

Universidad Tecnológica Nacional
Facultad Regional Avellaneda



Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

Materia: Laboratorio de Computación II

Apellido: *Laporte* Legajo: *111655*

Nombre: *Marcos* División: *2°A*

Flujo del programa

El programa se basa todo en el Form principal, desde este podemos leer datos, abrir las distintas ventanas y realizar acciones con respecto a los clientes y las ventas.

Inicio

Ventas:				Clientes:					
	Id	Precio	Senia	DniCliente	Nombre	Apellido	Telefono	Dni	Debe
▶	1	4969,5703	3500	29557358	Anakin	Skywalker	165161516	2301045	0
	2	8633,12	8633,12	45038419	Peter	Parker	4781583080	28818884	0
	3	1296,5701	500	45421268	Natasha	Romanoff	6623469230	29557358	1469,5703
					Loki	Odinson	7758642509	30234289	0
					The Mighty	Thor	1132434159	42587412	0
					Obi Wan	Kenobi	12356987	44578963	0
					Anthony	Stark	6375125832	45038419	0
					Clint	Barton	9496175297	45421268	796,57007
					Bruce	Banner	9133370598	49827708	0
					Qui Gon	Jinn	918917891	51951981	0
					Steve	Rogers	5345408117	58135220	0
					Stephen	Strange	7363465932	83150463	0

Agregar pedido Modificar cliente Pagar pendiente Agregar cliente Eliminar cliente

Ya hay algunos datos cargados dentro del programa.

Desde el **Inicio** podemos realizar las siguientes acciones:

Agregar Cliente

Al presionar este botón, se abrirá una ventana con 4 campos para completar con datos del usuario, seguido de dos botones *Agregar* y *Cancelar*. Ninguno de estos campos puede quedar vacío, y deben seguir las siguientes reglas:

- **Nombre:** Sólo letras y espacios.
- **Apellido:** Sólo letras y espacios.
- **Teléfono:** Entre 3 y 15 números.
- **DNI:** Entre 7 y 9 números.

Datos del cliente

Nombre:

Apellido:

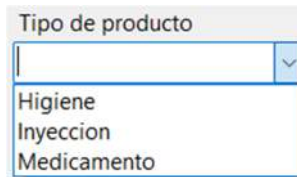
Teléfono:

DNI:

Agregar pedido

Para agregar un pedido tenemos unos pasos a seguir:

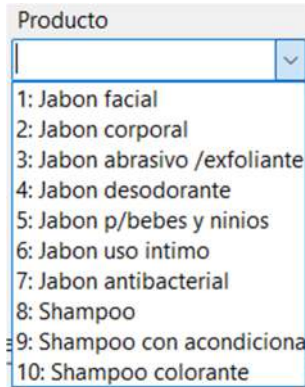
1. Seleccionamos un cliente desde la lista (no podemos realizar un pedido si no hay clientes)
2. Seleccionamos uno de los tres **tipos de productos** en la *primera lista desplegable*.



Tipo de producto

Higiene
Inyeccion
Medicamento

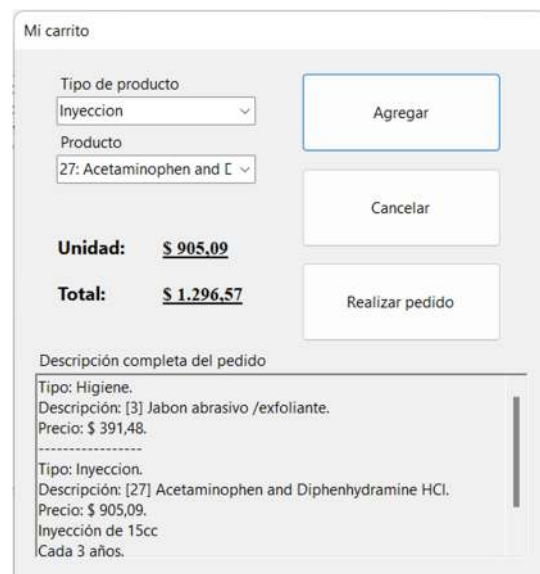
3. Seleccionamos el **producto** que queremos en la *segunda lista desplegable*.



Producto

1: Jabon facial
2: Jabon corporal
3: Jabon abrasivo /exfoliante
4: Jabon desodorante
5: Jabon p/bebes y ninios
6: Jabon uso intimo
7: Jabon antibacterial
8: Shampoo
9: Shampoo con acondiciona
10: Shampoo colorante

4. Una vez seleccionado, abajo nos muestra el precio del producto, junto a su descripción, si decidimos agregarlo, presionamos el botón *Agregar*.
5. Si nos arrepentimos del pedido podemos cancelarlo (*Cancelar*), o si queremos seguir apretamos *Realizar pedido*.



Mi carrito

Tipo de producto: Inyeccion

Producto: 27: Acetaminophen and C

Unidad: \$ 905,09

Total: \$ 1.296,57

Descripción completa del pedido

Tipo: Higiene.
Descripción: [3] Jabon abrasivo /exfoliante.
Precio: \$ 391,48.

Tipo: Inyeccion.
Descripción: [27] Acetaminophen and Diphenhydramine HCl.
Precio: \$ 905,09.
Inyección de 15cc
Cada 3 años.

Agregar

Cancelar

Realizar pedido

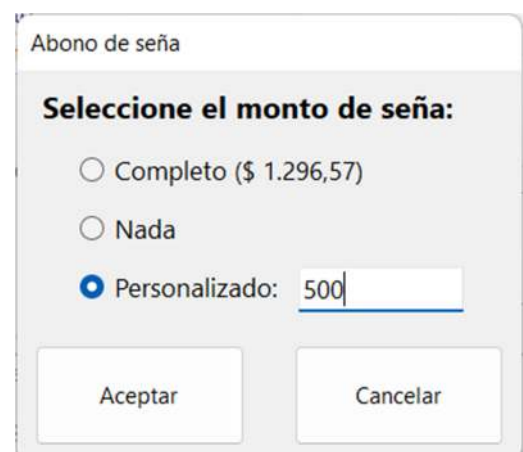
Abonar seña

Dentro de este menú tenemos tres opciones para abonar una seña:

El campo **Completo**, es seguido por el precio total de la venta.

El valor en el campo **Personalizado**, no puede ser negativo ni mayor al valor total. Y si el monto es menor al 5% del valor total de la compra, este será denegado y tomado como 0.

Luego de seleccionar una opción, *Aceptamos* o *Cancelamos*.



Abono de seña

Seleccione el monto de seña:

☐ Completo (\$ 1.296,57)

☐ Nada

☒ Personalizado: 500

Aceptar

Cancelar

Vuelta al inicio

De esta forma, ya tenemos una nueva venta asociada al DNI de un **Ciente**.

Una vez agregamos el pedido, podemos ver que se agrega a lista de la izquierda.

Mi Farmacia				
Ventas:				Clientes:
	Id	Precio	Senia	DniCliente
▶	1	4969,5703	3500	29557358
	2	8633,12	8633,12	45038419
	3	1296,5701	500	45421268

	Nombre	Apellido	Telefono	Dni	Debe
▶	Anakin	Skywalker	165161516	2301045	0
	Peter	Parker	4781583080	28818884	0
	Natasha	Romanoff	6623469230	29557358	1469,5703
	Loki	Odinson	7758642509	30234289	0
	The Migthy	Thor	1132434159	42587412	0
	Obi Wan	Kenobi	12356987	44578963	0
	Anthony	Stark	6375125832	45038419	0
	Clint	Barton	9496175297	45421268	796,57007
	Bruce	Banner	9133370598	49827708	0
	Qui Gon	Jinn	918917891	51951981	0
	Steve	Rogers	5345408117	58135220	0
	Stephen	Strange	7363465932	83150463	0

Agregar pedidoModificar clientePagar pendienteAgregar clienteEliminar cliente

Si presionamos sobre el texto Ventas, este nos abrirá la aplicación de Notas con los datos de todas las ventas. Lo mismo con Clientes.

Eliminar cliente

Esta opción es bastante simple y deductiva, al seleccionar un cliente y presionar el botón, nos pide una confirmación. Si confirmamos la acción, se borra el valor de la base de datos y continúa con la ejecución.

Pagar pendiente

Esta acción es muy parecida a la anterior, tan solo que, en vez de eliminar al cliente, establece su deuda en \$0. Esta acción también pide una confirmación, y modifica el campo **debe** del cliente en la base de datos a 0.

Cerrar

No hay un botón *cerrar* como tal, pero podemos cerrar el programa desde la **X** en la esquina superior derecha. Al presionar este botón, nos pide una confirmación. Si decidimos cerrar, se escriben los datos de clientes en un XML y ventas en XML y .txt.

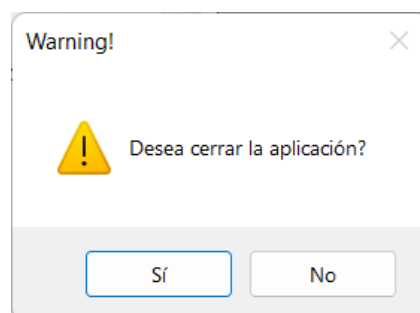
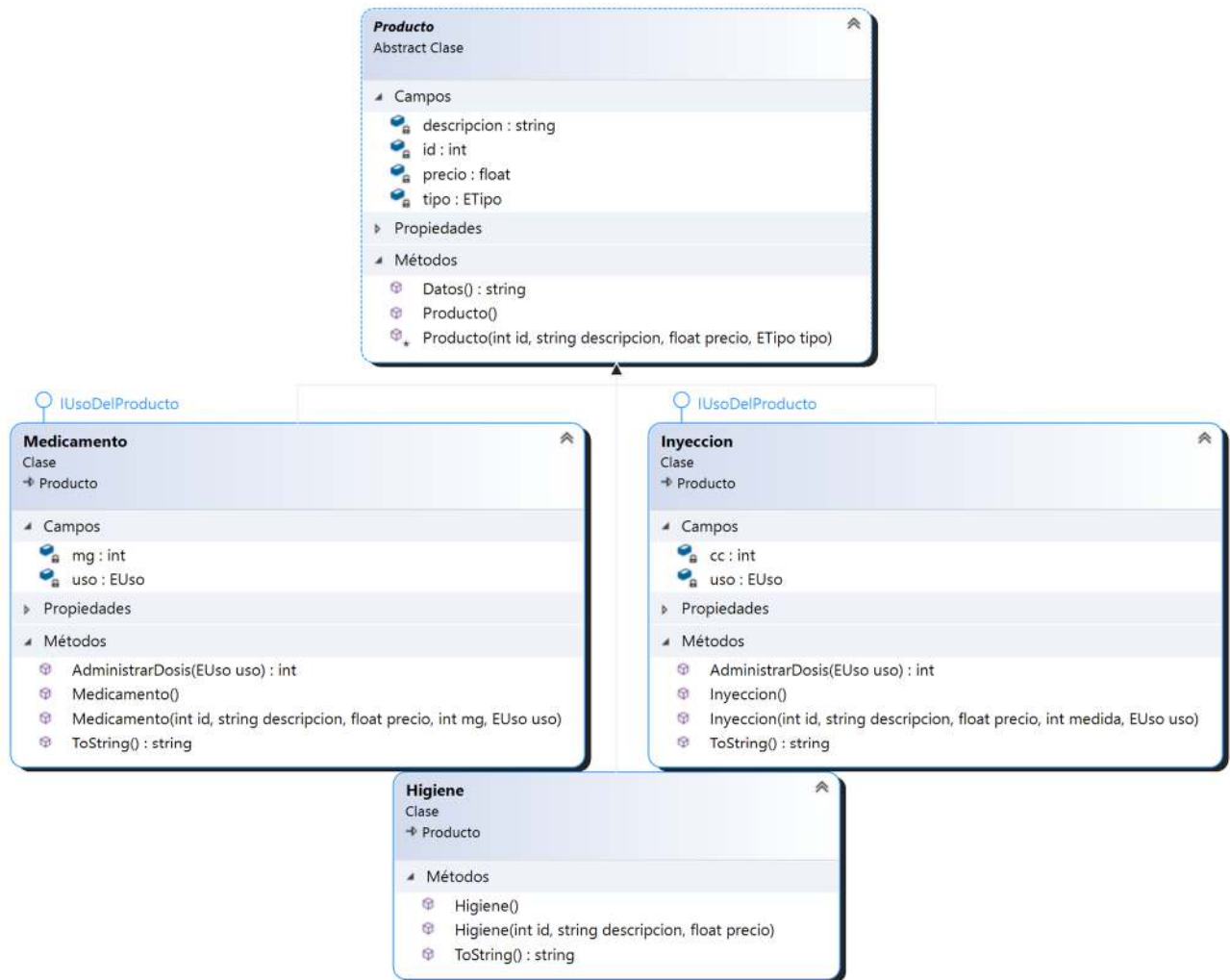
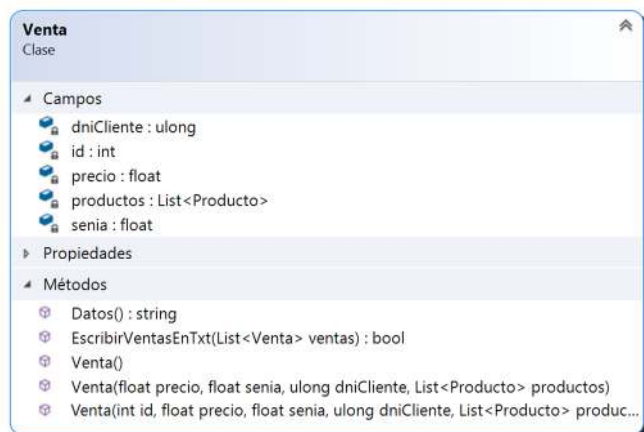


Diagrama de clases

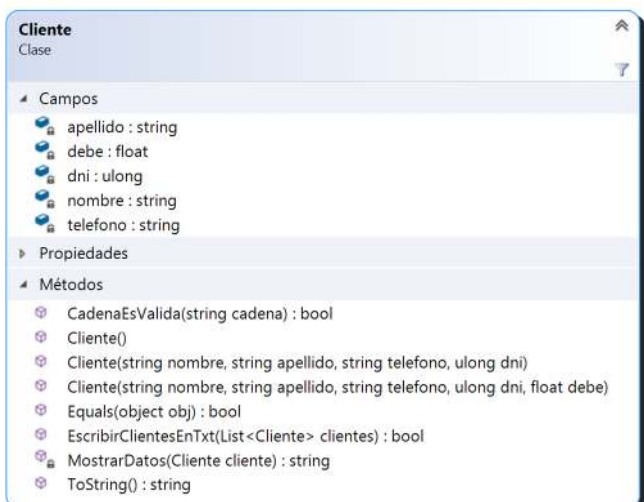
Producto y derivadas



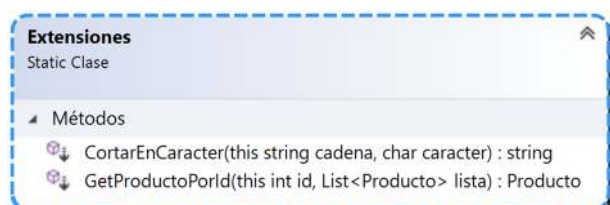
Venta



Cliente



Extensiones



Temas incluidos

Excepciones

Decidí centrar las excepciones en el objeto [Cliente](#), las cuales se encargan de que los datos de este sean válidos y no haya fallos más adelante. Respecto a estas excepciones, cree una clase del tipo *Exception* como clase padre, para luego derivar de esta las otras excepciones y así hacerlo más fácil de atrapar si se lanza alguna.

Estas excepciones pueden encontrarse controladas en *FrmPrincipal* y en *FrmAgregarCliente*.

Unit Testing

Dentro de la solución, se encuentra el proyecto **UnitTesting**, la cual tiene dos clases:

ClienteTest

Aquí realizo 3 pruebas de cuatro métodos de la clase [Cliente](#):

- Constructor de [Cliente](#) (2)
- CadenaEsValida

ProductoTest

En esta clase realizo dos test del mismo método de [Producto](#), [GetProductoPorId](#).

Generics<T>

El tipo genérico lo incluí en la clase **Serializadora**, la cual se encarga de la lectura y escritura de archivos. Decidí utilizarla allí para hacer más fácil la tarea de manejo de archivos y no hacer sobrecarga de métodos por solo un tipo de dato.

Interfaces

En este trabajo utilicé una sola interfaz, la cual tiene un método.

Esta interfaz la utilizo en dos [derivadas de Producto](#), **Inyección** y **Medicamento**.

Archivos y serialización

El manejo de archivos fue especialmente el tema más interesante, me encontré con varios problemas por los datos que contenía y las propiedades que requería. Finalmente pude resolver los problemas y terminé con tres archivos .XML:

- ListadoDeProductos,
- ListadoDeClientes,
- ListadoDeVentas;

y dos archivos .txt donde se muestran todas las ventas realizadas y los clientes respectivamente (se guardan en Documentos):

- ListadoDeVentas,
- ListadoDeClientes.

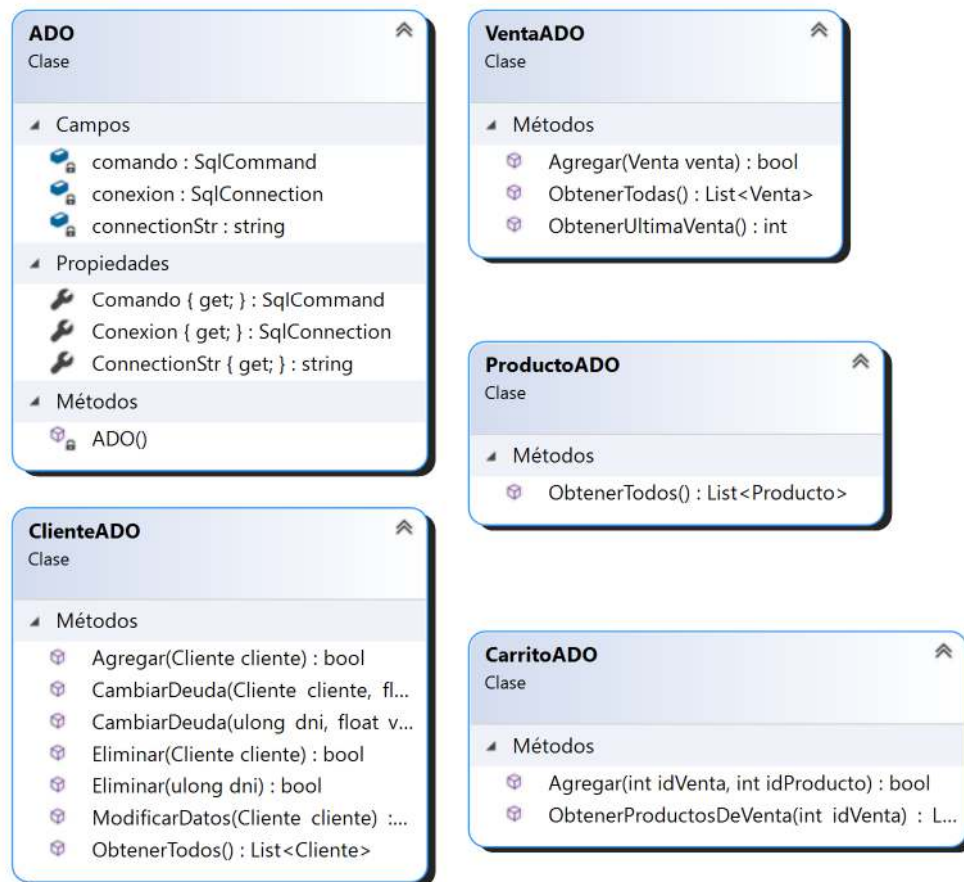
SQL y base de datos

En esta instancia de trabajo práctico decidí cambiar un poco lo hecho en el trabajo anterior. Los datos se almacenan en la base de datos **Farmacia**, con 4 tablas en vez de listas en la clase. Las tablas son:

- **Clientes:** Se guardan los datos de los clientes.
- **Productos:** Aquí se guardaría el "stock" de productos disponibles. Contiene un ID incremental.

- **Ventas:** Tiene 4 datos de las ventas (precio, seña, DNI del cliente y un ID incremental).
- **Carrito:** Esta tabla se encarga de conectar las últimas dos. Consta de un id para la venta y otro para el producto.

Estas tablas son manipuladas con sus clases respectivas dentro de la carpeta ADOs.



Eventos y delegados

El programa cuenta con dos eventos, los cuales decidí crearlos como tipo de dato de delegado. Los eventos que cree son los siguientes:

SeniaInvalida (FrmMontoSenia)

Delegado: `public delegate void DelegadoSeniaInvalida(float valor)`

Manejador: `public void ManejadorSeniaInvalida(float valor)`

AbrirArchivoTxt (FrmPrincipal)

Delegado: `public delegate void DelegadoAbrirArchivo(string nombreArchivo)`

Manejador: `private void ManejadorAbrirArchivoTxt(string nombreArchivo)`

Hilos

Decidí que, al invocar AbrirArchivoTxt, este realice la acción dentro de un **Task**. Ubicado en FrmPrincipal, es invocado cuando se presiona el label de clientes o el de ventas.

Métodos de extensión

Para finalizar, realicé 2 métodos de extensión, para facilitar el acceso a métodos bastante intuitivos.

- CortarEnCaracter(this string, char)
- GetProductoPorId(this int, List<Producto>)

Ambos en la clase Extensiones.