Universidad Tecnológica Nacional
Facultad Regional Buenos Aires
Ingeniería en Sistemas de Información

AÑO 2018

EJERCICIOS PATRONES DE DISEÑO

Materia: Diseño de Sistemas

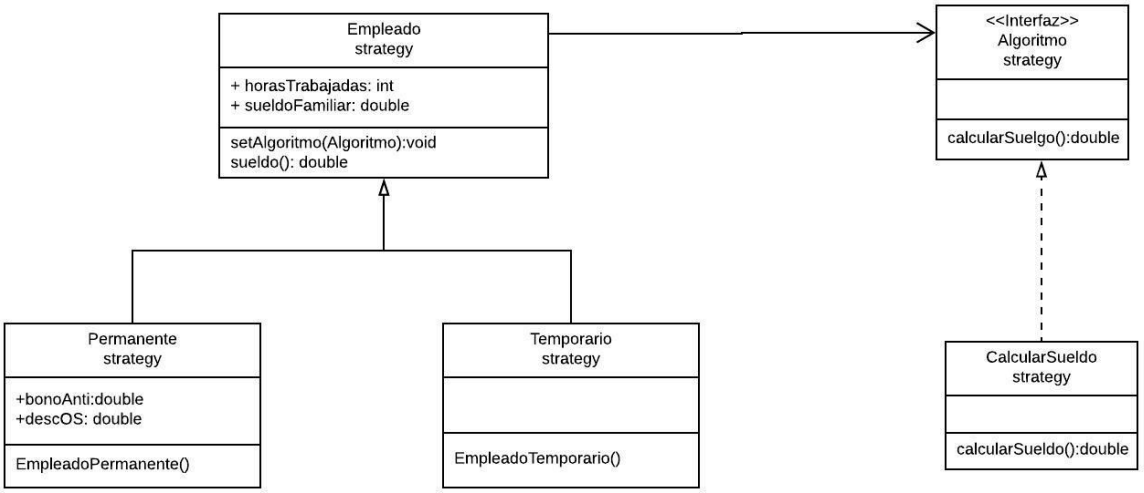
Docente:

Pablo Sabatino

Fecha prevista de entrega: 30/05/2018

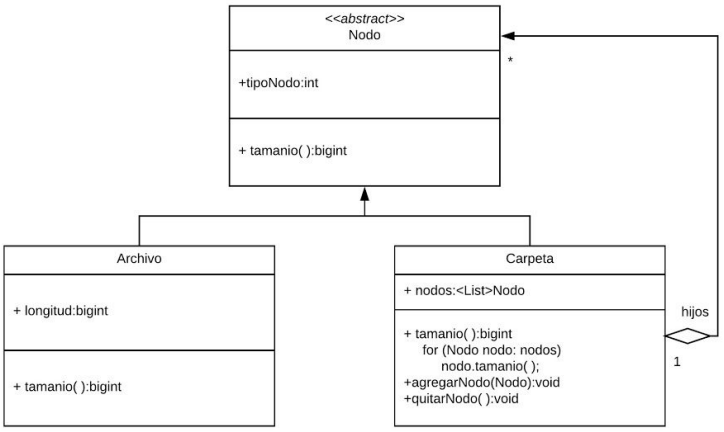
Nombre	Legajo
Angel Dario	144.506-6
Mendez Angelica	144.095-0
Nicora Facundo	163.093-3
Otero Alejandro	146.820-0
Peralta Liliana	143.242-4

Ejercicio 1:



Se eligió el patrón **Strategy** para calcular el salario según el tipo de empleado ya que dependiendo del tipo se emplean diferentes factores que se aplican al mismo.

Ejercicio 2:

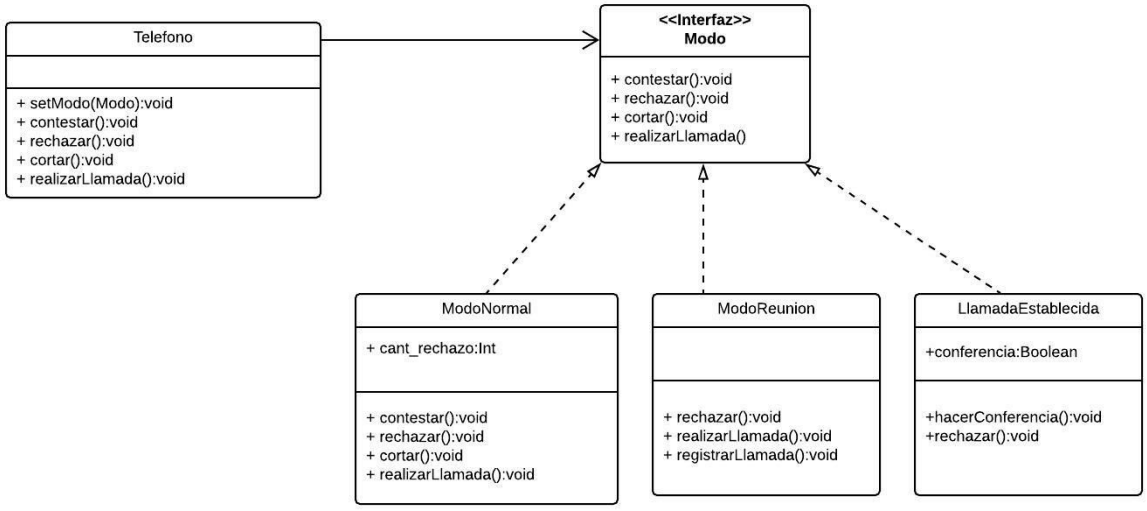


Elegimos **Composite** para este ejercicio ya que necesitamos componer los objetos de manera recursiva, es decir, que una carpeta puede contener otras carpetas y archivos. Los archivos serían las hojas del patrón y la carpeta el composite.

El método tamanio de Archivo simplemente devuelve su longitud. El método tamanio de Carpeta aplica recursividad sobre cada Nodo (Archivo o Carpeta) que contenga la carpeta en cuestión, es decir que si es un Archivo obtendrá su longitud y si es otra Carpeta obtendrá por recursividad la longitud de todos los archivos contenidos en ella.

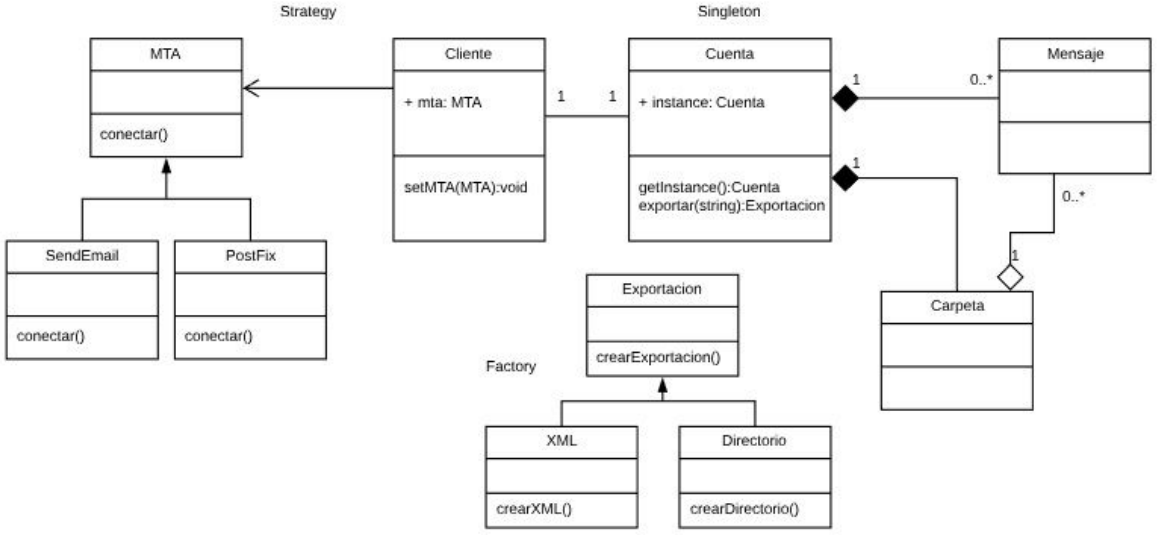
Aclaración: El código mostrado en el método tamanio de Carpeta no es el completo, sólo colocamos eso para mostrar la recursividad del mismo.

Ejercicio 3:



Elegimos **State** para éste ejercicio ya que según el modo en la que se encuentra el celular el comportamiento será uno u otro, es decir, el modo se actualizará según el uso que le da el usuario.

Ejercicio 4:

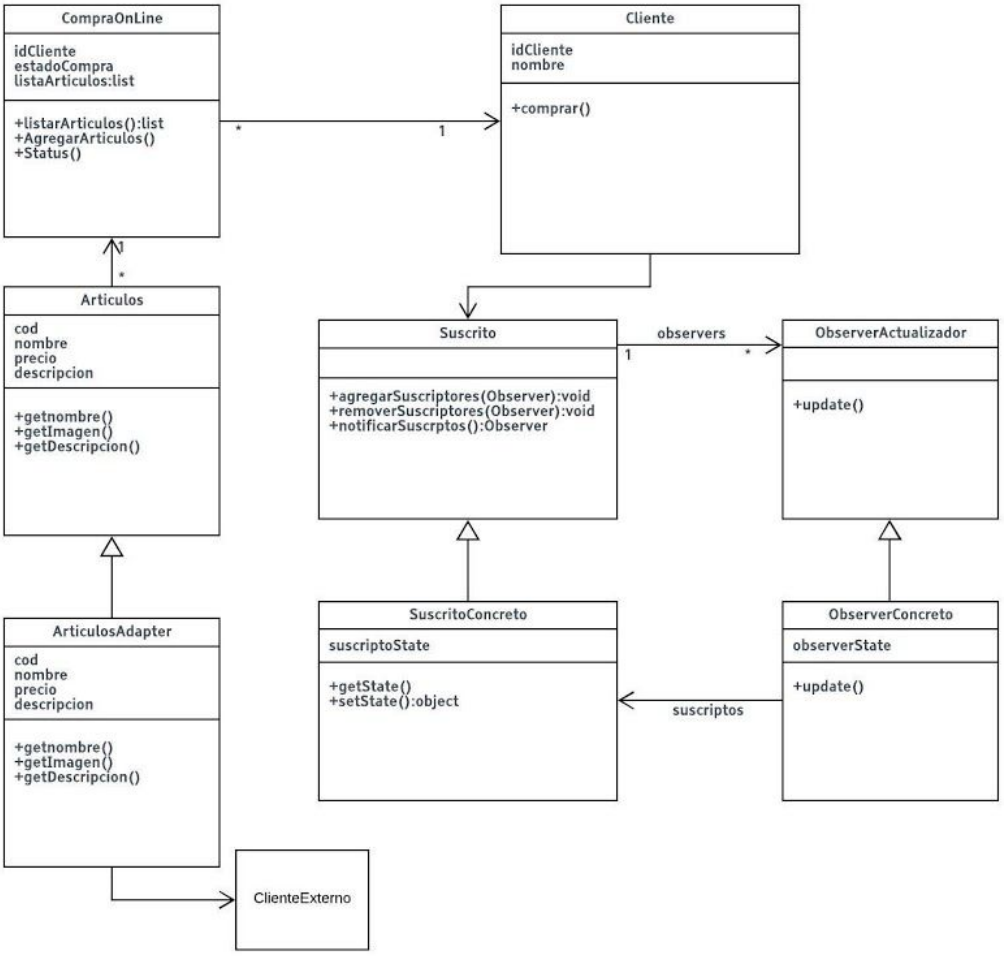


Se usó el patrón **Singleton** para la clase Cuenta ya que solo puede existir una única instancia de esta clase.
Se eligió el patrón **Strategy** para seleccionar la estrategia correspondiente al Mail Transfer Agent que el usuario haya elegido para el cliente.
Y se utilizó el patrón **Factory Method** para crear los diferentes formatos de exportación dependiendo la elección del usuario.

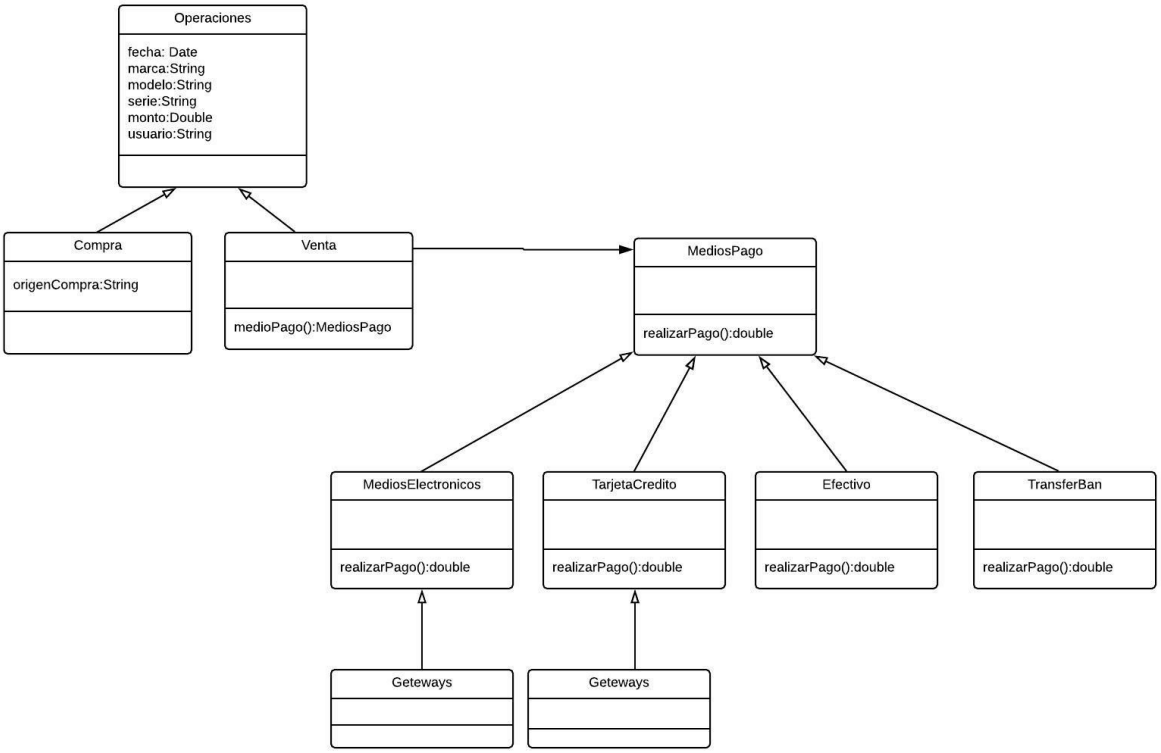
Ejercicio 5:

Para resolver la funcionalidad :
diseñar una solución donde sea el servidor quien notifique a los clientes (app) las promociones especiales y no los clientes quienes lo consulten si hay nuevas ofertas.
Utilizamos el Patrón **Observer**; ya que cuando se informa de una nueva Promoción, este es el encargado de notificar a los usuarios suscritos.

Detectamos el uso del patrón **Adapter**, para recuperar los artículos de los servicios web expuestos por los proveedores.
Aclaraciòn: Exponemos los mètodos utilizados para el uso de los patrones.



Ejercicio 6:



Elegimos usar una **herencia** de clase para las operaciones que se pueden realizar en el sistema ya que muchos de los atributos son compartidos sin importar la operación al que corresponde. Luego un **Factory Method** para las formas de pago, ya que se implementara el pago seleccionado por el usuario.