Trabajo Práctico 4°

División 2°D
Alberto Giffard

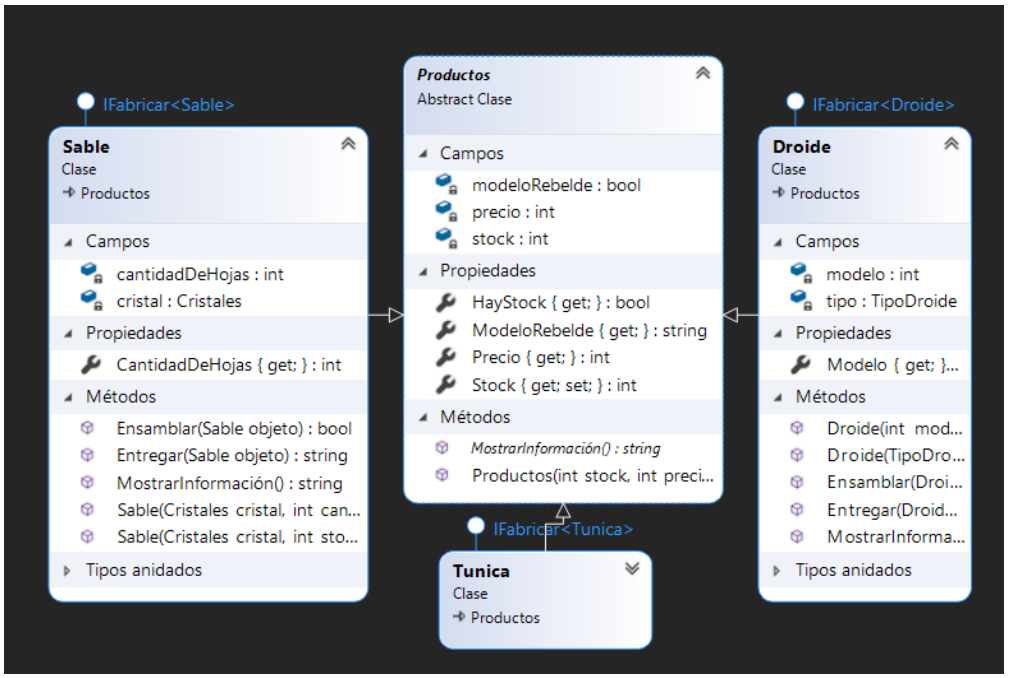
Se diseñó un sistema para una fábrica con temática de Star Wars el cual contará con tres productos para su elección y fabricación, los cuales son:

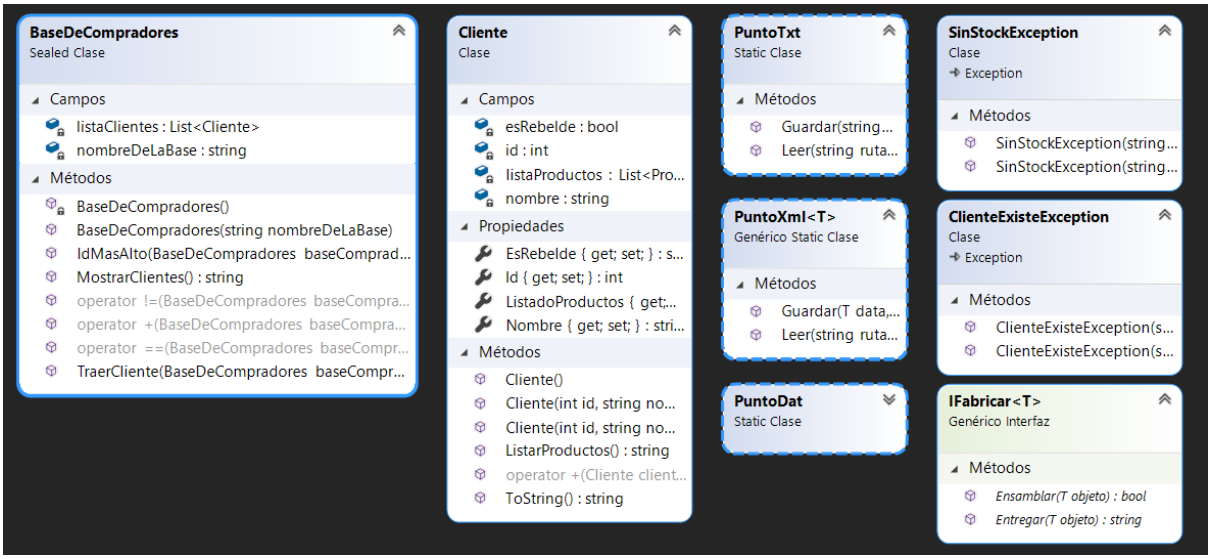
- Sables de luz
- Túnicas
- Droides

Cuenta con su propia lista de clientes, donde cada uno de ellos tendrá un documento único, el cual dentro del programa llevará el nombre de "**ID**" (el usuario no lo puede modificar ni escoger).
En el primer formulario se debe indicar si ya es cliente y a partir de ahí se podrá ir por el camino del formulario de alta de nuevo cliente o ir directo a realizar la fabricación.

Cada vez que un producto es agregado indicará a través de MessageBox el proceso de fabricación, ensamble, entrega y/o comentario.

La fábrica está compuesta por 1 solución, 4 proyectos, 13 clases, una interfaz y 3 formularios:





Excepciones

Se cuentan con distintas excepciones a lo largo de la fábrica pero, se cuentan con dos clases de excepciones creadas, las cuáles son:

1. "**ClienteExisteException**": Se lanza cuando se intenta agregar un cliente que ya existe dentro de la lista de clientes de la fábrica.
   Se implementa en:
   - FrmPrincipal.cs (línea 159)
   - BaseDeCompradores.cs (línea 112)
   - Proyecto Test, Program.cs (línea 42)
2. "**SinStockException**": Se lanza cuando se intenta agregar un producto que no tiene stock.
   - Proyecto Test, Program.cs (línea 59)
   - FrmProductos.cs (línea 175)
   - FrmProductos.cs (línea 223)
   - FrmProductos.cs (línea 261)
   - Productos.cs (línea 47)

Test Unitarios

En el proyecto "**ProbandoFuncionalidad**", se realizan distintas pruebas (un total de 9) para verificar el correcto funcionamiento de los métodos más complejos/usados.

Tipos Genéricos e Interfaces

Al ser productos tan distintos opté por realizar una interfaz que implementa los tipos genéricos, para que el valor T sea reemplazado por el tipo de las clases que lo implementan; por lo tanto encontramos su uso en:

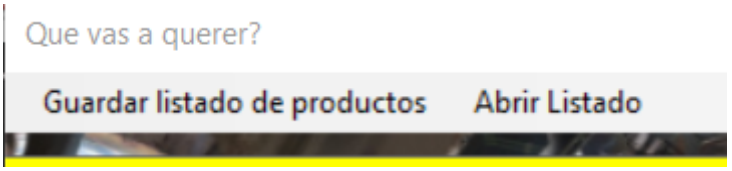
- IFabricar.cs
- Droide.cs (línea 12, 53, 70)
- Tunica.cs (línea 12, 42, 59)

- Sable.cs (línea 12, 54, 71)

Los métodos puede que hubiesen sido más lógicos como métodos estáticos, pero para fines prácticos y de la consigna opté por implementarlos de instancia.

Ejemplo únicamente de tipo genérico también se suma la clase: PuntoXml

Archivos
Se crean tres clases, una para los archivos de texto, binarios y xml. Todos cuentan con dos métodos uno que guarda y otro que lee, se implementan en la vista del formulario de productos en la barra superior izquierda.



1. PuntoTxt.cs
    - FrmProducto.cs (línea 308, 395)
2. PuntoDat.cs
    - FrmProductos.cs (línea 336, 422)
3. PuntoXml.cs
    - FrmProductos.cs (línea 364, 449)

Base de datos
Dentro de la clase “Conexion” se crean todos los métodos necesarios para comunicarse con la base de datos, específicamente en la tabla “TableTP4” con las dos tablas: Cliente, Producto; las cuáles son relacionales.

No se pueden eliminar clientes si este cliente cuenta con productos creados, por lo tanto se tendrán que eliminar primero los productos para poder eliminar al cliente de la base de datos.

1. Conexion.cs
2. FrmCliente.cs (Línea 80)
3. FrmPrincipal.cs (Línea 150, 247)
4. FrmProducto.cs (Línea 135, 164, 212, 249, 472)
5. PruebasUnitarias.cs (Línea 103, 113)
6. Proyecto Test, Program.cs (línea 146, 158, 169)

También a la par de donde se encuentra la solución se encuentra el .bak con la tabla realizada para el programa, lleva por nombre “TableTP4_AlbertoGiffard.bak” y se utilizó “.” para el Data Source. Si se quiere modificar, en la clase “Conexión” en la línea 20, se encuentra el string del Data Source.

Hilos / Threads
Se diseñó un texto en el formulario principal el cual nos irá informando en que paso/status nos encontramos el cuál será ejecutado en un hilo secundario, dependiendo de si estamos creando un nuevo cliente, queriendo fabricar, darnos de baja o incluso cerrando la aplicación.

- FrmPrincipal.cs (Línea 26, 91, 92, 101, 320)

Eventos / delegados
A través de un evento en la clase Cliente cuando se traiga el listado de productos del cliente así como cuando se setea el valor de si es rebelde o se suma un producto será invocado el método de ListarProductos() el cual trae en un variable de tipo String los datos de los productos.

- Cliente.cs (Línea 9, 17, 90, 100, 108, 176)
- FrmProductos.cs (Línea 137)

Métodos de extensión
Se diseñó un método el cual validará si el cliente es rebelde o no, en tal caso que no lo sea dirá “No (Es Sith)”, dando un valor agregado a la información de cada cliente.

- ExtensionCliente.cs
- Cliente.cs (Línea 78)

Notas

- Se dió una perspectiva de fábrica y venta, porque me pareció que sólo fabricación quedaba por la mitad, agregando así elementos como clientes, presupuesto, etc.
- El formulario principal cuenta con música para tener en cuenta por si se está usando audífonos o volumen alto.
- Con más de 90 horas invertidas, contando el tiempo de diseño y armado de lógica antes de incluso abrir el IDE, espero que les guste este trabajo práctico que con mucho gusto realice :)