INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS CIUDAD DE MÉXICO

PATRONES DE DISEÑO

DE LA ASIGNATURA

DISEÑO Y ARQUITECTURA DE SOFTWARE

PREPARADO POR

Jesús Norberto Reyes Muñoz A01651207
Ernesto Iván Ochoa Hidalgo A01651425
Rafael García Cadenas A01334363
Abraham Armando Silva Apanco A01651490

marzo 2019

Problemática

Se debe realizar los objetos de negocio para una empresa de paquetería. Otro equipo aparte (desconocido) está realizando la GUI.
Los envíos contienen un ID único, una dirección de entrega, una dirección del remitente, un código postal de entrega, un código postal del remitente y el peso.
Un cliente *n* cantidades de envíos. El cliente hace relación a un RFC que es una dirección fiscal y una razón social.
México es una país muy grande por lo que hay diferentes proveedores que se subcontratan dependiendo de la región para la entrega.

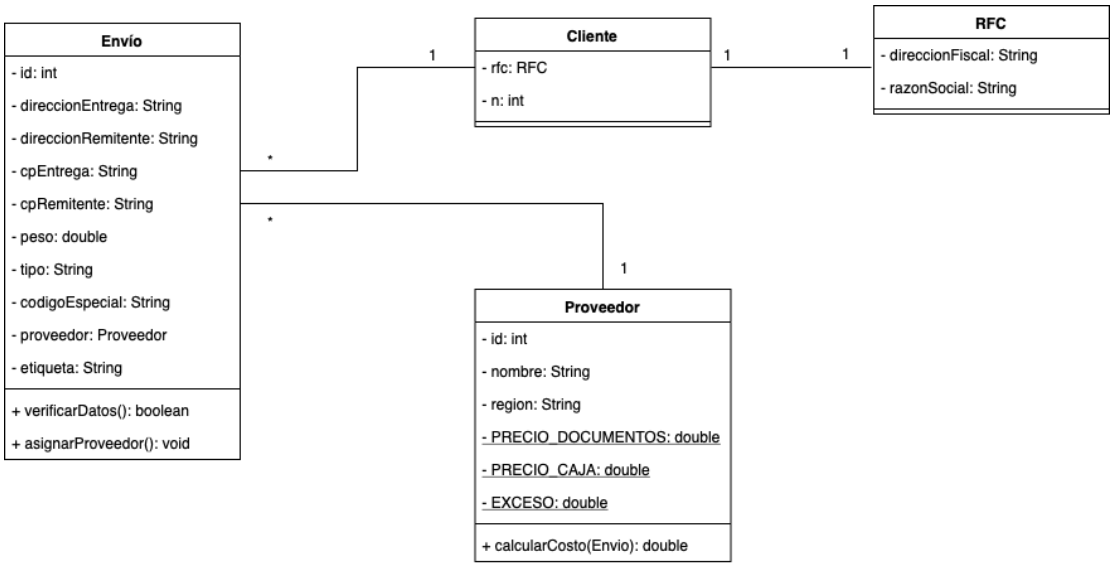
- EnvioExpress: localizado en sonora.
- BombaEnvios: Localizado en Yucatán.
- PastesGo: Localizado en Pachuca.

Hay diferentes tipos de envío y dependiendo del proveedor hay un precio diferente por 250 gramos.

	EnvíoExpress	BombaEnvíos	PastesGo
Documentos	39	42	59
Caja	25	20	19
Exceso de dim.	500 por cc extra	Sin cargo	600 por cc extra

Los envíos pueden estar marcados por código especiales: Frágiles, firma en contra entrega, no dejar si no se encuentra la persona. Puede haber una combinación de uno o más de estas etiquetas. Un envío solo puede estar asignado a una ruta en especial por día.

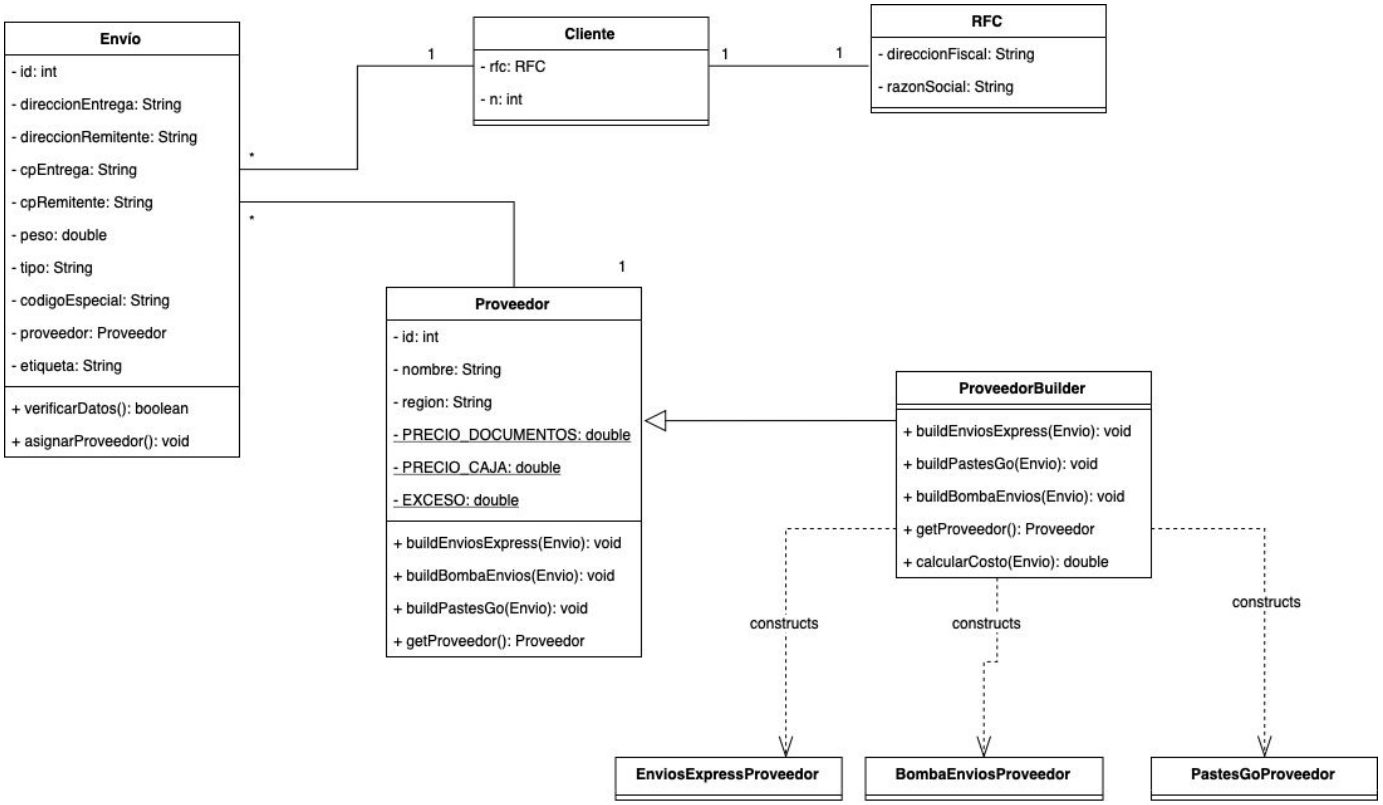
Diagrama base

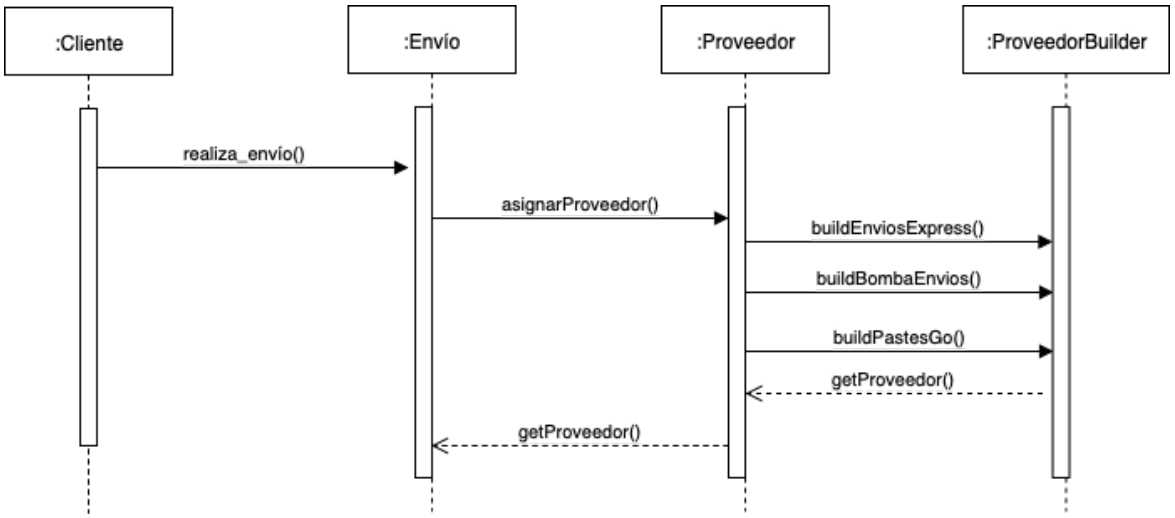


Solución con patrones de diseño

A continuación, se muestran los patrones de diseño utilizados para resolver la problemática propuesta. Se muestra la documentación a través de diagramas de clase y secuencia, así como la justificación a la selección de cada patrón.

Creational Design Pattern: *Creational*



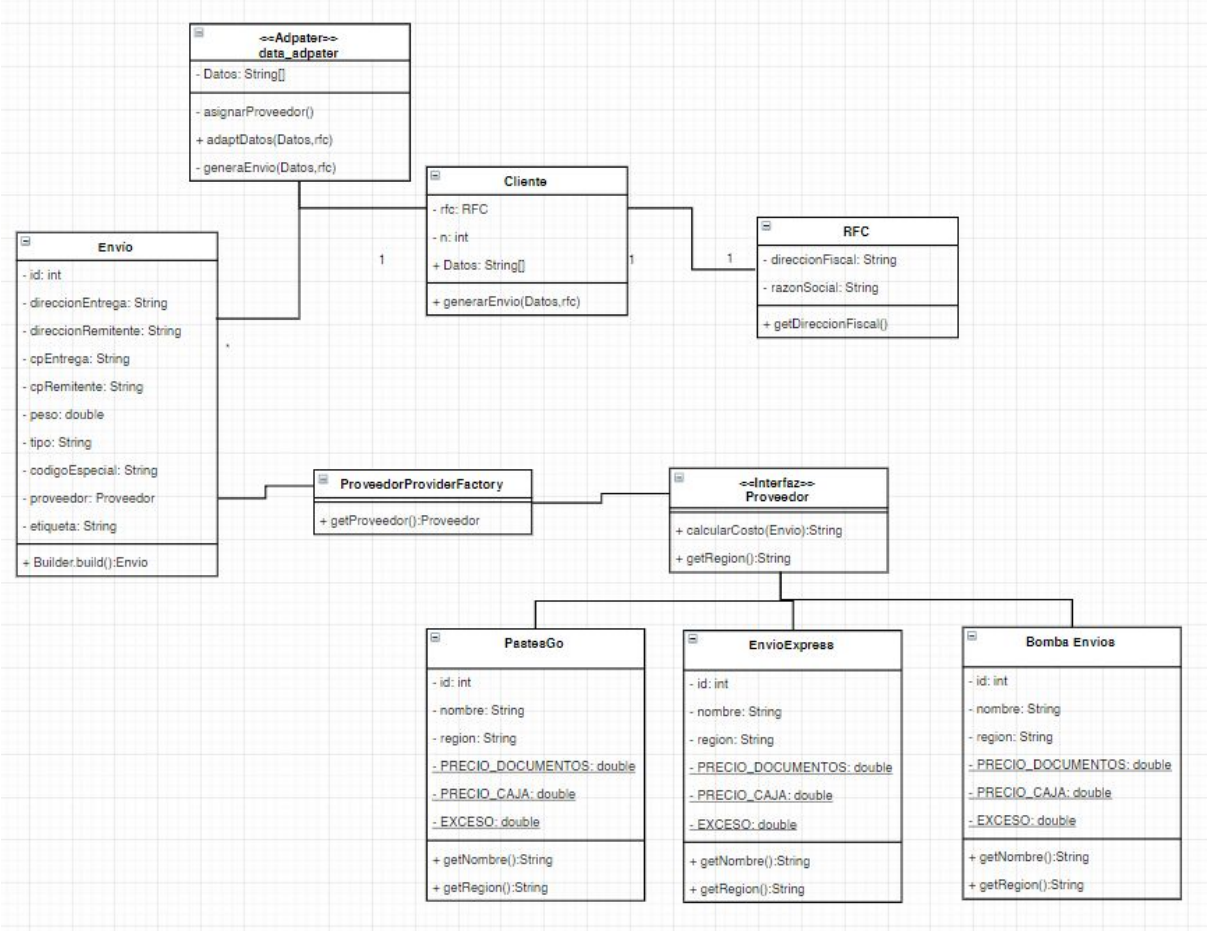


El patrón de diseño Factory, está pensado para poder separar la complejidad de la construcción de un objeto complejo, de modo que su representación pueda ser más sencilla y dividida en diferentes partes. En este caso, la entidad Envio está realizada con el creational pattern de builder porque separamos cada representación del envío de su instanciación debido a que cada uno puede tener etiquetas diferentes que los convertiría en envíos distintos Y la entidad Proveedor puede tener diferentes representaciones derivadas de los diferentes escenarios que pueden ocurrir por la naturaleza de la aplicación, es decir, si se realiza un envío de documento desde una dirección cercana a la región de Yucatán, con un código especial de firma a la entrega y sin exceso de dimensiones, entonces la instancia de un Proveedor será distinta a la de otro envío con otras características particulares.

Las tres subclases de Proveedor (EnvioExpressProveedor, BombaEnviosProveedor y PastesGoProveedor) heredan de la clase Proveedor los parámetros y método, sin embargo, el método se particulariza para cada una de ellas por los costos que maneja cada uno de esos proveedores, esto hace que el cálculo del costo de envío sea diferente para cada proveedor.

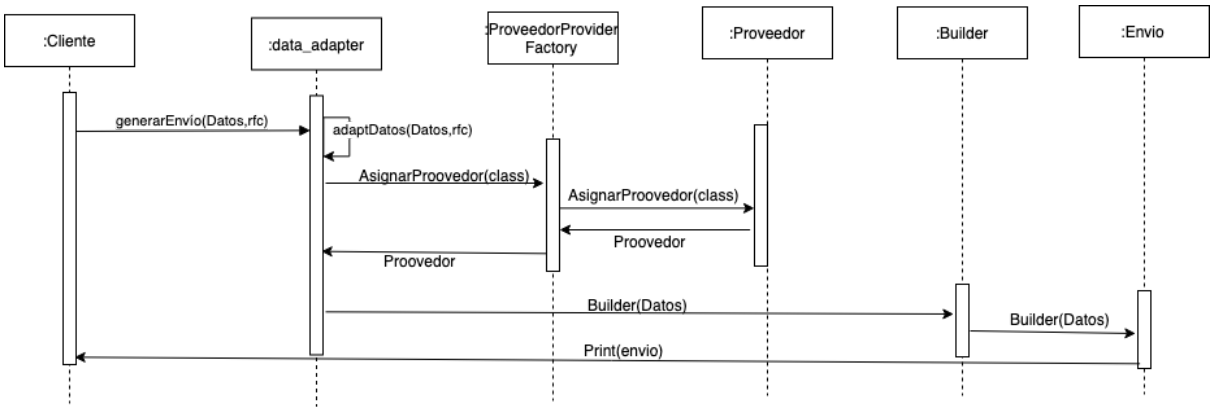
Este patrón de diseño, nos está permitiendo tener una mejor visibilidad sobre cuáles son los diferentes proveedores que puede haber, dependiendo de las variantes del envío, y hacer más simple la entidad proveedor. Incluso, sabiendo que seguimos en el diseño de la arquitectura, no sólo este sino todos los patrones, nos auxilian a representarla de una forma más simple y entendible a aquellos no familiarizados con el tema.

Structural Design Pattern: *Adapter*

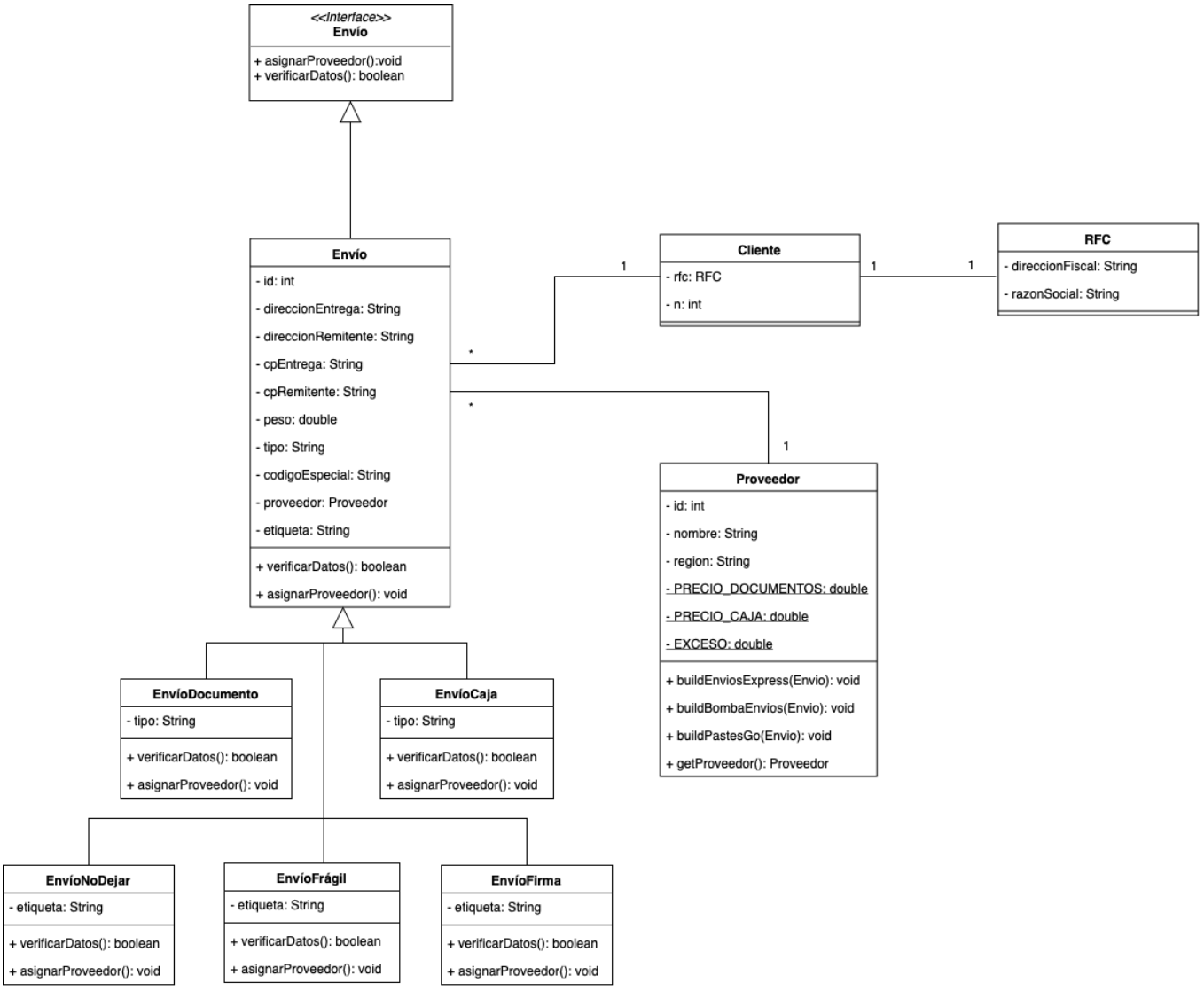


Dado a que no es posible tener una entidad Envío para cada cliente, es necesario tener una manera de poder ajustar los datos que proporcione el cliente, a los formatos de los datos requeridos por nuestro Envío.

Implementando el patrón de diseño estructural “Adapter”, nos es posible crear un Adapter, el cual recibe los datos y el rfc del cliente a hacer el envío, los normaliza y genera el envío ya con los datos ajustes al formato requerido por el objeto Envío. Aún implementando el patrón de diseño adaptativo, resulta conveniente mantener el patrón de diseño Builder, con el fin de eficientar la manera de seleccionar un proveedor.


:Cliente
:data_adapter
:ProveedorProvider Factory
:Proveedor
:Builder
:Envio
generarEnvío(Datos,rfc)
adaptDatos(Datos,rfc)
AsignarProovedor(class)
AsignarProovedor(class)
Proovedor
Proovedor
Builder(Datos)
Builder(Datos)
Print(envio)

Structural Design Pattern: *Decorator*



Dado que los envíos pueden tomar diferentes representaciones de acuerdo con lo señalado en su etiqueta o en su tipo, por ello, el patrón *decoration* resulta ser muy adecuado al implementar una interfaz que las demás clases la extiendan y éstas a su vez puedan subdividirse para representar cada uno de los posibles tipos de envíos.

En el momento de la implementación a través del código, se puede tener una mejor organización y visibilidad de los objetos envío.