestra ventana de bienvenida, entre otras, además para poder hacer pruebas a nuestra aplicación generaremos un directorio **test** y con la librería unittest generaremos todas esas pruebas para ver el correcto funcionamiento de nuestra aplicación de HMNLogistics.

Y por último. Para arrancar nuestra aplicación deberemos de tener un fichero **app.py** que nos permita poner en un funcionamiento la app. Para poder arrancar la aplicación desde el escritorio con un acceso directo deberemos de indexar un .exe con la librería pyinstaller al nivel de este fichero, para que recoja todas las rutas de todos los ficheros de manera correcta, además con el .exe se nos generar dos directorios extra que los sostengan, que son el directorio **build** y el directorio **dist**.

### 3. CODIFICACIÓN

Para la codificación utilizaremos un entorno de desarrollo, en este caso Visual Studio Code para facilitar el desarrollo de la aplicación, y utilizaremos el lenguaje de programación Python con el que gestionaremos el funcionamiento de la app, creando ficheros “.csv” para guardar los datos y conectándonos en localhost a un SGBD, en este caso MySQL. Para codificar como hemos mencionado anteriormente usaremos la POO. Para generar las interfaces hemos utilizado la librería ‘Tkinter’, y para el reconocimiento de voz ‘Speech Recognition’.

Además hemos añadido un .exe que nos permite la ejecución desde el escritorio (mediante un acceso directo). Tras finalizar todo el proceso de codificación, nos quedó la siguiente estructura de directorios y ficheros:



```

    └── files
        ├── articulos.csv
        ├── clientes.csv
        ├── datos_empleados.csv
        ├── departamentos.csv
        ├── distribuidores.csv
        ├── empleados.csv
        ├── pedidos.csv
        └── sucursales.csv

    └── gui
        >   └── __pycache__
        └── articulos
            >   └── __pycache__
            ├── menu_add_articulos.py
            ├── menu_articulos.py
            ├── menu_del_articulos.py
            ├── menu_lis_articulos.py
            └── menu_mod_articulos.py

        └── clientes
            >   └── __pycache__
            ├── menu_add_clientes.py
            ├── menu_clientes.py
            ├── menu_del_clientes.py
            ├── menu_lis_clientes.py
            └── menu_mod_clientes.py

```

```

    └── departamentos
        >   └── __pycache__
        ├── menu_add_departamentos.py
        ├── menu_del_departamentos.py
        ├── menu_departamentos.py
        ├── menu_lis_departamentos.py
        └── menu_mod_departamentos.py

    └── distribuidores
        >   └── __pycache__
        ├── menu_add_distribuidores.py
        ├── menu_del_distribuidores.py
        ├── menu_distribuidores.py
        └── menu_lis_distribuidores.py

    └── empleados
        >   └── __pycache__
        ├── menu_add_empleados.py
        ├── menu_del_empleados.py
        ├── menu_empleados.py
        ├── menu_lis_datos_empleados.py
        ├── menu_lis_empleados.py
        └── menu_mod_datos_empleados.py

```

```

    └── pedidos
        >   └── __pycache__
        ├── menu_add_pedidos.py
        ├── menu_del_pedidos.py
        ├── menu_lis_pedidos.py
        ├── menu_mod_pedidos.py
        └── menu_pedidos.py

    └── sucursales
        >   └── __pycache__
        ├── menu_add_sucursales.py
        ├── menu_del_sucursales.py
        ├── menu_lis_sucursales.py
        ├── menu_mod_sucursales.py
        └── menu_sucursales.py

        └── menuhmn.py

    └── img
        ├── carretilla.ico
        ├── hmnlOGISTICS.jpeg
        ├── logo.png
        ├── logoescritorio.ico
        ├── logohmncolor.png
        └── microfono.png

```

```

    └── test
        └── test_almacenes
            ├── test_almacen_articulos.py
            ├── test_almacen_clientes.py
            ├── test_almacen_departamentos.py
            ├── test_almacen_distribuidores.py
            ├── test_almacen_empleados.py
            ├── test_almacen_pedidos.py
            └── test_almacen_sucursales.py

        └── test_clases
            ├── test_articulos.py
            ├── test_clientes.py
            ├── test_datos_empleados.py
            ├── test_departamentos.py
            ├── test_distribuidores.py
            ├── test_empleados.py
            ├── test_pedidos.py
            ├── test_sucursales.py
            └── test_validator.py

            └── app.exe
            ├── app.py
            └── app.spec

            └── borradores.txt
            └── condiciones.txt

            └── README.md

```

## 4. PRUEBAS

Para realizar las pruebas hemos utilizado la librería estándar de Python ‘unittest’ que nos permite hacer test unitarios, al tener muchos test pero de forma similar vamos a explicar los test relacionados con los artículos. Ya que con esta explicación englobamos a todos los elementos, además también explicaremos el validador que hemos utilizado para comprobar los datos introducidos por el usuario:

```
class TestArticulo(unittest.TestCase):
    def test_crear(self):
        articulo = Articulo()
        self.assertEqual(articulo._cod_articulo, "A302")
        self.assertEqual(articulo._cod_cliente, "C343")
        self.assertEqual(articulo._nombre, "Balón de Joma")
        self.assertEqual(articulo._descripcion, "Balón de Joma 2022")
        self.assertEqual(articulo._categoria, "DEPORTES")

    def test_str(self):
        articulo = Articulo("A302", "C343", "Balón de Joma",
                            "Balón de Joma 2022", "DEPORTES")
        self.assertEqual("A302;C343;Balón de Joma;Balón de Joma
2022;DEPORTES", str(articulo))
```

En este caso para probar que un artículo sea válido, en la función `test_crear`, creamos un artículo con todos sus atributos y en `test_str` probamos mediante los atributos de la primera función si funciona nuestro `__str__(self)` de la clase `Artículo` que es el formato que generaremos para los ficheros.

```
import unittest

from almacen.almacen_articulos import AlmacenArticulos
from clases.articulos import Articulo

class TestAlmacenArticulos(unittest.TestCase):
    def test_add_articulo(self):
        #CAMBIAMOS EL CÓDIGO DE ARTÍCULO QUE ES LO QUE DETERMINA LA
        #IGUALDAD DEL ARTÍCULO
        articulo1 = Articulo("A567", "C783", "AJEDREZ NICANORA", "AJEDREZ
DE LA 3º EDICIÓN", "OCIO")
        articulo2 = Articulo("A432", "C783", "AJEDREZ NICANORA", "AJEDREZ
DE LA 3º EDICIÓN", "OCIO")
        articulo3 = Articulo("A934", "C783", "AJEDREZ NICANORA", "AJEDREZ
DE LA 3º EDICIÓN", "OCIO")
        articulo4 = Articulo("A432", "C783", "AJEDREZ NICANORA", "AJEDREZ
DE LA 3º EDICIÓN", "OCIO")
```

```

arrayarticulos = [articulo1, articulo2, articulo3]

self.assertTrue(articulo1 in arrayarticulos)
self.assertFalse(articulo4 in arrayarticulos)

def test_del_articulo(self):
    articulo1 = Articulo("A567", "C783", "AJEDREZ NICANORA", "AJEDREZ
DE LA 3º EDICIÓN", "OCIO")
    articulo2 = Articulo("A432", "C783", "AJEDREZ NICANORA", "AJEDREZ
DE LA 3º EDICIÓN", "OCIO")
    articulo3 = Articulo("A934", "C783", "AJEDREZ NICANORA", "AJEDREZ
DE LA 3º EDICIÓN", "OCIO")
    articulo4 = Articulo("A678", "C783", "AJEDREZ NICANORA", "AJEDREZ
DE LA 3º EDICIÓN", "OCIO")

    articulo_buscado1 = "A567"
    articulo_buscado2 = "A566"

    arrayarticulo_buscado = [articulo1, articulo2, articulo3,
articulo4]
    self.assertTrue(articulo_buscado1 in arraydefecto)
    self.assertFalse(articulo_buscado2 in arraydefecto)

```

En este caso, testeamos el Almacen de artículos las adiciones y borramos de artículos (en caso de modificar no necesitamos un función `test_modificar`, ya que borramos artículo y luego lo añadimos con las modificaciones que envía el usuario al sistema)

La primera función crea una lista de artículos que se encuentran en el sistema y hace 2 pruebas una en la que añade un objeto ya existente en el sistema y por tanto este le evita añadir un Artículo con un id de artículo ya dado de alta y en el segundo añadimos un nuevo Artículo por tanto se añade sin ningún problema.

La segunda función borra artículos mediante su id de artículo, esta función genera una lista de 4 artículos y mediante 2 ides de artículos comprueba si están o no en el sistema, una no está por tanto no nos dejaría hacer un borrado y otro si que se encuentra dado de alta por lo tanto nos dejaría borrarlo sin ningún problema.

Como ya dijimos anteriormente sería un poco recursivo explicar todos los elementos ya que funcionan de manera similar así que por último explicaremos las pruebas que hemos realizado al validador.

```

import unittest

from clases.validator import Validator

class TestValidator(unittest.TestCase):
    def test_validador_fecha(self):
        #VALIDA UNA FECHA POSTERIOR A LA ACTUAL CON FORMATO 'YYYY-MM-DD'
        fecha1 = '2030-03-12'

```

```

fecha2 = '2020-03-12'
fecha3 = 2020
self.assertTrue(fecha1)
self.assertFalse(fecha2)
self.assertFalse(fecha3)

def test_validador_dni(self):
    dni1 = '03222782E'
    dni2 = '03222782EB'
    dni3 = 32227829
    self.assertTrue(dni1)
    self.assertFalse(dni2)
    self.assertFalse(dni3)

def test_validador_telefono(self):
    telefono1 = 934201342
    telefono2 = 74862598665868 #+9 Números
    telefono3 = '934567823' #Telefono debe de ser de tipo 'int'
    telefono4 = 'bhfiuewhig'
    self.assertTrue(telefono1)
    self.assertFalse(telefono2)
    self.assertFalse(telefono3)
    self.assertFalse(telefono4)

def test_validador_es_numero(self):
    #ACEPTA TANTO 'INT' COMO 'FLOAT'
    es_numero1 = 354553
    es_numero2 = 3.45
    es_numero3 = '3.45'
    es_numero4 = 'fkjewbvkew'
    self.assertTrue(es_numero1)
    self.assertTrue(es_numero2)
    self.assertFalse(es_numero3)
    self.assertFalse(es_numero4)

```

La clases TestValidator valida las 4 funciones que tenemos implementadas en clase Validator original.

La primera función valida el formato de la fecha en este caso utilizamos el formato 'YYYY-MM-DD' y que la fecha posterior sea posterior a la del sistema, hemos hecho 3 pruebas, una válida y dos falsas (una debido a que la fecha es anterior a la del sistema, y otra porque es de tipo numérico)

La segunda función valida el formato del dni (es decir, que el DNI, contenga 8 números y 1 letra, hemos realizado 3 pruebas una válida y dos inválidas (una que contiene 2 letras en vez de 1, y una que es de tipo numérico).

La 3º función valida el formato del teléfono, básicamente el teléfono del usuario tiene que tener 9 dígitos y ser de tipo ‘int’, hemos realizado hasta 4 pruebas, 1 correcta y 3 incorrectas (la primera porque contiene más de 9 dígitos, otra que contiene 9 dígitos pero en este caso el tipo de dato es ‘string’ con lo cual tampoco nos sirve, y la última prueba contiene letras).

La última función nos valida que un dato sea o de tipo ‘int’ o ‘float’ es decir o un número entero o un número decimal (esta función por ejemplo la hemos usado para validar, el peso y el precio de nuestros pedidos), hemos hecho hasta 4 pruebas, 2 válidas con un número entero y otro decimal, y 2 inválidas, una de tipo ‘string’ y otra que contiene letras.